Contribution à la gestion des systèmes de stockage, une approche verticale : de l'architecture à l'applicatif

Jalil Boukhobza

HDR / UNIVERSITÉ DE BRETAGNE OCCIDENTALE

sous le sceau de l'Université Bretagne Loire

Mention : Informatique

École Doctorale SICMA

présentée par

Jalil Boukhobza

Préparée au Lab-STICC UMR 6285

Contribution à la gestion des systèmes de stockage, une approche verticale : de l'architecture à l'applicatif

Thèse soutenue le 4 juillet 2017

Devant le jury composé de :

M. Luc BOUGANIM,
Directeur de recherche (DR1) INRIA/ Rapporteur

M. Daniel HAGIMONT,
Professeur, INPT/ENSEEIHT / Rapporteur

M. Pierre SENS,
Professeur, Université de Paris 6 / Rapporteur

M. Zili SHAO,
Professor associé, Université Polytechnique de Hong Kong / Examinateur

M. William JALBY,
Professeur, Université de Versailles / Examinateur

M. Jean-Philippe DIGUET,
Directeur de recherche CNRS / Examinateur

M. Jean-Philippe BABAU,
Professeur à l'Université de Bretagne Occidentale / Examinateur

M. Frank SINGHOFF,
Professeur à l'Université de Bretagne Occidentale / Examinateur

A la mémoire de mon père

A ma famille

Contribution to storage management, a vertical approach: from architecture to application

Remerciements

Je voulais tout d'abord adresser mes chaleureux remerciements aux membres du jury pour avoir accepté d'évaluer ce travail. En particulier, je remercie les trois rapporteurs, M. Luc Bouganim, Directeur de Recherche à l'INRIA Rocquencourt, M. Daniel Hagimont Professeur à l'INPT/ENSEEIHT Toulouse, et M. Pierre Sens, Professeur à l'Université de Paris 6, pour leur confiance et pour avoir lu, analysé et évalué consciencieusement ce document. Je tenais aussi à remercier tout particulièrement M. Zili Shao, Professeur associé à l'Université Polytechnique de Hong Kong, qui a accepté de venir de très loin pour 2 jours à Brest afin d'assister à ma soutenance. Aussi, j'en profite pour le remercier pour son accueil au sein de son laboratoire pendant 6 mois en 2016. Ce fut une expérience formidable. Merci aussi à M. William Jalby, Professeur à l'Université de Versailles Saint Quentin en Yvelines pour sa disponibilité et pour avoir accepté de présider le jury. Enfin, j'adresse un grand merci aux collègues Jean-Philippe Babau et Jean-Philippe Diguet pour leur soutien tout le long de ma carrière au département informatique et au Lab-STICC et pour avoir accepté d'examiner ce rapport.

Aucune des contributions décrites dans ce document n'aurait pu aboutir sans le support et l'aide conséquente de plusieurs collègues. Un travail de recherche est, de mon point de vue, par essence un travail collaboratif qui se nourrit de l'expérience et de la connaissance de chacun. Ainsi, il ne peut être que plus riche et fructueux. Plusieurs collègues ont grandement contribué au murissement de ce travail et à son aboutissement. Je suis particulièrement reconnaissant à Frank Singhoff, Stéphane Rubini, Laurent Lemarchand, Jean-Philippe Babau et Eric Senn. J'adresse aussi ma gratitude aux collaborateurs extérieurs avec qui j'ai beaucoup travaillé et avec qui je continue à explorer des problématiques intéressantes, je pense à Kamel Boukhalfa et à Yassine Hadjadj Aoul.

Nous ne pouvons consacrer du temps à la recherche qu'à condition que certains collègues prennent du leur pour assumer les tâches administratives parfois lourdes et qui rendent service à tous. Je pense à tous les directeurs de département qui se sont succédés depuis mon recrutement, Philippe Le Parc, Vincent Rodin, Frank Singhoff, Jean-Philippe Babau, et maintenant Alain Plantec, aux directeurs de site du laboratoire Laurent Nana et Emanuel Radoi et aux collègues responsables de filière ou d'année, chevilles ouvrières du département, Catherine, Damien, Erwan, Goulven, Mickaël, Laurent, Mounir, Philippe… Merci aussi à ces collègues et à d'autres qui contribuent à la bonne entente au sein du département. Un merci aussi aux collègues de l'IRT avec qui j'apprécie de travailler depuis 4 ans maintenant.

La liste ne peut être complète sans remercier ceux qui ont mis sur pied la plupart des travaux de recherche décrits dans ce document, les étudiants en thèse. J'exprime ainsi ma gratitude à Pierre Olivier, qui a beaucoup contribué au montage de cette thématique, Yahia Benmoussa, Cheikh Salmi, Nam Hải Trần, Hamza Ouarnoughi, Arezki Laga, Islam Mohammed Naas, Ydroudj Assia, Djillali Boukhelef, Jean-Emile Dartois et Amina Chikhaoui. J'ai beaucoup appris de chacun d'entre vous.

Rien de tout cela n'aurait été possible sans le support de ma famille, Assil, Anas, Linda, ma mère, Taha, Hasna, Nora, et Djelloul. Ma gratitude envers vous est infinie.

Enfin, je présente toutes mes excuses à tous ceux qui se seraient sentis lésés car oubliés sur cette liste, je vous remercie et vous demande de bien vouloir excuser cette maladresse.

Document Outline

How to read this report

The following document is organized in 4 parts:

- **Part 1:** Contains a full CV and a summary **in French** of the research conducted since I was recruited at the Universty of Western Brittany (Université de Bretagne Occidentale, referenced as UBO). This part contains 3 sections:
  - A CV;
  - A full summary of the research conducted: different research topics tackled, research projects; collaborations, students' supervision, and scientific activities. This section introduces to Parts 3 and 4;
  - Professional services and administrative responsibilities;
- **Part 2:** This part contains a full summary of the research achieved in UBO and introduces to Parts 3 and 4. Part 2 is written **in English** and is mainly a translation of section 2 of Part 1 for non-French speakers. This part concludes with a detailed outline of Part 3 and 4 in page 55.
- **Part 3:** describes the achieved work already summarized in Part 1 and Part 2. It details three of the past contributions we made. It is structured in 3 chapters. Each chapter is related to a specific topic tackled.
- **Part 4:** describes current work and perspectives already summarized in Part 1 and Part 2. It contains 3 chapters, 2 of them are related to current projects and one is about perspectives for future work.

Part 1 and Part 2 enclose the same **self-contained research summary**. Part 1 is more exhaustive as it covers a full CV in addition to a description of teachings and professional activities (a brief English version of the CV can be found in Appendix 1).

In Part 3 and Part 4, each tackled research topic is covered in an independent chapter in order to facilitate the reading. More details about the outline of these parts are given in page 55, once the research topics have been introduced.

Part 1: CV and summary (in French)

CV RÉSUMÉ

1.1 Etat civil

Jalil BOUKHOBZA, 38 ans

Adresse professionnelle: Université de Bretagne Occidentale, Laboratoire Lab-STICC, Département d'Informatique 20 Avenue Victor Le Gorgeu, BP808 Brest 29285, FRANCE

☎ 02 98 01 69 73 (bureau), 06 60 20 28 32 (personnel)
02 98 01 80 11 (fax)
@ boukhobza@univ-brest.fr
http://syst.univ-brest.fr/boukhobza

1.2 Situation actuelle

Grade: **Maître de conférences** depuis le 1er septembre 2006, Classe normale, 6ème échelon (depuis le 1er mars 2015)

Section: 27ème (Informatique)

Emploi: n°3416 (numéro national 1143) à l'Université de Bretagne Occidentale, Département d'Informatique, UFR Sciences et Techniques, Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC, CNRS UMR 6285)

Bénéficiaire d'une **Prime d'Excellence Scientifique** depuis septembre 2013.

Mis à disposition à l'IRT (Institut de Recherche Technologique) b<>com à hauteur de 20% du temps de travail.

1.3 Formation

Oct. 2000 – Déc. 2004 : Thèse de doctorat en informatique de l'Université de Versailles Saint Quentin en Yvelines au sein du laboratoire PRiSM:
Sujet : « **Etude et Analyse des Performances et Simulation des Accès aux Fichiers sous PC** »
Directeur de thèse : Pr Claude Timsit
Membres du jury :

M. Jean-Michel Fourneau:	Président du jury	Professeur à l'université de Versailles
M. Jean-Marc Delosme:	Rapporteur	Professeur à l'université d'Evry
M. Abdelaziz Mzoughi:	Rapporteur	Professeur à l'université Paul Sabatier
Mme. Wei Monin:	Rapporteur	Experte senior France Telecom R&D

Mention : Très Honorable
Financement : contrat de recherche avec France Telecom R&D

Oct.1999 – Sep.2000 : Diplôme d'Etudes Approfondies M.I.S.I (Méthodes Informatiques des Systèmes Industriels), Université de Versailles Saint Quentin en Yvelines
Option: « Parallélisme ».
Mention et rang : Assez bien, 2ème de la spécialité
Sujet de stage: « **Gestion des configurations de systèmes de stockage** ». Responsable: Pr. Claude Timsit

Sep.1994 – Sep.1999 : Diplôme d'ingénieur d'état en électronique, option informatique industrielle, Université de Boumerdès, Institut d'Electricité et d'Electronique (ex I.N.E.L.E.C), Algérie.
Option: « Informatique Industrielle ».
Mention et rang : spécialité: Bien, 3ème, Ingéniorat (les 5 ans): Assez bien.
Sujet de projet : « **Parallel Processing In Graphics** » Responsable: M. Abbes Bouklachi.

1.4 Expériences professionnelles

Aou. 2016 – Jan. 2017 Chercheur invité à **l'Université Polytechnique de Hong Kong** dans le cadre d'une **CRCT**.
Depuis Déc. 2013 : Mis à disposition à l'IRT (Institut de Recherche Technologique) b<>com à hauteur de 20% du temps de travail.

Depuis Sep. 2006 : Maître de conférences en informatique (sec. 27) à l'Université de Bretagne Occidentale, département informatique, UFR Sciences et Techniques, Lab-STICC UMR 6285.

Sep. 2004 – Aou. 2006 : Attaché temporaire d'enseignement et de recherche (ATER) à l'Université de Versailles St Quentin en Yvelines, département informatique, UFR Sciences et Techniques, laboratoire PRiSM UMR 8144.

Sep. 2000 – Déc. 2004 : Doctorant et vacataire à l'Université de Versailles St Quentin en Yvelines, département informatique, UFR Sciences et Techniques, laboratoire PRiSM UMR 8144.

1.5 Enseignement

Le tableau ci-dessous récapitule les enseignements dispensés tout au long de ma carrière. Plus de détails seront fournis dans la section suivante. Les enseignements en gras sont ceux pour lesquels je suis responsable d'UE (ou de partie d'UE dans le cas d'Architecture et Système 2).

1.6 Résumé des activités d'enseignement

Tableau 1. Récapitulatif des enseignements. Les enseignements suivis d'une étoile sont ceux pour lesquels j'ai élaboré la majorité, voire l'ensemble des supports de cours.

Formation concernée	Année	Enseignements dispensés /proportion approximative du cours que j'ai créé.	Département d'enseignement	Nb d'heures / nb d'années enseignées/années
1er cycle / Licence	1ère année	C2I / 0%	UFR Droit (UBO)	~20h/2/2007-2009
			UFR Sciences et techniques (UBO)	~20h/2/2007-2009
		Applications de l'informatique (réseaux et internet) / 0%	UFR Sciences et techniques (UBO)	~10h/2/2007-2009
	2ème année	**Architecture et système 1*** (Microprocesseur, assembleur, microcontrôleur, système) / **~70%**	Dept. Informatique (UBO)	~60h/12/2006-*
		Opérateur de calcul / 0%		~25h/1/2006
		Langages de programmation / 0%		~30h/2/2006-2008
	3ème année	**Architecture et Système 2*** (système d'exploitation, architecture et langage VHDL)/ **~50%**	Dept. Informatique (UBO)	~30h/12/2006-* (partie Système) ~20h/2/2006-2008 (partie Architecture)
		Algorithmique et langage C/0%		~20h/1/2007
		Microprocesseurs*/100%		~20/7/2008-*
		Architecture numérique/0%	Dept. Informatique (UVSQ)	~40h/5/2000-2005
2ème cycle / Master	2ème année	**Système d'exploitation pour l'embarqué*/100%**	Dept. Informatique – M2 Logiciels pour les Systèmes Embarqués	~50h/8/2008-*
		Introduction à l'embarqué**/100%**		3h/3/2013-*
Ecoles d'ingénieurs	2ème année	Introduction aux systèmes d'exploitation et Langage C/10%	ENSTA Bretagne	~30h/2/2008-2010
			ISTY-Versailles	~60h/5/2000-2005
	3ème année	Systèmes d'exploitation et Linux embarqué**/100%**	ENSTA Bretagne	~15h/3/2012-2015
	2ème année	Système d'exploitation pour	Dept. Systèmes	~15h/6/2011-*

		l'embarqué***/100%**	Electroniques. Embarqués, ENSEIRB, Bordeaux	
Etranger	1ère année de Master	Systèmes temps réel (**dispensé en anglais)/~30%**	Univ. Science et Technologie de Hanoï, Vietnam	~25h/1/2015
	2ème année de Master	Système d'exploitation***/100%**	Univ. de Tlemcen, Algérie	~20h/1/2012
	1ère et 2ème année ingénieur	Hiérarchie mémoire***/100%**	Ecole Supérieure d'Informatique (ESI), Algérie	4h/2/2016-*

1.7 Activités d'enseignement

Mes activités d'enseignement ont commencé en 2000, dès ma première année de thèse, et ont convergé vers des thématiques liées à l'architecture des machines et aux systèmes d'exploitation à différents niveaux. J'ai enseigné pour plusieurs publics différents : établissements universitaires, écoles d'ingénieurs ainsi que dans deux universités ainsi qu'une école d'ingénieur à l'étranger. J'ai principalement réalisé mes enseignements dans des départements d'informatique, mais aussi des départements d'électronique (tel que celui de l'ENSEIRB, IPB, Bordeaux).

Dans les paragraphes qui vont suivre, je résumerai l'ensemble de mes activités d'enseignement en décrivant les enseignements fait à l'UBO et dans d'autres établissements. Je rappellerai par la suite les enseignements réalisés pendant ma thèse, enfin j'aborderai un point qui me parait très important qui est la formation par la recherche.

1.7.1 Activités d'enseignement à l'Université de Bretagne Occidentale

Dans cette section, je décrirai les Unités d'Enseignement (UE) majeures dans lesquelles je me suis le plus impliqué et dont je continue à assurer l'enseignement au sein du département informatique, et ce dans les deux filières du département : ingénierie et scientifique (filières qui disparaitront à la prochaine rentrée). J'évoquerai ensuite les UE que j'ai enseignées occasionnellement.

1- **Architectures et Systèmes 1**, Licence 2ème année, environ 60 heures[1].
2- **Architectures et Systèmes 2**, Licence 3ème année, environ 30 heures.
3- **Systèmes d'Exploitation pour l'Embarqué**, Master 2ème année, environ 48 heures.

1. Architectures et Systèmes 1 (20h CM, 20h TD, 20hTP, cours enseigné depuis 2006, environ 70% des supports de cours/TD/TP crées) : L'objectif de cette UE est d'enseigner aux étudiants les principes de base d'un microprocesseur et d'introduire la notion de système d'exploitation. J'ai divisé cette UE en trois parties :

- Une partie conception de microprocesseur, dans laquelle je reprends les éléments de base de l'électronique numérique avant d'aborder les méthodologies de conception de microprocesseurs simples. Cela est fait en deux temps : un premier consacré à l'élaboration de circuits arithmétiques simples, à savoir des additionneurs, soustracteurs, et multiplieurs de différents types, et un deuxième temps centré sur l'élaboration d'unité de contrôle au travers la modélisation avec des automates d'états finis. Pour cette partie, en TP, les étudiants sont amenés à élaborer un processeur simple, qui supporte un jeu d'instructions réduit, de bout en bout, grâce à un simulateur de portes logiques. Pour chaque TP, les étudiants ont un cahier des charges concernant le ou les composants à développer et sont évalués la semaine suivante au travers d'une démonstration de quelques minutes par étudiant ou par groupe d'étudiants. J'accorde dans mes UEs, de manière générale, une très large part de contrôle continu.
- Une deuxième partie de développement en assembleur permet de compléter ce qui a été vu en première partie. Lors de mes première années d'enseignement de ce cours, j'utilisais l'assembleur 68000, qui même si très ancien, permettait de très vite être autonome et a l'avantage de présenter une sémantique très proche à celle du langage C. Ces dernières années, j'ai introduit l'assembleur ARM en TP et j'ai démarché l'entreprise ARM afin de bénéficier d'un don de carte microcontrôleur ARM. Cela me permet : (1)

[1] Sur cette liste, le nombre d'heures est donnée pour un groupe TD et un de TP

d'utiliser un assembleur plus récent et d'actualité (même si je continue à faire quelques séances de 68000), (2) de développer avec des outils industriels, (3) de s'appuyer sur des plateformes réelles, ce qui est perçu très positivement par les étudiants.

- Une troisième partie dans laquelle je monte en niveau d'abstraction en introduisant la notion de systèmes d'exploitation avec des TP simples en langage C.

2. Architectures et Systèmes 2 (10h CM, 10h TD, 10h TP, **cours enseigné depuis 2006, environ 50% des supports de cours/TD/TP crées) :** Cette UE constitue une suite de l'UE précédente pour les deux parties, « architectures » et « systèmes ». Pour la partie architecture, dont j'ai assuré les TD et TP pendant 2 ans, il s'agit principalement d'introduire la notion de pipeline dans les architectures de microprocesseur et d'en faire une implantation simple en TP en utilisant un langage de description matérielle qui est le VHDL. La partie dans laquelle je suis le plus investi est la partie Système. Dans cette partie, j'introduis les notions les plus importantes d'un système d'exploitation, à savoir, les tâches, la communication, la synchronisation, la gestion de la mémoire, et la gestion de fichiers. En TD, je procède principalement par des exercices de conception qui ne sont pas nécessairement liés à un système d'exploitation particulier. En TP, les étudiants font principalement du développement POSIX et testent des APIs d'autres normes telles qu'IPC System V. Les thématiques abordées en TP sont le développement multi processus, multi threads, communication et synchronisation, gestion de la mémoire et des fichiers.

3. Systèmes d'Exploitation pour l'Embarqué (16h CM, 32h TP, cours enseigné depuis 2008, cours entièrement élaboré) : C'est une UE que j'ai créé dans le cadre du Master 2 Logiciels pour les Systèmes Embarqués. Le but de cette UE est double : (1) aborder d'une manière plus détaillée les aspects introduits en 3ème année de licence. J'enseigne dans cette UE, d'une part, des aspects avancés quant à la gestion des différents services du système d'exploitation : gestion des processus et structures du système la mettant en œuvre, ainsi que la gestion de la mémoire, des fichiers et des périphériques. D'autre part, j'enseigne aussi des aspects avancés du développement système : développement noyau, développement de pilotes de périphériques, etc. (2) j'aborde plusieurs études de cas de systèmes d'exploitation pour l'embarqué tel que FreeRTOS et VxWorks.

Les TPs sont structurés en deux parties : (1) TPs système et développement POSIX, et (2) une partie projet par petits groupes, dans laquelle les étudiants mènent une étude système de bout en bout, depuis la rédaction du cahier des charges jusqu'à la présentation du projet avec une soutenance. Quelques exemples de projets peuvent être cités: développement d'un traceur d'accès à la mémoire flash au niveau pilotes, conception et implantation d'un *timer* sur FPGA (*Field Programmable Gate Array*) avec développement du pilote pour l'interfaçage avec le système, etc. Parmi les objectifs des projets, il y a le développement noyau, la manipulation d'outils de Linux embarqué, en plus des aspects liés au travail en groupe (répartition des tâches et des rôles) et prise en compte de plusieurs dimensions concernées par le projet (gestion du temps, faisabilité technique, moyens humain, comptes rendus hebdomadaires, etc.).

En plus de ces trois UEs, j'ai été impliqué dans plusieurs autres enseignements de façon occasionnelle (la quasi-totalité, voire la totalité des supports ont été repris) :

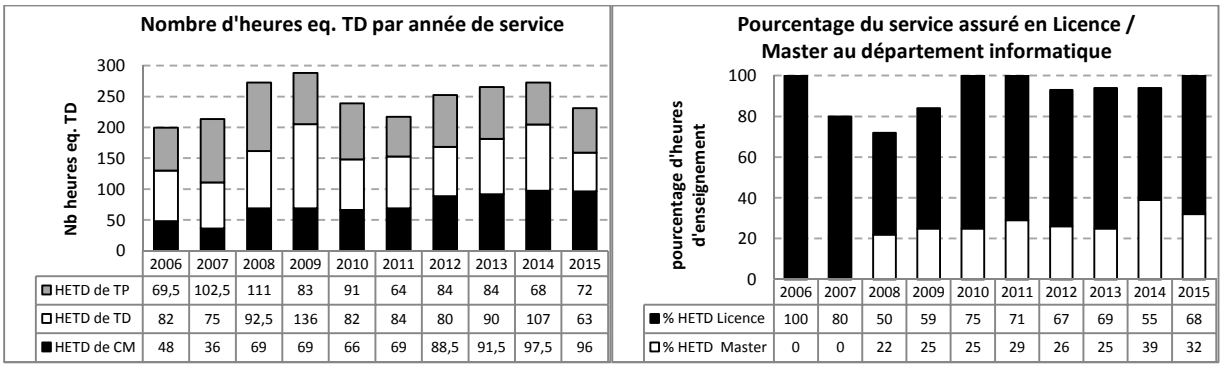
- **Langages de programmation :** il s'agissait ici de transmettre aux étudiants de 2ème année de licence informatique et électronique, au travers de TD et TP, les bases du développement avec comme cas d'étude deux langages de programmation, un langage impératif et qui est le C et un langage objet, il s'agit de SmallTalk.
- **Algorithmique et Langage C :** je suis intervenu en TD et TP dans cette UE dont le but est d'approfondir la maitrise du développement en langage C.
- **Opérateurs de calcul :** cette UE, dont j'ai assuré les CMs, TDs et TPs, avait comme objectif d'aller au-delà des cours d'architecture de processeur classique, en proposant aux étudiants de 2ème année de Licence mathématique-informatique de les former sur des aspects avancés de conception d'unité arithmétique et logique en considérant l'analyse de la performance des différentes conceptions d'opérateurs de base (addition, soustraction, multiplication, et division).
- **Applications de l'informatique :** dans cette UE, il s'agissait de transmettre aux étudiants de 1ère année de Licence les notions de base en réseaux et en internet.
- **C2I :** j'ai aussi participé aux TP du Certificat Informatique et Internet pour différentes UFRs (Droit et Sciences et Techniques) et dans différents départements pendant deux ans.

La part des contrôles continus et des projets : Dans l'ensemble des cours pour lesquels j'ai été responsable d'UE, j'ai donné beaucoup d'importance d'une part, aux contrôles continus, et d'autre part, à l'apprentissage par projet. Concernant les contrôles continus, j'ai instauré une évaluation de l'ensemble des TP par démonstration. Pour chaque TP, les étudiants doivent préparer puis présenter une démo courte à la séance suivante avec un retour immédiat sur leur travail. Ce mode de contrôle continu a permis, de mon point de vue, d'une part d'impliquer davantage les étudiants et par conséquent d'augmenter le taux de présence, et d'une manière générale d'améliorer l'acquisition des connaissances ce qui a permis de sensiblement augmenter la moyenne des notes de la majorité des étudiants, ces derniers étant encouragés à être à jour de leur cours tout au long de l'année. Par ailleurs, je donne beaucoup d'importance à la note du contrôle continu qui représente 50% de la note finale. Concernant les projets, un projet de TP est prévu dans l'ensemble des UEs qui sont sous ma responsabilité. Ces projets sont réalisés en groupe ou individuellement. Pour les étudiants de licence 2ème année, les projets sont réalisés en binôme avec une séance de démonstration et un temps pour les questions et l'échange à la fin du projet. En 3ème année de licence, les étudiants travaillent leur projet d'Architectures et Systèmes 2 individuellement et rédigent un rapport de projet. Enfin, dans mon UE de 2ème année de master, les étudiants mènent un projet en groupe, établissent le cahier des charges, font un compte rendu hebdomadaire de leur activité, et le projet est sanctionné par une soutenance, une démonstration et un rapport.

1.7.1.1 Analyse et bilan quantitatifs

La Figure 1 représente un bilan quantitatif de mes enseignements à l'UBO sur les dix années de service. Le graphique de gauche représente la répartition des heures en CM, TD et TP rapportée en heures équivalent TD. Au niveau des CMs, on peut observer deux inflexions, l'une en 2008, qui correspond à la prise en charge du cours de 2ème année de Master, Systèmes d'Exploitation pour l'Embarqué, et une autre en 2012, correspondant à l'enseignement du cours de systèmes d'exploitation pour les étudiants en licence 3ème année. Concernant les pourcentages de CM, TD et TP, ils se sont stabilisés autour de 35% de CM, 35% de TD et 30% de TP ces dernières années.

Lors de mon recrutement en 2006, l'ensemble de mes cours étaient réalisés en Licence (1ère, 2ème et 3ème année). En 2008, j'ai débuté mes enseignements au niveau master. Depuis, mon service s'est stabilisé à environ 30% d'enseignement en Master et 65% en Licence au département informatique. Le pourcentage restant est relatif à des enseignements réalisés à l'UBO en dehors du département informatique (ex : C2I), ou dans des établissements conventionnés avec l'UBO (ex : ENSTA Bretagne).



	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
HETD de TP	69,5	102,5	111	83	91	64	84	84	68	72
HETD de TD	82	75	92,5	136	82	84	80	90	107	63
HETD de CM	48	36	69	69	66	69	88,5	91,5	97,5	96

	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
% HETD Licence	100	80	50	59	75	71	67	69	55	68
% HETD Master	0	0	22	25	25	29	26	25	39	32

Figure 1. Graphique représentant l'évolution du service d'enseignement en termes d'heures équivalent TD de CM, TD et TP, et en termes de pourcentage d'enseignements réalisés en Licence et en Master (à noter que l'année sur la courbe représente celle du début de l'année scolaire).

1.7.2 Activités d'enseignement hors Université de Bretagne Occidentale

Depuis ma titularisation en tant que maître de conférences à l'UBO, j'ai saisi différentes opportunités de donner des cours dans d'autres établissements. Je me suis essentiellement investi dans deux écoles d'ingénieurs dans lesquelles j'ai entièrement élaboré les cours que je dispense : **l'ENSTA Bretagne** à Brest et **l'ENSEIRB** à Bordeaux. J'enseigne dans ces établissements depuis 5 ans et j'ai conçu des cours (CM, TD, et TP) sur des thématiques liées aux systèmes d'exploitation et l'embarqué pour différents niveaux (bac +4, +5). Par ailleurs, j'ai participé au montage du cours de

système d'exploitation à l'ENSEIRB dans la filière Systèmes Electroniques Embarqués (SEE) et j'ai également contribué au choix des plateformes de développement embarquées à utiliser.

En plus des cours en école d'ingénieur en France, j'ai aussi enseigné à l'étranger :

- D'abord à **l'Université Abou Bekr Belkaid, Tlemcen, Algérie,** ou j'ai été invité à dispenser une semaine de cours que j'ai élaboré sur les systèmes embarqués aux étudiants de 2ème année de master en Réseaux Systèmes Distribués (RSD) en 2012.
- A **l'USTH, University of Science and Technology of Hanoi, Vietnam,** où j'ai dispensé, pendant une semaine, des cours en **langue anglaise** en 2015, sur les systèmes temps réel. J'ai partiellement mis à jour les cours qui avaient été initialement crées par un collègue.
- Et à **l'Ecole Supérieure d'Informatique**, Alger, Algérie, ou j'ai été invité à dispenser quelques heures de cours aux étudiants de première et deuxième année ingénieur sur les hiérarchies mémoire en 2016 et 2017.

L'expérience consistant à confronter sa pédagogie et ses enseignements à différents publics est, de mon point de vue, indispensable pour l'évolution de l'enseignant. Cela lui permet de périodiquement sortir de sa « zone de confort » en enseignant dans des établissements ou pays différents, dans une langue différente, et devant des publics aux parcours et cultures variés, en remettant en permanence en question sa pédagogie et méthode d'enseignement afin de les faire évoluer. À ce moment de mon parcours d'enseignement, cette diversité d'expérience m'apparait avoir été très enrichissante et il me semble essentiel de renforcer cette exigence d'ouverture.

1.7.3 Activités d'enseignement pendant la thèse (Univ. Versailles St Quentin)

Pendant mes années de thèse (2000-2004) et d'ATER (2004-2006), j'ai enseigné dans deux départements à l'Université de Versailles St Quentin en Yvelines (UVSQ) : le département d'informatique et l'Institut des Sciences et Techniques des Yvelines (ISTY), une école de l'UVSQ. J'ai principalement dispensé des cours de microprocesseurs et micro contrôleurs et des cours de systèmes d'exploitation et de langage C à hauteur d'une centaine d'heure par an.

1.8 Formation par la recherche

J'accorde beaucoup d'importance à la formation par la recherche, cela s'est traduit par des encadrements d'étudiants à différents niveaux :

- Travail d'étude et de recherche au niveau de la **1ère année de master** sur une durée d'un mois et demi : **13 étudiants encadrés entre 2009 et 2017**.
- Projet de recherche en laboratoire pour les étudiants en **2ème année de master** : **14 étudiants encadrés entre 2009 et 2017**.
- Stage de fin d'étude de longue durée : **8 étudiants encadrés en stages de longue durée** de niveau 2ème année de master ou dernière année d'école d'ingénieurs
- Thèse de doctorat : **10 étudiants encadrés dont** 4 soutenues.

Les deux derniers types d'encadrement seront détaillés dans la section 2.10.

1.9 Relations avec les entreprises dans le cadre de l'enseignement

En tant que responsable des stages du master 2 Logiciels pour les Systèmes Embarqués (voir la section 1.11) et responsable de l'UE Technologies Emergentes (TEM), j'ai été en permanence en contact avec des industriels. En effet, en tant que responsable des stages, j'allais à la rencontre des industriels dans des salons (par exemple : salon *RTS Embedded Systems* à Paris) afin de tisser des liens. Ces échanges sont nécessaires à plus d'un titre, au-delà de l'intérêt très concret d'avoir accès à des propositions de stage ou, à plus long terme de préparer des débouchés pour nos étudiants, des liens forts avec les industriels permettent aux acteurs académiques que nous sommes, de toujours nous adapter aux réalités du monde industriel, de nous développer dans une dynamique commune, pour former des futurs professionnels qui pourront répondre aux besoins de la société. J'ai été, et je suis, le principal interlocuteur avec les industriels dans le cadre de ce master. L'UE TEM de ce même master est une UE dont les enseignements sont principalement assurés par des intervenants extérieurs tant industriels qu'académiques. L'objectif de cette UE est d'inviter des conférenciers afin de compléter la formation des étudiants avec des points d'actualité non abordés durant

le semestre de cours du master 2. J'ai assuré la responsabilité pédagogique de cette UE, et par conséquent, j'ai été le principal point de contact avec les intervenants extérieurs, industriels ou académiques.

1.10 Collaborations d'enseignement à l'international

Dans le cadre de ma responsabilité de la 2ème année du Master Logiciels pour les Systèmes Embarqués, j'ai été amené à monter des collaborations avec plusieurs universités du Maghreb. L'objectif était, d'une part, de fiabiliser les recrutements d'étudiants étrangers, et d'autre part, de pouvoir leur offrir un meilleur accueil en leur permettant de travailler plus tôt sur leur projet d'études en France tout en leur facilitant des démarches administratives qui peuvent être très lourdes. J'ai ainsi été à l'initiative de l'élaboration de l'ensemble des conventions ci-dessous :

1- **L'Institut National des Postes et Télécommunications (INPT), Rabat, Maroc :** une convention cadre a été signée en 2013 afin de permettre aux étudiants en dernière année de cette école d'ingénieurs marocaine de très bon niveau (reconnue par la CGE française) de se spécialiser en systèmes embarqués à l'UBO. En effet, cette spécialité étant absente à l'INPT, nous avons décidé, dans une démarche partagée, entre collègues de l'INPT et de l'UBO de donner l'opportunité aux étudiants de la suivre dans le cadre de leur cursus à l'INPT. Cette convention est entrée en application en 2013, et nous avons pu accueillir 1 étudiant la première année, en 2013, 4 en 2014, 2 en 2015, et 3 en 2016.
2- **L'Ecole Supérieure d'Informatique (ESI), Alger, Algérie :** une convention du même type que celle de l'INPT a été établie avec l'ESI avec les mêmes objectifs. La convention cadre ainsi que la convention particulière ont été signées et nous avons pu accueillir 1 étudiant la première année (rentrée 2016).
3- **L'Université Abou Bekr Belkaid, Tlemcen, Algérie :** une convention cadre a été élaborée et signée entre l'UBO et l'Université de Tlemcen. Nous avons aussi engagé le montage d'une convention de co-diplomation ente le Master 2 LSE de l'UBO et le Master 2 Réseaux et Systèmes Distribués (RSD). L'objectif est de permettre aux étudiants de Master RSD de faire leur spécialisation à l'UBO en dernière année.

1.11 Responsabilités pédagogiques

Depuis mon recrutement, j'ai eu l'opportunité d'assumer différentes responsabilités pédagogiques résumées dans cette section. Tout d'abord je vais énumérer la liste des responsabilités avant de détailler les tâches assurées pour chacune d'entre elles (par ordre chronologique) :

Responsabilités passées :

- 2007-2008 : Responsable de la 3ème année de licences informatique IUP et CDA pendant un semestre (~60 étudiants)
- 2009-2011 : Responsable d'une salle Système au département informatique
- 2010-2012 : Correspondant auprès de la Coordination Nationale pour la Formation en Microélectronique et en nanotechnologies(CNFM) pour les outils Xilinx.

Responsabilité actuelle :

- Depuis 2009 : Responsable et président de jury de la 2ème année de Master Logiciels pour les Systèmes Embarqués – LSE (~20 étudiants)

1.11.1 Responsabilités passées

1.11.1.1 Responsable de la 3ème année de licences informatique IUP et CDA (2007-2008) :

Cette responsabilité a été assurée pendant un semestre en remplacement du responsable, en mobilité pour un CRCT. L'objectif était d'assurer la gestion de la rentrée des étudiants, une soixantaine, la mise au point de l'emploi du temps, et la gestion du jury. C'était la première responsabilité pédagogique assurée à l'UBO.

1.11.1.2 Responsable d'une salle Système au département informatique (2009-2011) :

Il s'agissait ici de gérer le planning d'une salle système dans le département. En effet, cette salle permettait de pouvoir disposer des droits administrateurs sur les machines. Il était donc important d'en gérer le planning entre les différents

intervenants. Cette responsabilité a disparu depuis le passage du département à la virtualisation, les TP systèmes se font depuis lors sur des machines virtuelles.

1.11.1.3 Correspondant CNFM pour les outils Xilinx (2010-2012) :

J'ai été le point de contact entre le CNFM et l'UBO (département informatique) pour tout ce qui concernait les licences des outils Xilinx utilisés essentiellement en 1ère et 2ème année de master, ainsi que dans plusieurs projets étudiants. J'effectuais donc annuellement les demandes et montage de dossiers nécessaires permettant de disposer des licences en question.

1.11.2 Responsabilités actuelles

Responsable de 2ème année de Master Logiciels pour les Systèmes Embarqués – LSE (depuis 2009) :

Il s'agit de la responsabilité la plus lourde que j'assure au sein du département informatique depuis 2009, mais également la plus stimulante sur le plan pédagogique. Dans ce cadre, j'assure plusieurs activités résumées ci-dessous :

- **Montage des dossiers d'habilitation/évaluation/accréditation** : j'ai élaboré le dossier d'habilitation/accréditation du M2LSE à deux reprises depuis 2010. Aussi j'ai mené le groupe de réflexion concernant l'évolution du master.
- **Gestion des recrutements** : j'organise les recrutements des étudiants locaux, nationaux et étrangers. Pour cela, j'ai initié plusieurs collaborations avec des universités à l'étranger, au travers de conventions, afin de donner une visibilité internationale au master et de fiabiliser le flux d'étudiants. Dans le cadre de ces conventions de collaboration, et dans un esprit d'ouverture internationale, j'organise les recrutements conjointement avec les universités/écoles d'origine des étudiants.
- **Gestion et mise en place des emplois du temps** : Dans le cadre du M2LSE, je suis en charge de la coordination des intervenants et de l'élaboration de l'emploi du temps annuel.
- **Gestion des stages et des relations industrielles**: Je donne également beaucoup d'importance à l'entretien et au développement des relations industrielles avec les partenaires du M2LSE. À ce titre, je m'occupe de récolter les sujets de stage auprès de plusieurs dizaines d'industriels, de la diffusion auprès des étudiants, de la gestion du suivi et des soutenances de stage.
- **Présidence des jurys** : J'organise et préside l'ensemble des jurys d'année du M2LSE
- **Gestion des intervenants extérieurs** : Je m'occupe de la gestion des intervenants extérieurs, surtout dans le cadre de l'UE Technologies Emergentes (TEM) dédiée aux conférenciers extérieurs. Nous accueillons environ, une dizaine de conférenciers par an. Là encore, dans un esprit d'ouverture vers le monde industriel, mais également pour que les dernières avancées scientifiques, et technologiques bénéficient à nos étudiants.
- **Suivi du devenir des étudiants** : Je gère un groupe dédié au master 2 LSE via un réseau social professionnel.
- **Communication et réunions d'orientation** : Je me donne pour mission de valoriser le master, de le faire connaitre à travers plusieurs exposés par an. Ces exposés sont réalisés en interne et à l'extérieur, et ce au niveau du département informatique de l'UBO pour les étudiants de L3 et de M1, au niveau de l'UFR, et également pour des étudiants étrangers des universités avec lesquelles des conventions ont été signés.

2 RECHERCHE

2.1 Parcours de recherche

Maître de conférences depuis 2006, j'ai intégré l'équipe Architecture et Système (A&S) du laboratoire LESTER (Laboratoire d'Electronique des Systèmes TEmps Réel), laboratoire dont les tutelles étaient l'Université de Bretagne Sud et de l'Université de Bretagne Occidentale, et qui était associé au CNRS depuis le 1er janvier 2004 (FRE 2734). L'équipe localisée à Brest était reconnue pour son expertise dans le domaine du développement d'outils de synthèse pour les systèmes embarqués reconfigurables en s'appuyant sur des techniques d'ingénierie du logiciel. J'ai commencé mon travail de recherche, de début 2007 à fin 2008, en intégrant le projet européen FP6 MORPHEUS en participant aux travaux de l'équipe concernant le développement d'outils de synthèse de haut niveau permettant de porter des applications développées dans plusieurs langages différents, sur des architectures reconfigurables à différents grains dans un système sur puce complexe (System on Chip – SoC).

En 2008, le LESTER a fusionné avec d'autres laboratoires constituant le Lab-STICC (Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance), actuellement UMR 6285, dont je suis membre intégré au pôle Communications, Architectures, Circuits et Systèmes (CACS), et plus précisément dans l'équipe Méthodes et Outils, Circuits et Systèmes (MOCS). À la suite du projet européen MORPHEUS, j'ai travaillé sur la thématique de la modélisation et de l'optimisation de la consommation énergétique dans les systèmes embarqués. Au sein de cette thématique, je me suis particulièrement intéressé à deux études de cas : (1) l'étude de la consommation énergétique dans les mobiles pour les applications de streaming vidéo, en collaboration avec une université algérienne (voir la section 2.11), et (2) la modélisation et la simulation de la performance et de la consommation énergétique des systèmes de stockage embarqués à base de mémoire flash, dans le cadre du projet ANR Open-PEOPLE (plateforme ouverte pour l'estimation et l'optimisation de la consommation en puissance et en énergie) que j'ai rejoint à partir de sa deuxième année. Le point commun de ces deux études menées parallèlement est le positionnement au niveau système d'exploitation embarqué et l'approche. Durant cette même période, j'ai aussi mené d'autres études dans le domaine du stockage sur mémoire flash initiant ainsi une nouvelle thématique au sein du Lab-STICC, thématique devenue aujourd'hui mon sujet de recherche principal et que j'anime au niveau du laboratoire.

À partir de 2011 et jusqu'à ce jour, j'ai élargi le domaine d'application de mes recherches sur les systèmes de stockage, et plus généralement l'étude de la hiérarchie mémoire, vers les domaines du *Cloud computing* (projet Infrastructure de Nuage informatique Distribuée et Econome en Energie de l'IRT b<>com), des bases de données hautes performances (projet Flash-BaD avec l'entreprise KoDe Software) et embarquées, des systèmes temps réel et de l'internet des objets[2] (dans le cadre d'un projet avec Orange)

Actuellement, je suis mis à disposition, à 20% de mon temps de travail pour le compte de l'IRT b<>com dans lequel je travaille sur la thématique de l'optimisation des systèmes de stockage hybride (disques dur et mémoire flash) dans un Cloud.

Dans la partie qui va suivre, je vais résumer les travaux de recherche menés depuis que je suis maître de conférences à l'UBO selon les trois axes suivants :

- Axe 1 (2007 – fin 2008): introduction à la synthèse haut niveau pour architectures reconfigurables et outillage
- Axe 2 (2009 – à ce jour): Systèmes d'exploitation embarqués, performance et consommation énergétique
- Axe 3 (2012 – à ce jour): Optimisation des systèmes de stockage

Il faut noter que la répartition selon les trois axes est principalement chronologique et est donc forcément déséquilibrée du point de vue de l'investissement et de la production scientifique. L'axe 1 est un axe abordé durant les premières années à l'UBO et n'est pas représentatif de mes activités actuelles. À ce titre, il ne sera pas détaillé dans ce document (seulement résumé dans cette partie). En effet, très tôt, je me suis investi dans un nouveau thème de recherche lié au domaine du stockage (dès l'axe 2, puis dans l'axe 3) qui est devenu le thème central de mes recherches**. Ce document se focalise sur les axes 2 et 3.**

[2] Je m'intéresse particulièrement aux « objets » (au sens systèmes embarqués) et leur système de stockage.

La Table 1 résume les activités de recherches menées depuis 2007 sur les différents axes. Pour chaque axe, sont listés le domaine de recherche sur lequel j'ai travaillé, le domaine applicatif lié au domaine de recherche et enfin les outils scientifiques utilisés pour mener ces recherches.

Table 1. Diagramme temporel représentant le contexte applicatif de recherche, les thématiques de recherche et les méthodes et outils de recherche utilisés depuis mon intégration à l'UBO.

2.2 Axe 1 : Introduction à la synthèse haut niveau pour architectures reconfigurables et outillage

2.2.1 Contexte

Les architectures reconfigurables (telles que les FPGA) et les réseaux de communication NoC (Network-on-Chip) ont fait émerger des domaines de recherches qui ont été largement investigués [1]. Les plateformes actuelles de systèmes sur puce, ou SoC, peuvent être très complexes et contenir différents types d'unités de calcul sur lesquelles peut s'exécuter une application donnée : unités de calcul généraliste, de traitement de signal, unités reconfigurables, etc. En effet, une application donnée peut avoir une implantation logicielle ou matérielle et l'une ou l'autre des implantations peut être exécutée selon certaines contraintes qui peuvent être liées aux performances ou à la consommation énergétique par exemple. Il est question de co-design matériel/logiciel adaptatif sur plateforme hétérogène.

Idéalement, sur un SoC contenant plusieurs unités de calcul de types différents, on pourrait exécuter une application sur le cœur de calcul le plus adéquat selon les contraintes et l'état du système. Mais cela reste utopique, principalement à cause de l'hétérogénéité des outils de manière générale [1]. En effet, chaque processeur/technologie et IP (*Intellectual Property*) est manipulable par un ensemble d'outils différents et qui ne sont pas toujours inter opérables.

L'objectif du projet européen intégré MORPHEUS (IST 027342), était de proposer, d'une part, un SoC, hétérogène et contenant plusieurs unités de calcul dont certaines reconfigurables et de granularités différentes. D'autre part, MORPHEUS devait aussi proposer un ensemble d'outils intégrés et homogènes pour le développement spatial et temporel sur le SoC proposé.

2.2.2 Problématique

L'exploitation d'architectures hétérogènes reconfigurables sur puce requiert de nouveaux outils logiciels. La problématique majeure que nous avons considéré dans le cadre de cet axe est l'homogénéisation et la simplification, du point de vue de l'utilisateur, de l'exploitation de ce type d'architecture dans le but de réduire les temps de développement et donc de mise sur le marché. En effet, une chaîne de programmation doit permettre, en appui sur un modèle de calcul homogène, de migrer d'une spécification de haut niveau vers une configuration des ressources matérielles, couplée à un contrôle des transferts de données [2].

Dans le cadre du projet MORPHEUS, l'objectif de l'équipe A&S du LESTER UBO était de proposer une solution sous la forme (1) d'un modèle de calcul, (2) d'une structure de données portable favorisant l'intégration d'outils tiers afin de constituer une chaîne logicielle complète, et (3) d'une couche basse générique sachant traiter la synthèse dans un cadre reconfigurable. C'est sur le 2ème point que j'ai travaillé en collaboration avec deux membres de l'équipe durant mes premières années à l'UBO.

2.2.3 Contribution

L'objectif de notre contribution était donc de proposer un ensemble d'outils permettant de programmer un SoC hétérogène avec un flot de synthèse de haut niveau. Le développement d'une chaîne de programmation pour des architectures reconfigurables et hétérogènes requiert la mise en œuvre d'outils communiquant à deux niveaux :

1. un niveau abstrait, ou couche haute, permettant, indépendamment de toute cible matérielle, de modéliser une représentation algorithmique des traitements avec le flot de contrôle, le flot de données ainsi que la structure du programme source [3] ;
2. un niveau d'implantation ou couche basse, qui assure la configuration d'une cible matérielle et le transfert des données à partir d'une représentation algorithmique des traitements.

La couche haute se constitue d'une chaîne logicielle, chaque outil de la chaîne intervient spécifiquement sur la définition de la représentation algorithmique des traitements. Chaque outil de la couche basse a pour rôle la traduction d'une représentation algorithmique des traitements vers une cible matérielle particulière (voir la Figure 2).

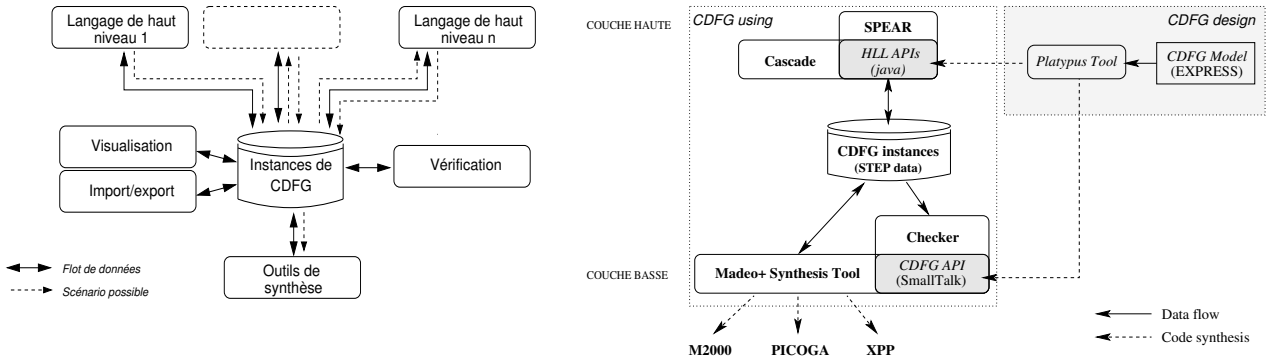


Figure 2. CDFG et outillage

Le format CDFG (*Control and Data Flow Graph*) hiérarchique [3] a été choisi comme représentation intermédiaire permettant une capture du flot de contrôle, du flot de données ainsi que de la structure du programme source permettant d'appliquer plus aisément des optimisations sur le code source. J'ai donc travaillé sur la définition du modèle de CDFG ainsi que le développement d'une partie des outils associés [4][5][6].

La Figure 2 montre la position centrale qu'occupe l'outillage autour de la représentation intermédiaire qu'est le CDFG. Un scénario possible est la mise en œuvre de plusieurs applications codées dans différents langages (en l'occurrence, C++, java ou Smalltalk dans le cadre de ce projet). Grâce aux APIs que nous avons développées, il est possible d'obtenir une représentation homogène de ces applications sous le format du CDFG. Ce CDFG, peut être validé et est utilisé par les outils de synthèse de bas niveau.

La mise en œuvre d'outils interopérables par échange de données nécessite la définition d'un format d'échange commun et d'un protocole d'échange standard. Ceux-là sont mis en œuvre par un mécanisme d'encodage et de décodage normalisé des données échangées. L'encodage produit une représentation échangeable des données à partir d'objets manipulés dans un dépôt de données. Le décodage effectue l'opération inverse. La portabilité d'un système d'échange est garantie par l'utilisation d'un format neutre et par la possibilité de disposer d'un encodeur et d'un décodeur quel que soit le système. L'évolution des systèmes est facilitée si le dispositif d'échange prévoit la production automatique des encodeurs et décodeurs à partir d'une spécification formelle pivot de la structure et de la sémantique des données échangées.

La technologie STEP [7] a été exploitée pour la spécification du CDFG et pour la mise en œuvre des échanges de données. La robustesse du système repose sur une définition formelle précise des données échangées avec le langage

de modélisation de données EXPRESS [8], sur l'utilisation du format d'échange neutre de STEP et sur la production automatique des encodeurs et décodeurs pour tous les outils s'articulant autour de la définition du CDFG. Pour la modélisation EXPRESS et la production automatique des interfaces logicielles, nous avons utilisé l'environnement Platypus [9] développé par le Pr. Alain Plantec.

Dans le cadre du projet MORPHEUS, les applications ont été générées à partir de deux outils distincts : (1) l'outil Cascade de Critical Blue permettant de générer les processus de calcul et (2) l'outil SPEAR [10] de Thalès permettant de générer les processus de communication. Ces deux outils utilisent des langages de programmation différents, Java et C++. Grâce à Platypus, nous avons généré des API dans ces différents langages afin de retomber sur une représentation homogène du CDFG écrites en STEP. Le CDFG est utilisé par l'outil de synthèse (utilisant le langage de programmation Smalltalk), toujours grâce aux APIs (voir Figure 2).

2.2.4 Production scientifique

Mon travail sur cet axe s'est étalé sur une durée d'un an et demi environ. Durant cette période, je me suis familiarisé avec le nouveau domaine, et avec l'approche et outils de l'équipe. Un travail conséquent de développement a été réalisé durant cette période. En effet, j'étais la principale interface entre les industriels Critical Blue et Thalès d'une part, et les développeurs de l'outil de synthèse et de Platypus d'autre part. Cette période de familiarisation ainsi que le travail de développement ont constitué une bonne introduction au domaine du reconfigurable et de l'embarqué en général sur lesquels je me suis appuyé pour le développement de l'axe 2.

Production scientifique :

Revues	Conférences internationales	Conférences nationales et ateliers
	[ci24]	[cn6] [w12]

2.3 Axe 2 : Systèmes d'exploitation embarqués, performance et consommation énergétique

2.3.1 Contexte

Les recherches réalisées sur cet axe ont débuté en 2009, année pendant laquelle les ventes de smartphone ont explosé. C'est l'un des facteurs ayant donné une impulsion extraordinaire au développement de systèmes embarqués de types variés. Ceci est principalement dû à la baisse des coûts de production. À titre d'exemple, le coût de la mémoire flash qui s'élevait à environ 10$/GO en 2007 a baissé d'un facteur supérieur à 5 pour atteindre 1.8$/GO en 2010 et passer en dessous de la barre symbolique du 1$/GO fin 2011[11][12].

L'explosion des ventes de smartphones et des systèmes embarqués a permis une multiplication et une complexification des usages de ce type de systèmes, ce qui s'est traduit par un besoin accru en performance. En effet, des applications telles que le visionnage de vidéos, le partage de photos, et les réseaux sociaux, ont poussé à une complexification du matériel utilisé : CPU, GPU, DSP, voire FPGA, et à une explosion du volume des données multimédia générées impliquant l'usage d'un système de stockage performant.

Cette complexification du matériel utilisé dans les systèmes embarqués, couplée à une diversification de l'applicatif a augmenté de plus en plus de pression sur les systèmes d'exploitation pour l'embarqué afin de gérer d'une manière efficace les ressources matérielles à disposition dans le but de garantir une qualité de service optimale pour l'utilisateur.

Un autre problème est celui de la consommation énergétique. En effet, *l'International Technology Road-map for Semi-conducteurs* (ITRS) a prédit une augmentation de la consommation énergétique d'un facteur 2.5 pour la décennie à venir [11]. Malheureusement, l'augmentation des performances des batteries alimentant ces systèmes embarqués ne suit pas cette tendance [13].

2.3.2 Problématique

Nous avons donc d'une part une complexification et une augmentation des performances du matériel utilisé dans les systèmes embarqués grand public dues à un usage applicatif de plus en plus varié, et d'autre part, des contraintes énergétiques de plus en plus difficiles à tenir au vu de l'évolution des batteries.

Dans ce contexte, l'optimisation des performances et de la consommation énergétique des systèmes embarqués passe nécessairement par une compréhension et une analyse fine des comportements : des applications, du système, et du matériel utilisés, mais aussi des interactions entre ces différents niveaux.

Nous nous sommes intéressés à trois problématiques principales dans le cadre de cet axe :

1. La compréhension et l'analyse de la consommation énergétique dans un environnement de systèmes embarqués complexes ;
2. L'intégration efficace de systèmes de stockage à base de mémoire flash dans les systèmes informatiques pour l'optimisation des performances ;
3. L'exploitation de matériel reconfigurable pour l'optimisation des services du système d'exploitation (OS pour *Operating System* dans le reste du document). Il est à noter que cette dernière problématique découle des problématiques traitées dans l'axe 1.

2.3.3 Contributions

Dans cette partie, chaque sous-section traite de l'une des trois problématiques abordées dans le cadre de cet axe.

2.3.3.1 Performance et consommation énergétique pour les systèmes embarqués complexes

L'objectif de ces travaux est principalement l'analyse et la compréhension de la consommation énergétique des systèmes embarqués. Nous avons choisi deux cas d'études différents. Un premier cas d'étude est centré sur l'application. En effet, nous avons considéré l'une des applications les plus utilisées et des plus gourmande en calcul et donc en énergie dans le cadre de la téléphonie mobile : **le streaming vidéo** [14][15]. Nous avons aussi considéré, dans une autre étude, une approche, cette fois-ci, centrée sur un service spécifique de l'OS, et qui est **le système de gestion du stockage**.

Ces deux études ont fait l'objet de deux thèses, celle de M. Yahia Benmoussa et celle de M. Pierre Olivier (voir section 2.10). La première thèse a été encadrée en co-tutelle avec l'Université Mhamed Bougara de Boumerdès, Algérie. La seconde est une thèse MESR. Les deux thèses ont bénéficié du support du projet ANR Open-PEOPLE (voir section 2.10.1.1) et ont été dirigées par Eric Senn (Maître de conférences, HDR, Lab-STICC, UBS, Lorient).

Les deux études ont été réalisées avec une approche méthodologique similaire (voir Figure 3). Cette méthodologie est basée sur trois étapes : (1) une étape expérimentale pour l'étude de la performance et de la consommation énergétique, (2) une étape de modélisation, (3) une étape d'application des modèles élaborés à des fins d'optimisation de certaines métriques de performance, ou à défaut de production d'outils d'évaluation de performance.

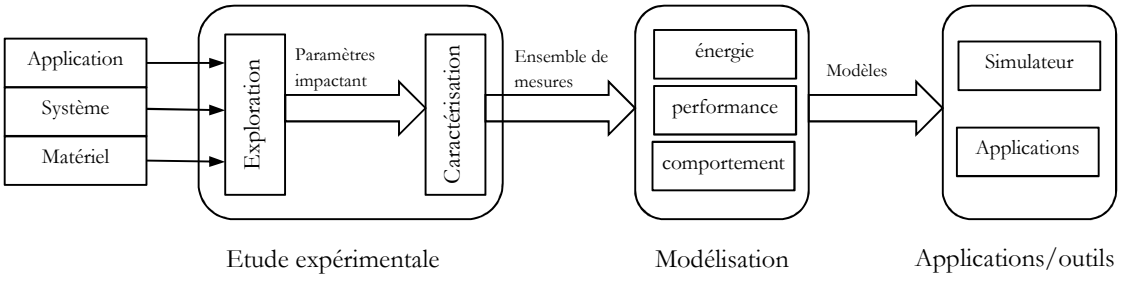


Figure 3. Méthodologie globale pour l'étude sur l'analyse de la consommation énergétique

Etude de la consommation énergétique du streaming vidéo

Dans le cadre de l'application du streaming vidéo, l'étape d'exploration des performances est basée sur une méthodologie expérimentale permettant d'évaluer de façon exhaustive et à différents niveaux d'abstraction, les performances et la consommation énergétique du décodage vidéo [16]. Cette étude expérimentale a été réalisée sur des plateformes embarquées sur lesquelles a été évalué un large éventail de configurations de l'application de décodage vidéo en termes de résolution, complexité de la vidéo, *bit-rate*, etc. Pour ce qui est des paramètres système, la consommation énergétique de l'exécution du système avec différentes fréquences a été mesurée. Pour ce qui concerne le matériel, d'une part, ont été évaluées plusieurs plateformes embarquées, et d'autre part, pour chacune d'entre elles,

plusieurs types de processeurs ont été testés, parmi lesquels un processeur généraliste (ARM) et un processeur de traitement de signal DSP. Cette étape a souligné l'importance de l'analyse de différents paramètres qui peuvent se rapporter à différents niveaux d'abstraction dans l'évaluation de l'efficacité énergétique globale d'un système donné.

Les mesures obtenues dans la première étape ont été utilisées pour construire des modèles de performance et de consommation d'énergie pour le décodage vidéo. Ces modèles permettent d'estimer avec une grande précision (de l'ordre de R^2= 97 %) la performance et la consommation d'énergie du décodage vidéo en fonction d'un nombre de paramètres comprenant la qualité de la vidéo et la fréquence et type du processeur [17]. De plus, en se fondant sur une caractérisation multi-niveaux et une approche de modélisation par décomposition en sous-modèles, nous avons montré comment les modèles développés, contrairement aux modèles empiriques classiques, peuvent être généralisables à d'autres plateformes matérielles [18].

Nous avons proposé, dans le cadre de cette étude, certaines applications possibles utilisant les modèles développés [19], par exemple pour un décodage vidéo adaptatif. Le principe consiste à exploiter la capacité du modèle de performance proposé pour prédire le temps de décodage d'une qualité vidéo donnée. Ceci est réalisé afin de mieux dimensionner les ressources de calcul dans un but de réduction de consommation énergétique. De plus, nous avons proposé une méthodologie permettant d'étendre les modèles à d'autres plateformes.

Etude de la consommation énergétique du système de gestion du stockage

Dans cette étude, en phase d'exploration, nous avons identifié les différents éléments d'un système de stockage qui impactent les performances et la consommation. Il s'agit de paramètres matériels : la mémoire flash, le CPU et la RAM ; de système, correspondant aux différents niveaux de la pile logicielle de gestion du stockage flash sous Linux [20][21][22] : le système de fichiers virtuel (Virtual File System, VFS), le système de fichiers, et le pilote NAND. Cette exploration est réalisée en fonction du comportement de la charge d'E/S de l'application en termes de type d'opération, taille des requêtes, mode d'accès, etc.

La phase de modélisation correspond à la représentation par des modèles de l'impact des différents éléments identifiés en phase précédente [23]. Des modèles de différents types sont proposés : (1) les modèles fonctionnels représentent les algorithmes de gestion du stockage (englobant les services de l'OS) ; (2) les modèles de performances et de consommation caractérisent le profil d'évolution des différentes métriques (consommation énergétique, temps de réponse, débit, etc.) pour une plateforme donnée ; (3) les modèles de charge d'E/S sont utilisés pour représenter la charge appliquée sur le système de stockage ; (4) enfin, des modèles spécifiques aux composants matériels flash sont présentés : il s'agit de modèles structurels, représentant son architecture, et opérationnels, relatifs aux différentes opérations supportées par la mémoire flash. A chaque modèle de performances et de consommation est associé une méthodologie d'extraction de paramètres permettant de construire le profil d'une plateforme matérielle et logicielle donnée [24][25]. Nous avons pris comme cas d'étude pour le modèle fonctionnel le système de fichiers JFFS2 [26].

Les modèles sont implantés dans un simulateur [27]. Cet outil permet, d'une part, d'obtenir des estimations concernant les performances et la consommation des systèmes de stockage à base de mémoire flash, et d'autre part, de fournir une infrastructure de développement pour y intégrer de nouvelles couches de gestion. Il peut ainsi être utilisé pour le prototypage de couches de gestion, la validation d'optimisations et les études comparatives concernant les performances et la consommation énergétique. Le simulateur est également utilisé pour valider les modèles précédemment présentés. La différence entre l'estimation et la mesure reste, dans la majeure partie des cas, sous la barre des 10 %.

2.3.3.2 Intégration des systèmes de stockage à base de mémoire flash

Les mémoires flash NAND sont des mémoires non-volatiles de type EEPROM *(Electrically Erasable and Programmable Read Only Memory*) Elles ont une structure hiérarchique. Elles sont composées d'une ou de plusieurs puces. Chaque puce est divisée en plusieurs plans. Un plan est composé d'un nombre fixe de blocs, chacun d'entre eux contenant un nombre donné de pages. Trois opérations sont supportées : lecture, écriture/programmation, et effacement.

Les contraintes de la mémoire flash peuvent être résumées comme suit : (1) l'effacement avant écriture : une donnée ne peut être écrite sur une page qu'à la suite d'un effacement, (2) l'asymétrie des opérations d'écriture et d'effacement. En effet, une écriture doit se faire sur une page, alors que l'effacement s'applique à un bloc entier, (3) le nombre limité de

cycles d'effacement que l'on peut réaliser sur une cellule mémoire donnée, et enfin (4) l'asymétrie des performances entre lecture, écriture et effacement [29][28].

Afin de palier à ces problèmes, plusieurs mécanismes ont été conçus [30][31]: (1) un mécanisme de traduction d'adresse permettant de sauvegarder l'emplacement physique d'une donnée En effet, du fait de la première et de la seconde contrainte, il n'est pas envisageable de modifier une donnée dans son emplacement original, (2) un mécanisme de ramasse miettes permettant de recycler les pages invalides dues au déplacement de données, et (3) un mécanisme permettant de répartir l'usure en équilibrant le nombre d'effacements subis par bloc. Ces mécanismes sont généralement implantés dans les contrôleurs de mémoire flash (SSD ou *Solid State Drive*, clé USB, carte SD, etc.) via une couche appelée la FTL (pour *Flash Translation Layer*). De plus, des mécanismes de cache prenant en compte les spécificités des mémoires flash ont été proposés [32][33][34]. Ils permettent essentiellement deux optimisations : (1) absorber les opérations d'écritures et donc éviter de les reporter sur la mémoire flash, (2) tamponner les données afin de permettre une réorganisation postérieure à des fins d'optimisation.

Dans le cadre de l'intégration et de l'optimisation des mémoires flash, nous avons proposé trois mécanismes dont un de cache, nommé C-lash (*Cache for flash*) [35][36], et deux mécanismes de traduction d'adresse nommés CACH-FTL (*Cache Aware Configurable Hybrid Flash Translation Layer*) [37][38] et MaCACH (*Maximum page-mapped region usage, Cache-Aware, and Configurable Hybrid FTL*) [39].

L'idée défendue dans C-lash est que, pour un certain nombre de types de charge d'E/S, il est possible de simplifier la conception d'un système de stockage à base de mémoire flash en omettant les trois services cités précédemment, à savoir la correspondance d'adresse, le ramasse-miettes et la répartition de l'usure (ou *wear levelling*), et de s'appuyer principalement sur un mécanisme de cache performant, en l'occurrence C-lash. C-lash prend en considération les caractéristiques des mémoires flash, les coûts des différentes opérations (lecture, écriture et effacement), en termes de temps d'accès et de durée de vie, en évinçant sur la mémoire flash seulement les blocs contenant un maximum de pages valides. Cela réduit grandement le nombre d'opérations d'effacement, et améliore les performances. C-lash prend également en compte les localités temporelles et spatiales des données. Les performances de C-lash ont été testées sur un large ensemble de charges d'E/S générées synthétiquement ou extraites des dépôts de benchmarks reconnus par la communauté (http://iotta.snia.org/). Les expériences menées ont prouvé la pertinence et l'efficacité de C-lash pour toutes les traces présentant un taux de séquentialité moyen ou élevé.

La traduction d'adresse en mémoire flash peut se faire selon deux granularités : la page ou le bloc. L'inconvénient de la traduction d'adresse par page est la taille de la table de traduction qui devient trop importante pour être stockée en RAM [28], mais ce type de traduction donne de très bonnes performances. La traduction par bloc a comme inconvénient majeur son manque de flexibilité qui aboutit à de mauvaises performances. À titre d'exemple, si les données de la $n^{ième}$ page du $m^{ième}$ bloc doivent être mises à jour, elles doivent l'être à la $n^{ième}$ position de ce même bloc ou d'un autre bloc. Dans le premier cas, cela présuppose la lecture de l'ensemble des données valides, l'effacement du bloc puis la réécriture de ces données, et dans le second cas, le déplacement de l'ensemble des données valides, y compris les données mises à jour, vers un autre bloc. Plusieurs études de l'état-de-l'art proposent des FTL à traduction d'adresse hybride dans lesquels on utilise les deux granularités pour établir la correspondance d'adresse [40][41][42]. Nous avons proposé dans un premier temps CACH-FTL, une FTL hybride qui utilise les données du cache du SSD afin de décider du placement de la donnée en zone à correspondance par page ou par bloc. CACH-FTL est configurable dans le sens où elle permet un large éventail de configurations. Cette configurabilité permet, hors-ligne, de régler le système d'une manière fine selon la charge d'E/S pour en optimiser les performances.

MaCACH est une FTL adaptative, une évolution de CACH-FTL dans laquelle le réglage des paramètres de la FTL se fait en-ligne en se basant sur un mécanisme d'asservissement de type PID (*Proportional Integral Derivative*). MaCACH s'adapte donc en fonction de la charge d'E/S et de l'état de la mémoire flash afin de maximiser les performances en utilisant d'une manière efficace l'espace à correspondance par page. Les tests réalisés sur MaCACH ont montré de très bonnes performances en termes de temps de réponse moyen et de nombre d'effacement comparativement aux FTL de l'état-de-l'art [40] [43][44], et surtout une plus grande flexibilité de par l'espace de configuration qu'offre cette FTL.

2.3.3.3 Optimisation des services de l'OS et architectures reconfigurables

Dans la continuité de l'axe 1 et dans le cadre du développement de l'usage et de l'outillage pour l'intégration des architectures reconfigurables dans les systèmes embarqués complexes, nous avons étudié la possibilité de coupler d'une part architecture logicielle à composants [47][48] et d'autre part architecture matérielle reconfigurable. L'architecture logicielle à composants permet la conception de logiciels comme un assemblage de composants interconnectés ce qui permet d'envisager, une fois le logiciel déployé, de faire évoluer sa structure. L'architecture matérielle reconfigurable permet, quant à elle, d'accélérer certains traitements. L'émergence de la reconfigurabilité partielle et dynamique [49][50] couplée à la puissance des circuit reconfigurables actuels nous permet d'envisager des mises en œuvres de systèmes d'exploitation sur circuit reconfigurable de type FPGA.

L'objectif de l'étude était donc d'utiliser les avantages des architectures logicielles à composant permettant d'isoler les services de l'OS et de les interfacer d'une manière standardisée avec les circuits reconfigurables afin de permettre une gestion flexible et évolutive des services de l'OS. Il s'agissait de poser les bases d'une telle approche, et le service choisi comme cas d'étude est celui de gestion de temps dans un OS temps réel (dans notre cas μCOSII).

Le service de gestion de temps de l'OS est extrêmement important et se base sur un *timer* matériel qui génère des interruptions périodiques notifiant ainsi l'OS de l'écoulement du temps. Ce dernier doit donc gérer ces interruptions et notifier les différents services et applications afin d'entreprendre les actions associées. Deux problèmes peuvent se poser quant à l'usage d'une telle technique : (1) la précision temporelle est limitée par le temps nécessaire à gérer les interruptions, et (2) ce service est peu flexible car il n'est pas modifiable en cours d'exécution mais seulement à la configuration du système. Afin de réduire le temps de gestion de l'interruption, et de le rendre indépendant de la charge du système et donc plus prédictible, nous avons conçu une version matérielle configurable du service de gestion de temps qui communique avec le système grâce à des interfaces standards [46]. Le *timer* mis en œuvre supporte la configuration matérielle partielle, permettant ainsi de changer la configuration (exemple : résolution) du *timer* en cours d'exécution. Ceci permet d'une part une meilleure précision : en effet, augmenter la fréquence ne cause plus une augmentation importante de la latence du système, et d'autre part le service de gestion de temps devient plus adaptable car reconfigurable en fonction de l'applicatif.

2.3.4 Production scientifique

Mon travail sur cet axe a débuté fin 2009 avec un fort développement de l'activité centrée sur le stockage ce qui a engendré un recul des autres thématiques (architecture reconfigurable). Les travaux de cet axe ont abouti à l'encadrement de deux thèses de doctorat soutenues, celle de M. Pierre Olivier et de M. Yahia Benmoussa.

Production scientifique : 14 articles dans des revues internationales, nationales ou chapitres de livre, 11 articles en conférence internationale, 4 articles en conférence nationale, 4 articles de workshop.

Revues et Chapitres	Conférences internationales	Conférences nationales et ateliers
[ri3][ri5][ri6][ri7][ri8][ri9][ri10][ri11][ri13][ri14][ri15][rn1][rn2][ch2]	[ci10][ci14][ci15][ci16][ci17][ci18][ci19][ci20][ci21][ci22][ci23]	[cn2][cn3][cn4][cn5][w8][w9] [w10][w11]

2.4 Axe 3 : Optimisation des systèmes de stockage

A partir de 2012, j'ai centré l'ensemble de mes activités sur les hiérarchies mémoire et plus précisément sur les systèmes de stockage et mémoires non-volatiles.

2.4.1 Contexte

De nos jours, la quantité des données numériques, sous toutes ses formes, suit une croissance exponentielle. En effet, ce volume croit plus vite que la loi de Moore (même si cette dernière n'est pas censée représenter cette métrique). À titre d'exemple, les données en ligne indexées par Google ont augmenté d'un facteur supérieur à 50 entre 2002 et 2009 (de 5 à 280 exaoctets)[51]. Cette tendance est loin d'être spécifique aux données du web, le volume de données des entreprises subit, lui aussi, cette croissance rapide avec une augmentation cumulative de 173% sur la même période [51][52].

Dans une étude récente [53], EMC (un acteur majeur du domaine du stockage) prédit que le volume de données numériques créées annuellement augmentera d'un facteur 44 de 2009 à l'horizon 2020. EMC prédit aussi que plus du tiers de cet ensemble de données sera stocké ou transitera via le Cloud. Cela démontre le besoin très important en capacité de stockage additionnelle afin de pouvoir absorber ce déluge de données.

En réalité, les applications deviennent de plus en plus intensives en traitement de données. À titre d'exemple, on peut citer les applications de partage de vidéos et d'images via les réseaux sociaux, les traitements transactionnels, les moteurs de recherche, le traitement des courriers électroniques, la télévision numérique, etc.

Plus généralement, les données numériques sont hétérogènes par nature et sont traitées et gérées en utilisant des techniques et méthodes différentes [51] : capture, analyse, traitement, classification, archivage, etc. Cette hétérogénéité des traitements ainsi que la volumétrie importante sont des facteurs qui mettent de plus en plus de pression sur les sous-systèmes de stockage et sur toute la hiérarchie mémoire poussant la communauté scientifique à réfléchir à de nouvelles technologies et méthodes permettant d'optimiser les performances de ces systèmes.

2.4.2 Problématiques

Dans ce contexte orienté données, il est de plus en plus central d'optimiser les performances des E/S du système de stockage. En effet, le sous-système de stockage représente le goulet d'étranglement historique du fait de l'écart de performance en croissance permanente par rapports aux unités de calcul.

Les systèmes de stockage à base de mémoire flash sont venus réduire l'écart de performance existant entre les unités de calcul et le système de stockage [45]. La mémoire flash a été intégrée aux systèmes informatiques actuels de trois manières différentes [45][28] : (1) comme extension de la mémoire principale, (2) comme accélérateur de système de stockage. Dans ce cas la mémoire flash est considérée comme un cache (par rapport au disque) qui stocke les données fréquemment accédées dans le but de réduire les accès aux disques, (3) comme un périphérique de stockage alternatif ou complémentaire au stockage traditionnel sur disque dur, généralement sous forme de SSD.

Les mémoires flash présentent des avantages certains en termes de performance et de consommation énergétique [54]. En revanche, leur coût et la forte demande en stockage nous laisse penser que les disques durs ne vont pas disparaitre à moyen terme. Il est alors important de considérer les systèmes de stockage actuels comme des systèmes hybrides (ou hétérogènes) contenant les deux types de stockage : disques durs magnétiques et mémoire flash.

C'est dans ce contexte que mes travaux dans le cadre de l'axe 3 s'inscrivent. La problématique majeure abordée consiste à étudier comment exploiter au mieux ces nouveaux systèmes de stockage hybrides et comment faire en sorte que différents types d'applications en tirent profit.

Je me suis principalement intéressé à deux applications dans le cadre de cet axe :

1. L'intégration des mémoires flash dans les infrastructures de Cloud
2. L'intégration des mémoires flash pour les Bases de Données.

2.4.3 Contributions

Dans cette partie, chaque sous-section traite de l'une des problématiques abordées dans cet axe.

2.4.3.1 Intégration des mémoires flash dans les infrastructures de Cloud

J'ai entamé le travail concernant l'intégration des mémoires flash dans les infrastructures de Cloud sur trois applications différentes. Le travail sur chaque application fait l'objet d'un projet et/ou contrat industriel différent.

Cloud IaaS

Le travail concernant l'intégration des mémoires flash dans un Cloud de type IaaS (*Infrastructure as a Service*) a été entamé avec l'IRT b<>com dans le cadre de son projet Infrastructure de Nuage informatique Distribuée et Econome en Energie (INDEED), voir section 2.8. L'objectif de nos recherches dans le cadre de ce projet est de proposer une architecture de stockage hydride avec comme fin l'optimisation des performances et la réduction du coût global du stockage. Dans le cadre d'un Cloud IaaS, le client se voit allouer, selon les termes du contrat passé avec le fournisseur

de Cloud, un environnement virtualisé. Cet environnement est constitué de ressources CPU, mémoire, stockage et réseau. Le client y exécute ses applications au travers de machines virtuelles (solution choisie dans INDEED).

Afin de permettre un dimensionnement des ressources allouées aux clients et l'optimisation des coûts d'exploitation, les fournisseurs de Cloud doivent, d'une part, évaluer le coût de mise à disposition des ressources [55] et, d'autre part, évaluer le respect de la QoS (Qualité de Service) demandée selon les termes du contrat SLA (*Service Level Agreement*) passé entre client et fournisseur au regard des ressources allouées [56].

L'approche globale suivie est celle des boucles autonomiques MAPE-K (*Monitor, Analyze, Plan, Execute - Knowledge*) [239][57]. Dans notre contexte, nous nous sommes appuyés sur les étapes suivantes (voir Figure 4, partie administration): (1) le **Monitoring** ou la supervision des E/S des VMs et **l'analyse** des charges générées, (2) le **Planning**, ou l'étude du placement des VMs sur le système de stockage hybride, pour ce faire, une étape d'évaluation du coût de l'exécution de ces VM sur un système de stockage donné doit être réalisée à la suite de laquelle, l'optimisation du placement est faite [58][59], et enfin (3) **l'exécution** du plan de placement.

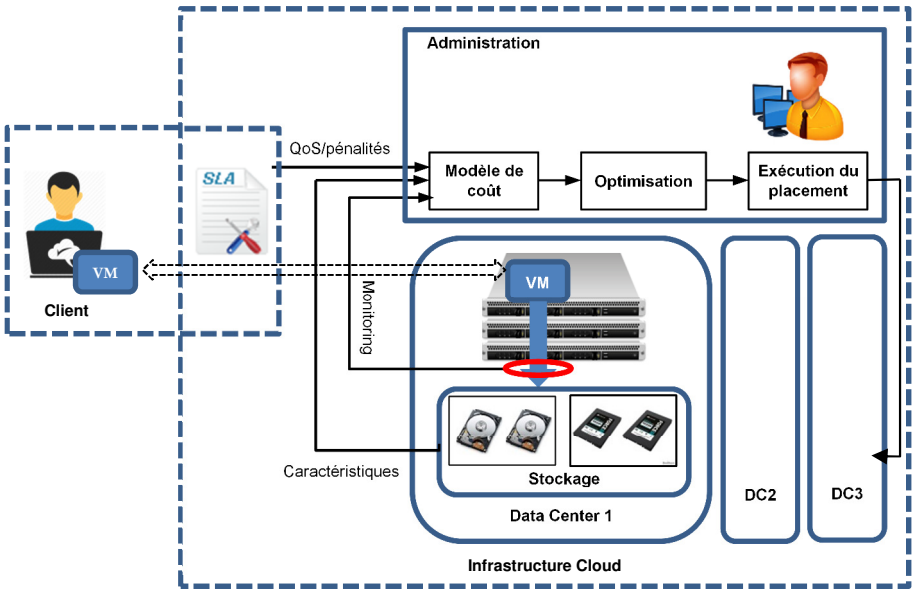


Figure 4. Architecture Cloud

Pour ce qui concerne la modélisation du coût global de l'exécution d'une VM, nous prenons en compte les propriétés du système de stockage, de la trace d'exécution de la VM et du SLA. La phase d'optimisation a pour objectif de minimiser le coût du placement des VMs sur les systèmes de stockage pour le fournisseur tout en garantissant la qualité de service, et ce en s'appuyant sur le modèle de coûts précédemment définis.

Cloud DBaaS

Le travail concernant l'intégration des mémoires flash dans le cadre d'un Cloud de type DBaaS (*DataBase as a Service*) est réalisé dans le cadre d'une collaboration avec l'USTHB (voir section 2.11). Un financement a été obtenu pour mener à bien ce projet (voir PHC Tassili GHEEMaS dans la section 2.8).

L'approche suivie dans le cadre de ce projet est la même du point de vue de l'administrateur (fournisseur de service) par rapport à celle décrite dans la Figure 4. Les principales différences résident dans : (1) l'application visée : en effet, nous ne considérons plus le placement d'une simple VM mais des objets de bases de données d'utilisateurs [60] (par exemple une table ou un index). (2) Une conséquence directe de la gestion des objets plutôt que des VMs est le nombre d'éléments à placer/stocker qui est beaucoup plus important. La technique d'optimisation à utiliser ne peut donc pas être la même, (3) le type de SLA visé sera fonction de l'application et du contexte applicatif.

Cloud et IoT

Ce travail est réalisé dans le cadre d'un projet avec Orange Labs (voir section 2.8) qui a débuté en octobre 2015. L'objectif est de proposer une approche centrée sur le contenu pour la gestion des données des objets connectés. Dans le contexte Orange, l'architecture de stockage peut tirer parti de l'architecture réseau par une maitrise de la localisation des données dans le réseau afin d'optimiser leurs latences et débits (E/S). Cette localisation peut être gérée d'une manière fine entre l'objet et le Cloud et dans le data center. Nous avons, pour ce faire, utilisé une architecture de type ***Fog*** dans laquelle une partie des ressources entre les objets et le Cloud est utilisée pour le stockage des données (ex : les *gateways*, *Local* et *regional Point of Presence*).

Deux points majeurs sont abordés dans ce projet : (1) le placement automatique de la donnée dans le *Fog* en fonction de l'accès à son contenu dans le tiers de stockage le plus adéquat en termes de performance, et ce en considérant le système de bout en bout, de l' « objet » jusqu'au data center [61], (2) la proposition de mécanismes permettant de générer différents niveaux de cohérence au sein de l'architecture de stockage proposée. Les performances de lecture et d'écriture d'une donnée dépendront du type de cohérence choisi.

2.4.3.2 Intégration des mémoires flash pour les Bases de Données.

Pour les applications de bases de données, il s'agit aussi de prendre en compte l'hétérogénéité des systèmes de stockage afin d'optimiser les performances d'accès aux données. Pour ce faire, nous intervenons sur trois axes : (1) étudier et modéliser des coûts d'exécution des requêtes sur les nouveaux supports de stockage de type flash [62][63], (2) revisiter les algorithmes de tri au vu de leur optimisation pour un système de stockage de type flash dans le cadre de Bases de Données (BD) haute performance [64], et (3) étudier les implications des piles logicielles du système d'exploitation dans le cadre des BD hautes performances et proposer des alternatives [65].

Le point (1) est abordé partiellement via la thèse de M. Salmi (voir la section 2.10.1). Ce même point, ainsi que les points 2 et 3 sont abordés avec l'entreprise KoDe Sotfware au travers d'une convention CIFRE, M. Laga a commencé son travail de thèse en décembre 2014.

2.4.4 Production scientifique

Mon travail sur cet axe a débuté fin 2012, et pour certains des domaines abordés, est encore à l'état embryonnaire (e.g. IoT, BD haute performance et stockage).

Production scientifique : 4 articles dans des revues internationales et chapitres de livre, 9 articles en conférence internationale et 3 articles de workshop.

Revues et Chapitres	Conférences internationales	Conférences nationales et ateliers
[ri1][ri2][ri12][ch1]	[ci1][ci2][ci4][ci5][ci6][ci9][ci12][ci13][ci20]	[w1][w4][w6]

2.5 Bilan

Ma thématique centrale de recherche à court et moyen termes est celle de l'optimisation de la performance et de la consommation énergétique des systèmes de stockage avec un accent sur l'aspect système, et ce, pour différents domaines d'applications (embarqué, Cloud, Bases de données, IoT).

Par rapport aux axes de recherche préalablement cités :

- Axe 1, Synthèse haut niveau pour architectures reconfigurables et outillage (2007 – fin 2008) : cette thématique a été abandonnée.
- Axe 2, Systèmes d'exploitation embarqués, performance et consommation énergétique (2009 – à ce jour): Mon implication dans cet axe de recherche était, au départ, centrée sur les systèmes d'exploitation et pouvait concerner plusieurs services. Avec le temps, mon centre d'intérêt s'est orienté vers les systèmes de stockage, et l'étude des systèmes d'exploitation à des buts d'optimisation des performances. Cet axe a donc naturellement rejoint l'axe 3.
- Axe 3, Optimisation des systèmes de stockage (2012 – à ce jour): cette axe représente désormais mon axe de recherche principal avec plusieurs projets en cours (Orange, Kode, PHC Tassili, IRT). Il s'agit principalement

de permettre à différentes applications de tirer profit de l'hétérogénéité des systèmes de stockage en remontant de la connaissance des périphériques de stockage vers l'applicatif et le système.

2.6 Perspectives de recherche

Avec l'explosion du volume de données numériques, la mutation des usages des données, le développement de nouvelles technologies de mémoire non-volatile en plus de la mémoire flash, plusieurs perspectives de recherche sont ouvertes. Les deux premières tendances impliquent un fort besoin en analyse de données qui ne peut être réalisée sans lecture/écriture performante, alors que la 3ème tendance nous pousse à repenser la manière dont sont gérés les périphériques de stockage au niveau système, voire même les méthodes d'accès aux données au niveau applicatif.

2.6.1 Perspectives à court termes : poursuite des projets en cours

Plusieurs projets ont démarré récemment et vont durer 2-3 ans. Ces projets collaboratifs visent des applications différentes, à savoir IoT, Cloud et Bases de données. Il s'agit donc de comprendre l'implication de l'hétérogénéité des systèmes de stockage sur ces différentes applications et services et de proposer des optimisations tant au niveau applicatif que système.

2.6.2 Perspectives à moyen et long termes : Intégration des mémoires non-volatiles

Plusieurs mémoires non-volatiles (NVM pour Non-Volatile Memories) ont émergées ces dernières années [66]. La plus connue est la mémoire flash NAND qui a permis de combler partiellement l'écart de performance qui se creuse entre la mémoire principale et le système de stockage. Les les plus étudiées ces dernières années : *Phase Change Memory* (PCM ou aussi PRAM), *Resistive Memory* (ReRAM), *Spin Torque Transfer Memory* (STT-RAM), *Ferroelectric Memory* (FeRAM), etc. Toutes ces mémoires présentent des propriétés intéressantes leur permettant de s'insérer dans la hiérarchie mémoire, telles que la consommation énergétique statique presque nulle, la densité de stockage, et les très bonnes performances en lecture. Néanmoins, toutes ces NVM ne sont pas au même stade de maturité.

D'un point de vue architectural et système, une nouvelle NVM peut être insérée dans la hiérarchie mémoire dans l'une des trois positions traditionnelles : à savoir au niveau des caches de processeur (SRAM), au niveau de la mémoire principale (DRAM), ou au niveau de la mémoire secondaire (disque durs). L'intégration de ces mémoires peut se faire de manière horizontale ou verticale [67]. Une intégration horizontale signifie que la NVM est intégrée au même niveau qu'une mémoire existante et que lors des différentes opérations (lecture ou écriture), le système a le choix quant au type de mémoire à utiliser. Lors de l'intégration verticale, la NVM est positionnée entre deux niveaux de mémoire existants, elle décale donc l'un des deux niveaux et agit comme mémoire tampon.

Du point de vue du système de stockage, une NVM peut être intégrée horizontalement ou verticalement. Tout comme pour la mémoire flash, certains mécanismes de gestion doivent être implantés et abstrait aux couches de plus haut niveau afin de permettre une intégration simple au sein de la pile logicielle traditionnelle de gestion du système de stockage.

Comme plusieurs NVMs ont des caractéristiques en performance comparables à celle de la DRAM, elles peuvent être intégrées à des niveaux plus hauts de la hiérarchie mémoire tel que la mémoire principale. Les NVMs ont des caractéristiques en performance assez spécifiques et différentes des mémoires volatiles telles que la SRAM et la DRAM. En effet, les performances sont asymétriques [66]: les lectures sont plus performantes que les écritures. L'endurance peut varier d'une manière significative d'une NVM à l'autre et est davantage impactée par les opérations d'écriture. Enfin, la plupart des NVM ne consomment de l'énergie que lorsqu'elles sont accédées, c'est d'ailleurs une propriété très intéressante permettant un gain en énergie qui peut être très important. Toutes ces caractéristiques doivent être prises en compte afin de pouvoir intégrer ces NVM comme mémoire principale ou avec de la DRAM afin de constituer une mémoire principale hybride. Avec une DRAM, une NVM peut être intégrée verticalement ou horizontalement. Dans le premier cas, la DRAM peut être considérée comme un cache pour la NVM dans le but de réduire les latences d'accès à la mémoire principale, ou d'avoir moins d'impact sur l'endurance de la NVM. La NVM peut aussi être intégrée horizontalement, sur le même bus que la DRAM. La gestion du placement de données peut être réalisée à trois niveaux : (1) au niveau matériel, (2) au niveau système d'exploitation, ou (3) au niveau applicatif. Dans le premier cas, l'hétérogénéité de la mémoire est abstraite au système d'exploitation et aux applications, et dans le second cas elle est abstraite aux applications.

Dans le cadre de l'intégration de ces NVMs dans les systèmes informatiques, je souhaiterais me baser sur la même approche orthogonale considérant le triplet application, système, architecture :

- **Architecture** : Du point de vue architectural, il s'agit d'explorer l'intégration des NVM dans la hiérarchie mémoire actuelle. Si l'on prend l'exemple de la PCM, la performance et l'endurance étant à mi-chemin entre la DRAM et la flash, elle peut être intégrée aux deux niveaux. Il s'agira aussi d'explorer les mécanismes de placement de données dans le cas d'une intégration horizontale et les mécanismes de cache dans le cas d'une intégration verticale permettant de tirer profit de la nouvelle architecture. L'évaluation des approches proposées peut se faire selon plusieurs critères : performances (débit, latence), consommation énergétique, endurance, etc.
- **Système** : Du point de vue système, il s'agit de réévaluer la structure des piles logicielles du système d'exploitation pour prendre en compte les nouvelles NVMs. En effet, la pile des E/S a été développée avec comme hypothèse une latence de l'ordre de la microseconde pour le stockage, ce qui ne sera plus le cas avec les NVM. Il s'agira donc de proposer de nouvelles structures plus efficaces permettant de tirer profit des performances des NVM. Par ailleurs, si la mémoire principale devient persistante, plusieurs mécanismes devront être partagés entre la mémoire et le sous-système de stockage (par exemple le système de fichiers). Les mécanismes classiques de gestion de la mémoire doivent aussi être revisités : segmentation, mémoire virtuelle, différents niveaux de cache, etc.
- **Application** : les NVM offrent la possibilité de pouvoir lire et écrire les données par octets, cette propriété se révèle très importante pour des applications ayant des charges d'E/S aléatoires qui nécessitent de charger un volume de données importants (blocs) pour n'en utiliser qu'une partie. Il s'agit donc ici de revisiter l'optimisation des applications au vue des nouvelles propriétés des NVM : accès par octets, asymétrie des performances et de la consommation énergétique, et endurance.

Systèmes de stockage et HPC

Je souhaiterais, dans le futur, développer une thématique autour de l'optimisation du stockage dans les plateformes de calcul haute performance. Dans ce cadre, l'aspect passage à l'échelle au niveau des services du système devient primordial. L'un des services les plus impactés est le système de fichiers. Ce dernier doit pouvoir évoluer avec le volume de données et sa performance avec les performances du stockage sous-jacent et la charge d'E/S appliquée. Lustre est un exemple de système de fichiers parallèle qui met en avant la scalabilité et les performances. Il s'agira d'étudier l'intégration des NVM au regard des systèmes de fichiers parallèles et des attentes du HPC.

2.7 Positionnement

2.7.1 Positionnement en France

Peu de laboratoires en informatique en France abordent les systèmes de stockages et mémoires non-volatiles comme thématique principale de recherche. En effet, on peut noter trois thématiques impliquant l'étude des systèmes de stockage (voir Figure 5) : **thème 1** : certaines équipes travaillent sur la gestion des données en prenant en compte des contraintes du système de stockage (en l'occurrence la mémoire flash), C'est le cas de l'INRIA Paris-Rocquencourt, projet SMIS (M. Bouganim) ou l'équipe « Bases de Données » du LIRIS UMR5202, INSA Lyon (M. Petit), **thème 2** : des équipes travaillent sur l'aspect architecture de stockage distribué et pour le calcul haute performance et /ou pour le Cloud, telles que l'INRIA KerData (M. Antoniu), CEA DAM (M. Deniel), LI-Parad Université de Versailles (M. Jalby), LIPN UMR 7030, Université de Paris XIII (M. Cérin), LIP6 Université Paris 6 (M. Sens), et enfin 3), **thème 3** : des équipes travaillent sur l'aspect technologique et microélectronique pour les mémoires non-volatiles (MRAM, EEPROM de type flash), telles que le LIRMM UMR 5506 département microélectronique (M. Torres), IM2NP UMR7334, équipe « mémoires », ou encore au sein du CEA LETI.

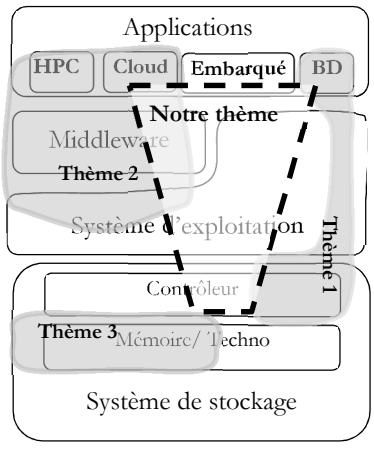


Figure 5. Positionnement dans le paysage français

Notre thématique de recherche est orthogonale et centrée sur le stockage comparativement aux thèmes préalablement cités. Contrairement aux thèmes 1 et 2, nous ne sommes pas localisés sur un applicatif particulier, mais pouvons complémenter ces thématiques. Je considère les applications comme différents cas d'études ayant des objectifs et contraintes qui peuvent être différents. Aussi, nous sommes à plus haut niveau que le thème 3 et en sommes donc complémentaires dans la perspective de l'intégration d'un système complet.

2.7.2 Thématiques connexes à l'international

Au niveau européen, plusieurs équipes travaillent sur la thématique du stockage, certains dans le domaine du calcul haute performance tels qu'ICS FORTH (équipe d'A. Bilas), J. Gutenberg Universtät (équipe d'A. Brinkmann), Hamburk Universtät (T. Ludwig), l'UP Catalunya (T. Cortes), IBM Storage (T. Haas), ou encore le CERN storage group (M. Lamana). D'autres se focalisent principalement sur l'optimisation du stockage pour les applications de bases de données tels que l'EPFL (DAIS et LABOS, respectivement les équipes de A. Ailamaki et de W. Zwaenepoel) ou encore IT Univ. Copenhagen (P. Bonnet).

Sur le continent américain, et plus spécifiquement aux Etats Unis, il existe un grand nombre de laboratoire travaillant sur la thématique du stockage, nous n'en citerons que certains: G. Ganger (Carnegie Mellon University), E. Zadok (Stony Brook University), A. Arpaci Dusseau (University of Wisconsin-Madison), D. Long (University of California, Santa Cruz), M. Zhao (Arizona State University), M. Seltzer (Harvard University), T. Li (University of Florida), S. Swanson (University of California, San Diego).

Enfin, en Asie, la thématique du stockage de données s'est beaucoup développée, surtout depuis l'essor des mémoires flash et des mémoires non-volatiles, en partie grâce à l'existence d'industriels très importants du domaine comme Samsung. Nous pouvons citer les équipes suivantes : E. Sha (Chongqing University, Chine), H. Chen (Shanghai Jiao Tong University, Chine), J. Shu (Tsinghua University), Z. Shao (PolyU Hong Kong), J. Xue (City U Hong Kong), T. W. Kuo (National Taiwan University), S. H. Noh, (UNIS, Corée du sud), Y. Won (Hanyang University, Corée du sud).

2.8 Projets de recherche et contrats industriels

Cette section introduit l'ensemble des projets sur lesquels j'ai pu travailler depuis mon intégration à l'UBO. Cela comprend un projet européen, un projet ANR, un projet de type partenariat Hubert Curien (PHC), trois projets avec des industriels (XanKom, Kode Software et Orange) ayant donné lieu à deux conventions CIFRE, deux projet IRT (successifs) et un projet exploratoire interne UBO. Ces activités sont résumées dans le Tableau 2. Sur ce tableau, plutôt que de donner les montants levés pour chaque projet, j'ai préféré citer explicitement ce que le projet m'a permis de financer (voir la colonne « Financement »).

Tableau 2. Projets menés.

Projets	Années de participation :	Type de projet	Rôle	Financement	Partenaires
FALCON	2017-2020	IRT b<>com	Membre, montage	Thèse + Fonctionnement	IRT b<>com
GHEEMaS	2016-2019	PHC Tassili	Co –initiateur, et Coordinateur	Mobilités de chercheurs+ 2 thèses en co tutelle	USTHB Algérie, CERIST Algérie
BaD FLASH	2015-2018	Contrat de prestation/CIFRE	Co –initiateur et Resp. scientifique	Thèse CIFRE + Fonctionnement	KoDe Software
Orange	2015-2018	Contrat de prestation/CIFRE	Co –initiateur et Resp. scientifique	Thèse CIFRE + Fonctionnement	Orange Labs
INDEED	2013-2016	IRT b<>com	Membre, montage, resp. d'un wp	Thèse + Fonctionnement	IRT b<>com
Open PEOPLE	2011-2012	ANR	Membre	Fonctionnement	INRIA Lilles, LORIA, Univ. Bretagne Sud, Univ. Rennes 1, Univ. Nice SA, Thalès Comm., InPixal

Think FPGA	2010-2011	Projet exploratoire (inter équipe UBO)	Co –initiateur / co- resp. scientifique	Stage M2+ Fonctionnement	Laboratoire LISyC (maintenant intégré au Lab-STICC)
XanKom	2009-2010	Contrat de prestation	Initiateur/resp. scientifique	Stage M2	PME XanKom
MORPHEUS	2007-2008	Européen FP6	Membre	Fonctionnement[3]	Thales France, Ace associated compiler experts b.v. Netherlands, Alcatel-Lucent Germany, Universita di Bologna Italy, Arttic France, CEA France, Critical blue UK, Thomson Germany, Intracom s.a. telecom solutions Greece, M 2000 France, Pact xpp technologies Germany, STMicroelectronics Italy, Technische Universitaet Braunschweig Germany, Technische Universitaet chemnitz Germany, Technische Universiteit Delft Netherlands, Universitaet Karlsruhe Germany[4]

La Figure 6 montre le séquencement des projets sur lesquels j'ai travaillé (granularité d'un an). Les projets notés en gras sont ceux pour lesquels j'ai contribué largement au montage et/ou à la coordination.

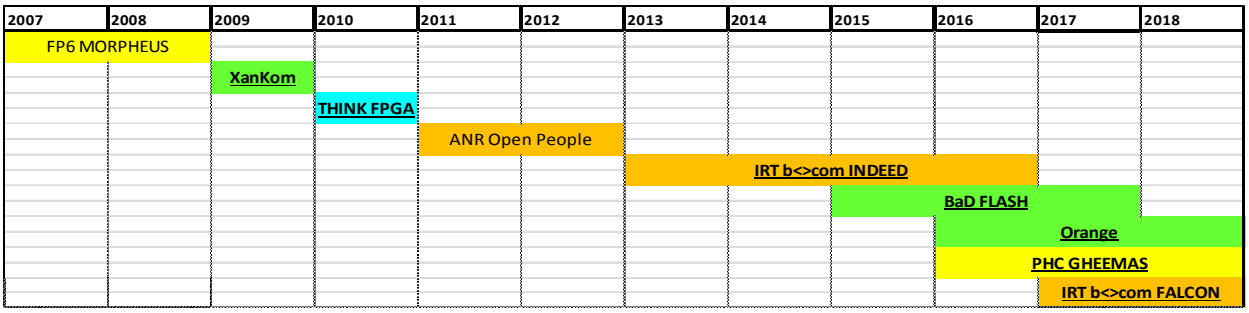


Figure 6. Diagramme temporel indicatif des projets menés depuis mon intégration à l'UBO.

2.8.1 FALCON - IRT b<>com (2017-2020)

L'idée de base du projet FALCON est l'exploitation et la revente de ressources Cloud inutilisées afin de réduire le coût TCO d'infrastructures privées. La revente est prévue en appui sur la technologie *Blockchain*. Ce projet adresse plusieurs verrous scientifiques : (1) du point de vue système, cela concerne l'évaluation au plus juste des ressources à utiliser et des ressources disponibles, le développement de méthodes de migrations efficaces, et l'étude de l'isolation des performances dans un environnement de matériel hautement hétérogène. C'est dans ce cadre que l'on va se focaliser sur l'apport des disques SSD pour l'isolation performances dans le cas de l'usage de containers. (2) le développement de solutions sûres pour la protection de workloads et le développement de solution *Blockchain*, 3) la gestion de l'interopérabilité dans une solution IaaS hétérogène ainsi que l'orchestration de taches dans un système dynamique.

Rôle : j'ai participé au montage du projet et je participe à l'encadrement de l'étudiant en thèse financée par l'IRT, M. Jean-Emaile Dartois dont j'ai co défini le sujet.

2.8.2 PHC (Partenariat Hubert Curien) Tassili GHEEMaS (2016-2019)

Le projet GHEEMaS (A Green and Highly Efficient and Evolutionary data and storage Management System for Cloud DataBase as a Service) rentre dans le cadre du Cloud Computing et du Big Data et plus spécifiquement, les services de bases de données dans le Cloud (DBaaS). Les applications visées sont celles des entrepôts et de l'analyse de données. Deux tendances majeures se conjuguent actuellement: (1) l'explosion exponentielle des données numériques, (2) le déport de traitement et de stockage vers le Cloud (EMC prédit aussi que vers 2020, un tiers des données numérisées passeront par le Cloud). La convergence de ces deux tendances fortes révèle de nouveaux défis liés au stockage et au traitement de l'information, respectivement évoquées par les termes *Big Storage* et *Big Data*. Le projet GHEEMaS se propose d'optimiser le coût lié aux ressources utilisées par le Cloud et l'empreinte énergétique en

[3] « Participation » sur ce tableau signifie que le projet a financé essentiellement ma participation à des déplacements, et divers achats de matériel.
[4] http://cordis.europa.eu/project/rcn/80670_en.html

attaquant ces défis sur trois axes majeurs : (1) la réduction du volume de données à traiter dans les bases de données dans le Cloud en utilisant des techniques de Big Data Mining, (2) l'utilisation intelligente des nouvelles technologies de stockage, i.e. mémoire flash, peu consommatrices en énergie au travers d'un placement de données optimisé, (3) le dimensionnement efficace, automatique et évolutif des ressources matérielles et logicielles utilisées.

Rôle : Co-initiateur et coordinateur du projet (côté français). 12 personnes (5 permanents UBO, 2 permanents USTHB Alger, 1 permanent CERIST Alger, 4 étudiants en thèse), ~50 hommes/mois, ~35K€.

2.8.3 Projet Orange (2015-2018)

La convergence du Cloud et de l'internet des objets crée de nouvelles opportunités. La gestion du stockage dans un scénario ou l'on couplerait l'usage du Cloud pour le stockage et les objets connectés nécessite un système de stockage global réactif et répondant à différents usages (dû à la pluralité des objets et des services). Cette réactivité est difficile dans un contexte où la donnée peut être stockée sur différents objets et/ou différents serveurs de stockage.

Afin d'optimiser la réactivité du système de stockage dans ce type de scénario, nous proposons, dans ce projet, d'agir principalement sur deux métriques :

- La performance : L'objectif ici est de proposer une architecture de stockage à tiers de bout en bout qui va du stockage de la données sur l' « objet » et jusqu'au data center avec des techniques de placement des données basées sur le contenu.
- La cohérence : en effet, en permettant au système de stockage de « relaxer » la cohérence selon certains critères liés au contenu, on peut agir sur la réactivité du système de stockage, car :
    - on peut utiliser la version de donnée permettant l'accès le plus rapide, même si cette dernière n'est pas tout à fait à jour.
    - on réduit le trafic du système de stockage en réduisant le trafic dû aux mises à jour qui peut être très important dans les systèmes.

Rôle : Co initiateur et responsable scientifique, encadrant de l'étudiant en thèse CIFRE M. Mohammed Islam Naas. 4 personnes (2 permanents de l'UBO, 1 permanent de Orange, 1 étudiant en thèse), ~18 hommes/mois, ~30K€

2.8.4 FLASHBaD – KoDe[5] Software (2015-2018)

L'objectif du projet FlashBaD (Mémoire flash et Bases de Données) est d'optimiser les performances des E/S du moteur de base de données de l'entreprise KoDe Software. Pour ce faire, nous nous proposons d'agir à deux niveau: (1) niveau système et (2) niveau du système de gestion des bases de données (SGBD). Pour le niveau système, nous proposons de modifier la pile d'E/S à deux niveaux. Au niveau des caches VFS: algorithmes de *read-ahead* et de *write back*, il s'agira d'optimiser ses mécanismes de gestion de cache selon la charge d'E/S et le système de stockage sous-jacent. En effet, l'étude menée dans le cadre de nos travaux à démontré des écarts de performance considérables en fonction de la configuration de ce cache. Nous nous proposons aussi d'agir au niveau de l'ordonnancement des requêtes d'E/S au niveau du pilote de périphérique et ce en fonction du système de stockage sous-jacent. Pour le niveau SGBD, l'objectif est de pouvoir appliquer les optimisations les plus adéquates en fonction du système de stockage utilisé et de ses réactions face aux requêtes envoyées par le SGBD. Il faudrait donc introduire une boucle rétroactive permettant d'appliquer des optimisations en fonction de la réaction du système de stockage à la charge d'E/S appliquée.

Rôle : Co initiateur, et responsable scientifique, encadrant de l'étudiant en thèse CIFRE, M. Arezki Laga. 5 personnes (2 permanents UBO, 2 permanents de KoDe, 1 étudiant en thèse), ~20 hommes/mois, ~35K€

2.8.5 INDEED - IRT b<>com (2013-2016)

Le projet INDEED (Infrastructure de Nuage informatique Distribuée et Econome en Energie) vise à étudier différents aspects relatifs au déploiement, au fonctionnement et à l'exploitation d'une infrastructure de nuage informatique (*Cloud Computing*) répartie offrant des garanties en termes de sécurité des informations stockées/traitées/véhiculées et de fiabilité/disponibilité des infrastructures virtualisées IT et télécom. La solution

[5] L'entreprise a changé de nom en 2016 pour prendre celui de *Koeus*.

fournira une QoS et une qualité d'expérience (QoE) adaptées aux différents clients adressés, au mode d'accès (e.g. Internet ou VPN) et au terminal (TV, PC, smartphone…) utilisé. Une attention particulière est également portée aux aspects consommation d'énergie lors de la définition et de la mise en œuvre des architectures sous-jacentes.

Les principaux résultats attendus du projet sont: (1) un middleware d'orchestration pour l'administration des data centers et l'accès avec garanties (SLA, sureté, …) quel que soit le type de réseau ou de terminal ; (2) des applications rendant possible l'administration intégrée des ressources informatiques et télécoms. Cette administration prendra en compte l'utilisation à la demande et à la dynamicité des ressources disponibles. (3) des interfaces applicatives (API) pour des offres de type SaaS (*Software as a Service*).

Nous sommes principalement intervenus sur la partie optimisation et orchestration du stockage dans le Cloud.

Rôle: j'ai participé au montage du projet, j'ai été responsable d'un *work package* et j'ai participé à l'encadrement de l'étudiant en thèse financée par l'IRT, M. Hamza Ouarnoughi dont j'ai co défini le sujet.

2.8.6 ANR Open PEOPLE (2009-2012)

Le projet Open-PEOPLE (*Open-Power and Energy Optimization PLatform and Estimator*) se propose de fournir une plateforme fédératrice ouverte pour l'estimation et l'optimisation de la consommation en puissance et en énergie des systèmes. L'objectif du projet était que les utilisateurs de la plateforme puissent estimer la consommation d'une application sur une architecture matérielle choisie parmi un ensemble d'architectures types paramétrables. Au sein du système réalisé, les composants utilisés sont choisis dans une bibliothèque de composants matériels et logiciels. L'estimation peut se faire à différentes étapes du raffinement de la spécification, la méthodologie développée, reposant sur une utilisation de modèles de consommation multi-niveaux, interopérables, et interchangeables, permettant une exploration facile de l'espace de conception.

À cet effet, la plateforme d'estimation logicielle, accessible par l'intermédiaire d'un portail Internet, est couplée à une plateforme matérielle constituée d'un banc de mesure automatisé, commandée via la plateforme logicielle. Une bibliothèque d'applications (benchmarks) est proposée pour la caractérisation des nouveaux composants et des nouvelles architectures. En plus des travaux de recherche nécessaires à l'établissement des méthodes pour l'estimation multi-niveaux des systèmes hétérogènes complexes, des travaux de recherche ont été menés afin de proposer des méthodes et techniques pour permettre l'optimisation de la consommation à partir des résultats fournis par Open-PEOPLE.

Rôle : Membre pendant ~2ans, j'ai rejoint le projet à la fin de la première année.

2.8.7 Think FPGA (2010-2011)

Le projet avait pour objectif de définir les principes de mise en place d'un RTOS, en l'occurrence μCOSII, à base de composants (Think) sur une architecture FPGA pour des applications embarquées de contrôle (domotique, robotique, etc). Cette approche a pour objectif de fournir un parallélisme d'exécution entre l'application et le RTOS, et au sein même de l'application en mettant en œuvre un ordonnancement temporel et spatial sur le FPGA. Il s'agissait d'allier la dynamicité de l'approche logicielle à base de composants à la flexibilité des architectures reconfigurables dans le but de permettre une très grande souplesse tant au niveau logiciel que matériel, et cela, associé à une amélioration des performances. L'approche a été démontrée en réalisant le portage d'un service du RTOS sur le FPGA, en l'occurrence le service de gestion de temps qui s'est montré plus précis et flexible dans son implantation matérielle.

Rôle : Co responsable scientifique et co initiateur, 4 personnes (3 permanents UBO, 1 stagiaire M2), ~10 hommes/mois, ~3000€.

2.8.8 Projet XanKom (2009-2010)

Le projet avec l'entreprise XanKom consistait à étudier les possibilités de portage d'un système d'exploitation, en l'occurrence GNU Linux, sur une carte spécifique qui devait être embarquée dans un équipement de liaison radiofréquence respectant le standard 802.11 (Wifi). Cet équipement est un terminal destiné à permettre le raccordement hertzien d'un abonné à l'arête dorsale (*backbone*) d'un réseau étendu. Le système d'exploitation RouterOS proposé par défaut avec la carte de routage RB411, utilisée par XanKom, est propriétaire et les paramétrages qu'il autorise sont vastes et peu adaptés aux besoins de l'entreprise. Le projet s'est décomposé en deux parties distinctes. Il

s'agissait tout d'abord de prendre en main le matériel et de mettre en place la chaîne de développement croisée nécessaire à la production du système d'exploitation. Cette première partie s'est soldée par une démonstration de la mise en œuvre minimaliste de la carte en permettant son démarrage à partir du noyau Linux et l'exécution de quelques services classiques. La seconde phase du projet constitue la part la plus importante du sujet proposé et concerne l'étude du portage effectif du système d'exploitation GNU Linux présentant les fonctionnalités attendues.

Rôle : Responsable scientifique et initiateur du projet, 3 personnes (1 permanent UBO, 1 stagiaire M2, 1 permanent XanKom), ~8 hommes/mois, ~3500€.

2.8.9 FP6 MORPHEUS (2006-2008)

Le déploiement à large échelle des systèmes embarqués ainsi que la complexité croissante des applications, posent de nouveaux défis en termes de puissance de calcul, de réduction des coûts de développement, de consommation énergétique, et de flexibilité. Ceci résulte d'une augmentation de la complexité des plateformes matérielles, et par conséquent, une baisse de l'efficacité des outils de développement et de programmation, ce qui rallonge les temps de mise sur le marché. Le projet MORPHEUS (*Multipurpose Dynamically Reconfigurable Platform for Intensive and Flexible Hetereogeneous Processing: A technology breakthrough for Embedded Computing*) a tenté de répondre à ce défi en développant une solution globale à base d'une plateforme de SoC complexe, hétérogène et contenant plusieurs unités de calcul dont certaines reconfigurables dynamiquement et de granularités différentes. D'autre part, MORPHEUS devait aussi proposer un ensemble d'outils intégrés et homogènes pour le développement spatial et séquentiel sur le SoC proposé.

Rôle : Membre pendant 18 mois

2.9 Collaborations industrielles

Les collaborations industrielles sont résumées dans la section 2.8, les entreprises concernées sont les PMEs XanKom et KoDe Software et des grands groupes tels qu'Orange.

Nous avons aussi entamé une collaboration avec DDN Storage, une entreprise fournissant des solutions de stockage hautes performances au travers de l'organisation d'un workshop francophone sur le stockage (voir la section 2.12). Cette collaboration a débouché sur un travail commun entamé en mars 2017.

Une autre collaboration a été entamée fin 2016 avec Yahoo ! (Ca, USA) sur l'optimisation des mémoires qui a débouchée sur une publication [w2]. Cette collaboration évolue vers un travail commun sur le thème du Big Data.

2.10 Encadrements

Dans cette section, sont résumés les encadrements d'étudiants en thèse ainsi que des étudiants en fin d'études (2ème année de master et dernière année d'école d'ingénieurs).

2.10.1 Encadrement de doctorants

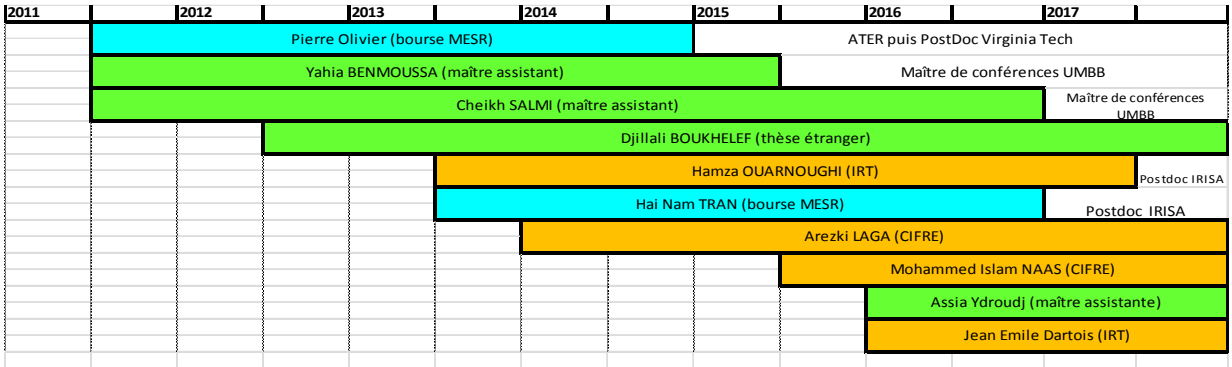


Figure 7. Diagramme temporel représentant les étudiants en thèse encadrés depuis mon intégration à l'UBO. Pour plus de précision sur les dates de début et de fin, se référer à la section suivante.

2.10.1.1 Thèses soutenues

Pierre Olivier (Octobre 2011-Décembre 2014, 39 mois)

Type de financement : Bourse de thèse MESR

Encadrants : Eric Senn (directeur de thèse, UBS Lorient), Jalil Boukhobza (~70%)

Membres du jury :

Rapporteur Mme Cécile Belleudy, Maître de conférences HDR à l'Université Nice Sophia Antipolis,
Rapporteur M. Luc Bouganim, Directeur de recherche à l'INRIA Paris-Rocquencourt
Examinateur M. William Jalby, Professeur à l'Université de Versailles St Quentin en Yvelines
Examinateur M. Frank Singhoff, Professeur à l'Université de Bretagne Occidentale
Examinateur M. Gaël Thomas, Professeur à Télécom SudParis
Examinateur M. Pierre Ficheux, Ingénieur dans l'entreprise OpenWide

Sujet : Estimation de performances et de consommation énergétique de systèmes de stockage à base de mémoire flash dans les systèmes embarqués

Ce travail cible l'évaluation des performances et de la consommation énergétique du service de stockage secondaire dans un système d'exploitation embarqué utilisant une mémoire flash NAND. Les contributions apportées dans cette thèse s'articulent autour d'une méthodologie de modélisation pour l'estimation des performances et de la consommation des systèmes de stockage embarqués de type FFS (*Flash File System*). Cette méthodologie est divisée en trois phases. En phase d'exploration nous identifions, via des micro-benchmarks, les éléments du système de stockage qui impactent les performances et la consommation énergétique. En phase de modélisation, cet impact est représenté sous la forme de modèles de différents types (par exemple : fonctionnels, de performances et de consommation). En phase de simulation, les modèles sont mis en œuvre dans un simulateur, permettant d'obtenir des estimations concernant les performances et la consommation d'un système de stockage à base de mémoire flash soumis à une charge d'E/S donnée.

Ce travail de thèse a été réalisé dans le cadre de l'axe 2.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri3][ri8][ri11][ri13]	[ci16][ci18]	[w4][w9][w11] [cn2][cn3][cn5]

Yahia Benmoussa (Mai 2011-Juin 2015, durée de 4 ans prévue dans la co-tutelle)

Type de financement : L'étudiant était maître assistant à l'Université Mhamed Bougara, Boumerdès (UMBB) Algérie. La thèse a été réalisée dans le cadre d'une co-tutelle que j'ai initié.

Encadrants : Eric Senn (directeur de thèse, UBS Lorient), Jalil Boukhobza (~60%), côté algérien : Djamel Benazzouz

Membres du Jury :

Rapporteur M. Daniel Menard, Professeur à l'Université de Rennes 1
Rapporteur M. Daniel Chillet, Maître de Conférences HDR, à l'ENSSAT
Examinateur M. Smail Niar, Professeur à l'Université de Valenciennes
Examinateur M. Frank Singhoff, Professeur à l'Université de Bretagne Occidentale
Examinateur M. Mouloud Koudil, Professeur École Supérieure d'Informatique, Alger, Algérie

Sujet : Modélisation et caractérisation de la consommation d'énergie du décodage vidéo sur SoC multi-cœurs hétérogènes et leurs applications

Ce travail de thèse propose une méthodologie pour la caractérisation et la modélisation des performances et de la consommation d'énergie du décodage vidéo, aussi bien sur des processeurs à usage général de type ARM que sur un processeur de traitement de signal (DSP). L'étape de caractérisation est basée sur une méthodologie expérimentale pour évaluer de façon exhaustive les performances et la consommation d'énergie du décodage vidéo. Les mesures obtenues dans cette étape ont permis de construire des modèles de performance et de consommation d'énergie pour le décodage vidéo. Les modèles proposés peuvent estimer avec une grande précision la performance et la consommation d'énergie de décodage vidéo en fonction d'un nombre de paramètres comprenant la qualité de la vidéo et la fréquence du processeur. De plus, en se basant sur une caractérisation multi-niveaux et une approche de modélisation par décomposition en sous-modèles, ce travail a montré comment les modèles développés, contrairement aux modèles empiriques classiques, sont facilement transposables à d'autres plateformes matérielles.

Ce travail de thèse a été réalisé dans le cadre de l'axe 2.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri6][ri7][ri10]	[ci10][ci14][ci15]	[w8][w10]

Hai Nam Tran (Septembre 2013 – Janvier 2017, 41 mois)

Type de financement : Bourse de thèse MESR

Encadrants : Frank Singhoff (directeur de thèse), Stéphane Rubini, Jalil Boukhobza (~20%)

Membres du Jury :

Rapporteur M. Pascal Richard, Professeur à l'Université de Poitiers
Rapporteur M. Laurent Pautet, Professeur à Telecom Paritech.
Examinateur M. Giuseppe Lipari, Professeur à l'Université de Lilles.
Examinateur M. José Rufino, Professeur associé à l'Université de Lisbonne

Sujet : Analyse de systèmes embarqués temps-réel en présence de mémoire cache

La démocratisation des processeurs contenant plusieurs niveaux de cache dans les systèmes embarqués temps réels nous pousse à repenser et à adapter la théorie de l'ordonnancement et de proposer de nouvelles méthodes de validation prenant en compte ce composant. Il s'agit dans cette thèse de proposer (1) une approche de modélisation de système temps-réel avec une mémoire cache et de l'intégrer dans un outil de simulation d'ordonnancement, en l'occurrence Cheddar, (2) des solutions d'ordonnancement qui prennent en compte l'usage par les tâches de la mémoire cache et l'interaction que cela implique sur les temps d'exécution, et enfin, (3) une méthodologie permettant d'investiguer l'intervalle de faisabilité d'un système avec cache.

Ce sujet de thèse s'inscrit dans l'axe 2.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri4]	[ci7][ci11]	[cn1][w3][w7]

Cheikh Salmi (Octobre 2011 – Mars 2017[6])

Type de financement : L'étudiant est maître assistant à l'Université Mhamed Bougara, Boumerdès (UMBB) Algérie.

Encadrants : Ladjel Bellatrèche (directeur de thèse, ENSMA Poitiers), Jalil Boukhobza (~20%)

Membres du Jury :

[6] Cet étudiant en thèse était maître assistant tout le long de sa thèse assurant ainsi un service d'enseignement complet.

Rapporteur M. Claude GODART, Professeur, LORIA, Université de Lorraine
Rapporteur M. Kokou YETONGNON, Professeur, LE2I , Université de Bourgogne
Examinateur M. Djamal BENSLIMANE, Professeur, IUT, Université Lyon1
Examinateur M. Mohamed MEZGHICHE, Professeur, Université de Boumerdes, Algérie
Examinatrice Mme. Béatrice MARKHOFF, Maître de conférences, Université François Rabelais Tours

Sujet : Vers une Description et une Modélisation des Entrées des Modèles de Coût Mathématiques pour l'Optimisation des Entrepôts de Données

Les entrepôts de données (ED) sont devenus une technologie mature. L'accentuation des demandes d'analyse est motivée par l'évolution technologique, les nouveaux paradigmes de programmation et l'ingénierie Dirigée par les Modèles (IDM). Durant la phase d'exploitation; des modèles de coût mathématiques sont utilisés pour quantifier la qualité des solutions proposées. Le développement de ces derniers nécessite des efforts de collection et d'analyse des paramètres pertinents. Pour bien simuler le fonctionnement d'un ED, toutes les dimensions d'un SGBD doivent être intégrées. Dans cette thèse, nous proposons de décrire en détail ces dimensions par la définition d'un méta-modéle. Vu la similarité et la hiérarchisation qui existent entre les supports de stockage, nous avons développé une ontologie de domaine dédiée aux supports de stockage. Elle permet d'expliciter leurs propriétés. Les similarités entre ces supports nous a motivé à hybrider le cache mémoire avec les mémoires flashs pour augmenter sa capacité afin de stocker un nombre important de résultats intermédiaires partagés par plusieurs requêtes décisionnelles. La réutilisation de ces résultats permet d'augmenter la performance du SGBD. Nos contributions sont validées à l'aide des expérimentations en utilisant nos modèles de coût théoriques et le SGBD Oracle.

Ce sujet de thèse s'inscrit dans l'axe 3.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
	[ci9][ci13]	

Cette liste de publications ne comprend que les celles auxquelles j'ai participé.

2.10.1.2 Thèses en cours

Jean Emile Dartois (Octobre 2016-)

Type de financement : L'étudiant est doctorant salarié dans le cadre d'un projet de l'IRT b<>com.

Encadrants académiques : Olivier Barais (IRISA, Rennes, directeur de thèse), Jalil Boukhobza (~50%)

Sujet : Gestion de ressources matérielles hétérogènes et garantie de service dans le Cloud, le cas des containers.

L'objectif de cette thèse est d'améliorer les technologies de conteneur Linux en leur offrant une visibilité sur les ressources matérielles sous-jacentes disponibles. Cette thèse étudie comment fournir un modèle abstrait utilisé à l'exécution pour prendre des décisions automatiques dans un environnement matériel hétérogène. Un focus particulier sera apporté au traitement et aux performances de la mémoire et des systèmes de stockage en respectant la qualité de service demandée par l'utilisateur et l'état global du système.

Ce sujet de thèse s'inscrit dans l'axe 3.

Assia Ydroudj (Février 2016-)

Type de financement : L'étudiante est maître assistante à l'Université d'Alger

Encadrants : Kamel Boukhalfa (directeur de thèse, USTHB), Jalil Boukhobza (~40%), Stéphane Rubini

Sujet : Contribution des mémoires non-volatiles dans l'optimisation des Big databases/warehouses dans un environnement Cloud.

Deux tendances majeures se conjuguent actuellement dans le domaine du traitement de la donnée: l'explosion exponentielle des données numériques et le déport de traitement et stockage vers le Cloud. La convergence de ces deux tendances couplée à l'émergence de nouvelles mémoires non-volatiles révèle de nouveaux défis liés au stockage et au traitement de l'information. L'objectif de cette thèse est d'investiguer l'apport des nouvelles technologies de mémoires non-volatiles (PCM, ReRAM) dans le cadre d'applications Cloud. Seront étudiés en particulier les applications de Big data sur des frameworks dédiés comme SPARK.

Ce sujet de thèse s'inscrit dans l'axe 3.

Mohammed Islam Naas (Novembre 2015-)

Type de financement : CIFRE avec Orange Labs Rennes

Encadrants académiques: Alain Plantec (directeur de thèse), Jalil Boukhobza (~90%)

Sujet : Stockage Cloud pour l'Internet des Objets, une approche centrée sur les contenus pour la cohérence et l'optimisation des performances.

L'objectif de cette thèse est de proposer une approche centrée sur le contenu (« Content Centric ») pour la gestion des données des objets connectés dans un Cloud. Le stockage n'est pas déterminé à priori, il est choisi en fonction de la donnée et de son utilisation. De plus, dans le contexte Orange, l'architecture de stockage peut tirer parti de l'architecture réseau par une maitrise de la localisation des données dans le réseau afin d'optimiser leurs latences et débits (I/O). L'apport d'une telle solution permettra de séparer la gestion du stockage des services. C'est au système de stockage de s'adapter pour mieux répondre aux besoins. Il en découle aussi que les infrastructures de stockage seront naturellement mutualisées entre services.

Ce sujet de thèse s'inscrit dans l'axe 3.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
	[ci1]	

Arezki Laga (Décembre 2014-)

Type de financement : CIFRE avec l'entreprise KoDe Software

Encadrants académiques : Frank Singhoff (directeur de thèse), Jalil Boukhobza (~80%)

Sujet : Optimisation de performance des Entrées/Sorties sur système de stockage à base de mémoire flash pour les bases de données haute performance

Dans le cadre des bases de données hautes performances, les E/S représentent historiquement un goulet d'étranglement très important et qui peut négativement impacter l'effort d'optimisation réalisé au niveau du SGBD. L'objectif de cette thèse est de reconsidérer la pile logicielle de gestion des E/S et de proposer des mécanismes d'optimisation et de garantie des performances d'E/S interagissant étroitement avec le SGBD afin de tirer profit des technologies de systèmes de stockage actuelles à base de mémoire flash.

Ce sujet de thèse s'inscrit dans l'axe 3.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri1]	[ci5]	[w4]

Hamza Ouarnoughi (Octobre 2013-)

Type de financement : L'étudiant est doctorant salarié dans le cadre d'un projet de l'IRT b<>com.

Encadrants : Frank Singhoff (directeur de thèse), Jalil Boukhobza (~40%), Stéphane Rubini

Sujet : Système de stockage distribué adaptatif, et hétérogène dans le contexte d'un Cloud distribué

Le but de la thèse est de proposer un système de gestion de stockage hydride intégrant des technologies hétérogènes de stockage de type magnétique et flash. Différentes architectures sont à évaluer afin de retenir la plus pertinente dans le cadre d'un cloud distribué. Des modèles de coût pour les solutions proposées sont développés. Enfin, ces modèles de coût sont utilisés afin de proposer des méthodes d'optimisation du placement de données dans un Cloud permettant d'en minimiser l'empreinte énergétique et le coût.

Ce sujet de thèse s'inscrit dans l'axe 3.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri2]	[ci6]	[w1][w6]

Djillali Boukhelef (Janvier 2012 - [7])

Type de financement : L'étudiant est financé par l'Université des Sciences et Techniques de Houari Boumédiène (USTHB), Alger, Algérie.

Encadrants : Kamel Boukhalfa (directeur de thèse, USTHB), Jalil Boukhobza (~40%)

Sujet : Placement des données sur stockage hybride et optimisation des entrepôts de données dans le Cloud

Les fournisseurs de Cloud actuels offrent le service DataBase-as-a-Service (DBaaS) pour le stockage et l'exploitation des bases de données offrant ainsi aux utilisateurs une forme d'accès à une base de données sans la nécessité de mise en place de matériel physique, d'installation de logiciels ou de configuration. L'objectif de la thèse est de proposer une infrastructure de placement des objets des différentes bases de données dans un système de stockage hybride afin de répondre aux exigences des clients à moindre coût.

Ce sujet de thèse s'inscrit dans l'axe 3.

Publications :

Revues	Conférences internationales	Conférences nationales et ateliers
[ri2]	[ci2][ci4]	

2.10.2 Encadrement de stages de fin d'étude

Ci-joint la liste des étudiants en stage de fin d'étude que j'ai encadré.

Mohammed Bey Ahmed Khernache (2017, 6 mois, stage recherche de M2 Logiciels pour Systèmes Embarqués – Brest), « Optimisation des algorithmes de tri pour des mémoires non-volatiles émergentes (NVM) », financement sur contrat.

Oussama Elhamer (2017, 6 mois, stage recherche de M2 Logiciels pour Systèmes Embarqués – Brest), « *Qualifying SSD Endurance and Performance* », financement par l'industriel DDN storage.

Mohamed Chakib Belgaïd (2015, 5 mois, stage de PFE ingéniorat – ESI, Alger, Algérie), « Développement et test de mécanismes de gestion de mémoire flash (FTL) sur carte OpenSSD », financement sur contrat.

Hamza Ouarnoughi (2013, 6 mois, stage recherche de M2 Logiciels pour Systèmes Embarqués - Brest), « Prise en compte des aspects énergétiques dans la conception des bases de données sur les mémoires Flash », collaboration LISI (ENSMA, Poitiers) – Lab-STICC (Brest), financement laboratoire. Production scientifique : [ri12][ci12].

Pierre Olivier (2011, 5 mois, stage recherche de M2 Logiciels pour Systèmes Embarqués - Brest), « Développement d'une architecture de gestionnaire de temps matériel pour RTOS pour intégration dans un système à composants THINK ». Financement sur le projet THINK FPGA. Production scientifique : [cn3].

[7] Il est à noter que la durée de cette thèse est plus longue que la normale du fait de l'activité professionnelle de l'étudiant. En effet, il a travaillé à plein temps pendant un peu plus de 2 années en début de thèse.

Ilyes Khetib (2011, 5 mois, stage recherche de M2 Logiciels pour Systèmes Embarqués - Brest), « Développement et test d'un simulateur de système de gestion d'accès à la mémoire flash de type cache+FTL ». Financement sur fond de laboratoire. Production scientifique : [ci20][cn4].

Bi Shen Yi (2010, 5 mois, stage professionnel de M2 Logiciels pour Systèmes Embarqués - Brest), « Portage d'un Linux Embarqué sur RouterBoard 411 », financé sur contrat avec l'entreprise XanKom.

Khalifa Rouis (2010, 6 mois, stage recherche de M2 recherche en informatique - Rennes), « Etude des performances des systèmes de stockage à base de mémoire flash ». Financement sur fond de laboratoire. Production scientifique: [ci23].

En plus des encadrements de stage d'étudiants de 2ème année de Master, j'ai aussi encadré plus d'une dizaine d'**étudiants de M2 dans des projets de recherche** d'une durée allant de 2 à 4 mois.

2.11 Collaborations académiques

Collaboration actives :

- **Université Polytechnique de Hong Kong (PolyU), Hong Kong, Chine** : initiée en août 2015 au travers d'une mobilité de deux semaines, cette collaboration se poursuit grâce à un **CRCT de 6 mois** que j'ai effectué entre aout 2016 et janvier 2017. Les thématiques abordées sont liées à l'optimisation des systèmes de stockage de type flash (ou autre NVM) pour des applications de Big Data (par exemple SPARK).
- **Université de Sciences et Techniques Houari Boumediene (USTHB), Alger, Algérie** : Je travaille très étroitement avec des enseignants chercheurs de l'USTHB (laboratoire LSI), sur le stockage dans le Cloud dans le cadre d'application de type DBaaS (voir thèse de Djillali Boukhelef et projet PHC Tassili GHEEMaS). J'effectue régulièrement des séjours d'une semaine depuis quelques années.
- **Université Mhamed Bougara, Boumerdès (UMBB), Boumerdès, Algérie**: à la suite d'une invitation pour un séminaire en 2010, une collaboration a été démarrée et a rapidement débouché sur l'encadrement commun d'un étudiant en thèse, M. Yahia Benmoussa (voir section 2.10.1 Encadrement de doctorants). La collaboration se poursuit actuellement avec M. Benmoussa (qui a soutenu sa thèse en 2015).

Collaborations passées :

- **Université Abou Bakr Belkaïd, Tlemcen, Algérie**: le directeur de département informatique de l'Université de Tlemcen (Pr. Abdelkrim Benamar) a été invité à passer une semaine (décembre 2012) dans notre laboratoire. Une convention cadre a été signée.
- **Université de Mérida, Mérida, Venezuela:** Un professeur de l'Université de Mérida (Pr. Armando Borrero) a effectué un séjour de recherche de 2 mois (septembre-novembre 2011) dans le laboratoire suite à mon invitation. Un travail sur l'étude statistique des traces d'E/S (stockage secondaire) a été initié.

2.12 Animation de la recherche

- **Co-président (co-chair) et initiateur d'EWiLi** (2011-*), The Embedded operating systems workshop
  Je copilote l'organisation de ce workshop depuis sa création en 2011. Un résumé des six éditions passées est présenté dans le tableau ci-dessous. Ce workshop gagne en popularité et en qualité d'année en année.

	EWILI'16	EWILI'15	EWiLi'14	EWiLi'13	EWiLi'12	EWiLi'11
Emplacement et date	Pittsburgh, US, 6 Octobre	Amsterdam, Pays Bas, 8 Octobre	Lisbonne, Portugal, 13-14 Novembre	Toulouse, France, 26-27 Aout	Lorient, France, 7 Juin	St Malo, France, 11 Mai
Dutée	1 jour	1 jour	1 jour et demi	1 jour et demi	1 jour	1 jour
Co-organisé avec	ESWEEK (Embedded System Week)	ESWEEK (Embedded System Week)	Standalone	ETR: Ecole D'été du Temps Téel	Standalone	Renpar, CFSE, SympA (atuellement COMPAS)
Conférenciers invités	**Pr. Daniel Mossé** (University of Pittsburgh)	**Pr. Tei-Wei Kuo** (National Taiwan University), **Julien Marechal** (ingénieur R&D Thales TCS)	**Dr.Oleg Sokolsky** (Univ of Pennsylvania), **Dr.Helder Silva** (EDISOFT), **M.Pierre Ficheux** (Openwide)	**Dr.Robert Davis** (Univ. of York), **Dr.Joel Sherrill** (OAR Corporation)	**Dr José F. Ruiz** (AdaCore), **Dr.Eric Senn** (Lab-STICC, Univ. Bretagne Occidentale)	**Dr. John Williams** (Univ. of Queensland, and Fondateur de Petalogix)

Taux d'acceptation	50%	53%	65%	69%	80%	75%
Nombre de participants	~20	~20	~30	~30	~80 (including an industrial exhibition)	~50
Publication des actes	ACM SIGBED Review	ACM SIGBED Review	ACM SIGBED Review	ACM SIGBED Review	ACM SIGBED Review	Actes de Renpar, SympA, CFSE
Sponsors	Lab-STICC, Univ. Bretagne Occidentale, IBNM, AdaCore	ACM SIGOPS France, Lab-STICC, Univ. Bretagne Occidentale	ACM SIGOPS France, Lab-STICC, Univ. Bretagne Occidentale, FCT, AdaCore	ACM SIGOPS France, Lab-STICC, Univ. Bretagne Occidentale, Captronic, AdaCore	ACM SIGOPS France, Lab-STICC, Univ. South Brittany, Univ. Bretagne Occidentale, Lorient Technopole	ACM SIGOPS France, Lab-STICC, Univ. South Brittany, ISSTB

- Co-organisateur de **WOPSSS Workshop On Performance and Scalability of Storage Systems** (2016-*) organisé conjointement avec ISC High Performance.
- **Co-organisateur et co-initiateur du workshop Per3S** (2015-*), Performance and Scalability of Storage Systems, co-organisé avec DDN, CEA, et UVSQ (http://syst.univ-brest.fr/per3s/). L'objectif de ce workshop est d'organiser des rencontres en France autour de la thématique du stockage et des Entrées/Sorties.
- **Co-président de session (track program co-chair) de la conference IEEE/IFIP EUC** (2014), The International Conference on Embedded and Ubiquitous Computing. J'ai co-présidé une session concernant les applications pour l'embarqué.
- **Co-président de session (track program co-chair) de la conference ECSI DASIP** (2012), Conference on Design & Architectures for Signal & Image Processing. J'ai co-présidé une session sur les systèmes d'exploitation pour l'embarqué.
- **Membre du comité scientifique d'ECOFAC** (2015), Ecole thématique COnception FAible Consommation pour les systèmes embarqués et temps réel.
- **Co-président de communication (Publicity co-chair) de la conférence IEEE NVMSA** (2015, 2016), Non-Volatile Memory Systems and Applications symposium.

2.13 Rayonnement et responsabilités scientifiques

2.13.1 Responsabilités en recherche au sein de l'établissement

- **Animateur d'une thématique au sein de l'équipe MOCS, pôle CACS du Lab-STICC (depuis 2010),** cette thématique est liée à l'optimisation des systèmes de stockage. Après avoir initié cette thématique de recherche en fin 2010, j'en suis l'animateur et le participant principal. Actuellement, **4 enseignants chercheurs et 6 doctorants** participent aux travaux de recherche.
- **Représentant de l'UBO au sein du Conseil Scientifique de l'IRT b<>com (depuis fin 2016)**.

2.13.2 Participation à des jurys de thèse

J'ai été invité à rapporter sur la thèse de doctorat de **M. Nezer Jacob Zaidenberg** intitulée "Applications of Virtualization in Systems Design", étudiant de **l'Université de Jyväskylä**, **Finlande**. Cette thèse a été soutenue le 12 juin 2012.

2.13.3 Invitations dans des universités étrangères

J'ai été invité à plusieurs reprises pour des séjours dans des universités à l'étranger, seuls les séjours recherche sont cités (des séjours pour des cours ont été réalisés et sont détaillés dans la section 1.7.2) :

- Aout 2016 – Janvier 2017 (**semestre CRCT**), Aout 2015 (2 semaines) : à l'**Université Polytechnique de Hong Kong (PolyU), Hong Kong, Chine**. Equipe du Pr. Zili Shao.
- Novembre 2015, séjour d'une semaine, **Ecole Supérieure d'Informatique d'Alger (ESI), Algérie** ; Equipe du Pr. Mouloud Koudil.
- Mai 2016, Novembre 2015, Février 2015, Novembre 2014, Octobre 2013, séjours d'une semaine**, Université des Sciences et Techniques Houari Boumediene (USTHB), Alger, Algérie**, Equipe du Dr. Kamel Boukhalfa.

– Novembre 2011, séjour d'une semaine, **Université Mhamed Bougara, Boumerdès (UMBB), Boumerdès, Algérie**. Equipe du Pr. Djamel Benazzouz.

2.13.4 Participation à des comités de programmes de conférences

Les événements sont classés d'abord par nombre de participation, puis par date de participation.

- **EWiLi** (2011, 2012, 2013, 2014, 2015, 2016, 2017), The Embedded operating systems workshop
- **SPECTS** (2012, 2013, 2014, 2015, 2016, 2017), The International Symposium on Performance Evaluation of Computer and Telecommunication Systems
- **IEEE MC-SoC**, (2014, 2015, 2016, 2017) International Symposium on Embedded Multicore/Many-core System-on-Chip.
- **IEEE NVMSA** (2015, 2016, 2017), Non-Volatile Memory Systems and Applications symposium.
- **ASP-DAC** (2017, 2018), Asia and South Pacific Design Automation Conference
- **WOPSSS** (2016, 2017), workshop on performance and scalability of storage systems.
- **CITS** (2015, 2016), The International Conference on Computer, Information and Telecommunication Systems
- **IEEE/IFIP EUC** (2013, 2014), The International Conference on Embedded and Ubiquitous Computing.
- **ECSI DASIP** (2011, 2012), Conference on Design & Architectures for Signal & Image Processing
- **LCTES** (2017), ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems
- **ICESS** (2017), IEEE International Conference on Embedded Software and Systems.
- **INFLOW** (2016), Workshop on INteractions of NVM/FLash with Operating-systems and Workloads.
- **COMPAS** (2016), Conférence d'informatique en Parallélisme, Architecture, Système.
- **SETIT** (2016), Sciences of Electronics, Technologies of Information and Telecommunications.
- **ISPS** (2015), International Symposium on Programming and Systems
- **CADS** (2015), The CSI International Symposium on Computer Architecture & Digital Systems.
- **ICWIT** (2013), The International Conference on Web and Information Technologies.
- **CIIA** (2013), The International Conference on Computer Science and its Applications.
- **SDIWC DICTAP** (2011), The International Conference on Digital Information and Communication Technology and its Applications.

2.13.5 Participation à des relectures pour des conférences (relecteur externe)

- **DAC** (2017), Design Automation Conference.
- **ICDT** (2017), International Conference on Database Theory
- **IEEE MASCOTS** (2006, 2013), International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.

2.13.6 Modérateur de session de conférence

J'ai participé à la modération de session de plusieurs conférences : **IEEE NVMSA** (2015, 2016), **IEEE/IFIP EUC** (2012), **SDIWC DICTAP** (2011), **SPECTS** (2011), **ICEE** (2009)..

2.13.7 Participation à des comités de lecture de journaux internationaux

Les journaux sont classés d'abord par nombre d'années de participation aux comité de lecture, puis par date de participation.

- IEEE Transactions On Computers (2014, 2015, 2016)
- IEEE Transaction On VLSI Systems (2014, 2015, 2016)
- Elsevier Journal of System Architecture (2014, 2015, 2016)
- ACM Transactions on Embedded Computing System (2008, 2016, 2017)
- ACM Transactions On Storage (2015, 2016)
- Elsevier Future Generation Computer Systems (2015, 2016)
- IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2016, 2017)
- Wiley Concurrency and Computation: Practice and Experience (2017)

- IET Electronics Letters (2016)
- Wiley Journal of Software: Practice and Experience (2015)
- IEEE Embedded Letters (2014)
- Springer Distributed and Parallel Database (2014)
- Springer Journal of Zhejiang University - Science C (2014)
- MDPI Computers (2014)
- Springer Journal of Supercomputing (2013)
- Springer Journal of Real-Time Image Processing (2012)

2.13.8 Relecture de chapitres de livre

- **Computer Organization and Architecture**, *10th Edition*, 2015, **William Stallings**, Pearson, relecture de 2 chapitres à la demande de l'auteur.
- **Organization, Design, and Architecture***, 5th Edition,* 2013, **Sajjan G. Shiva,** Taylor and Francis**,** relecture de 2 chapitres à la demande de l'éditeur.
- **Embedded Computing Systems: Applications, Optimization, and Advanced Design, 2013, IGI,** Mohamed Khalgui, Olfa Mosbahi, Antonio Valentini**,** relecture d'un chapitre à la demande de l'éditeur.

3 RESPONSABILITES COLLECTIVES

3.1 Responsabilités au sein du département et de l'UFR Sciences et Techniques

Depuis mon recrutement, j'ai eu l'opportunité d'assumer différentes responsabilités collectives résumées dans cette section :

- 2014-2017 : Responsable adjoint de la filière scientifique du département informatique (~180 étudiants), cette filière n'existera plus au lancement de la nouvelle dernière accréditation (sept. 2017).
- Depuis 2013 : Représentant du département informatique à la commission internationale de l'UFR (~2500 étudiants, ~270 enseignants chercheurs)
- Depuis 2012 : Membre du Conseil du département informatique (~350 étudiants, ~40 enseignants chercheurs)
- 2012-2017 : Membre du Conseil de filière scientifique du département informatique (~180 étudiants)
- Depuis 2011 : Référent pour les inscriptions d'étudiants étrangers au département informatique (campus France)

3.1.1 Responsable adjoint de la filière scientifique du département informatique (2014-2017) :

J'ai été nommé responsable adjoint de la filière scientifique du département. Il s'agit, d'animer, d'organiser, et de promouvoir la filière scientifique du département constituée d'environ 180 étudiants.

3.1.2 Représentant du département à la commission internationale de l'UFR (2013-) :

J'ai été nommé, par le conseil de département informatique, comme représentant à la commission internationale de l'UFR. Il s'agit de promouvoir les collaborations avec les universités étrangères. Cette commission se réunit une fois par an afin de classer les dossiers de demande de financement pour professeurs invités, et s'occupe de centraliser/diffuser l'information concernant les collaborations de l'UFR avec les universités étrangères et de les promouvoir.

3.1.3 Membre du Conseil de département informatique (2012-):

Je suis membre du conseil de département depuis 2012, et j'ai ainsi pu faire face aux aspects très concrets de la vie du département telles que les aspects financiers, la campagne des postes, l'évolution des formations, les questions pédagogiques, le montage des dossiers d'habilitation/accréditation.

3.1.4 Membre du Conseil de filière scientifique du département informatique (2012-2017) :

Je suis membre du conseil de la filière scientifique du département informatique en tant que responsable du M2LSE. Dans ce cadre, j'ai l'opportunité de participer à la réflexion sur des questions pédagogiques et sur l'évolution des formations et de la vie de la filière scientifique du département. Les filières n'existeront plus en 2017 à la suite de la réorganisation des diplômes du département informatique.

3.1.5 Référent pour les inscriptions d'étudiants étrangers au département informatique (2011-) :

Je suis le référent au niveau du département informatique pour ce qui concerne l'inscription des étudiants étrangers (campus France). Les procédures étant peu stables au début, je suis l'interface entre le département informatique et le service qui gère les étudiants internationaux de la Direction des Etudes et de la Vie Etudiante (DEVE). Je considère que l'accueil des étudiants étrangers est un plus pour notre établissement en général et pour notre Master en particulier, tout autant qu'une opportunité pour les étudiants étrangers qu'il faut soutenir et pour laquelle il faut faciliter l'accès.

3.2 Expertise scientifique

- **Agence Nationale de la Recherche** ANR, Expert pour le programme « Ingénierie Numérique & Sécurité» - INS 2013, 1 projet expertisé.
- Membre de la **commission de sélection** de l'UBO, 27ème section, recrutement de maître de conférences 2013, poste n° 4148.

3.3 Bilan des responsabilités collectives

Depuis ma titularisation, mon investissement au sein du département informatique a été croissant. À la suite des responsabilités pédagogiques occupées, j'ai été nommé dans les différents conseils cités précédemment avec comme objectif la participation active à la vie du département. La responsabilité de la 2ème année de master m'a permis de m'insérer dans le conseil de filière, puis dans le conseil de département. Enfin, on m'a nommé responsable adjoint de la filière scientifique. Je me suis aussi particulièrement investi dans le développement des relations internationales du département. Je me suis vite impliqué dans le recrutement et l'accueil des étudiants étrangers, le montage de conventions avec des établissements étrangers, et les relations internationales de manière générale, au niveau du département et de l'UFR. En effet, j'estime que la mobilité entre établissements, celles des étudiants ou des chercheurs, est nécessaire au développement et au rayonnement du département et du laboratoire et que cela nécessite un investissement et une prise en charge particulière.

Part 2: General introduction

1 RESEARCH CAREER BRIEF DESCRIPTION

Associate professor since 2006, I joined the team "Architecture and System" (A & S) of the LESTER laboratory (Laboratory of Electronics of TImE Real Systems), a joint laboratory between the University of South Brittany (UBS) and the University of Western Brittany (UBO) that was associated with the CNRS between January, 1st in 2004 (FRE 2734) and 2008. The team located in Brest was recognized for its expertise in the development of synthesis tools for reconfigurable embedded systems based on software engineering techniques. I started my research by working on European project MORPHEUS FP6, between early 2007 and late 2008. I participated in the team's work on high-level synthesis tools that allowed to port applications developed in several different high level languages on many reconfigurable architectures in a complex System on Chip (SoC).

In 2008, the LESTER laboratory merged with other laboratories to create Lab-STICC (Laboratory of Information Sciences and Techniques, Communication and Knowledge), currently CNRS UMR 6285. I am member of the "Communications, Architectures, Circuits and Systems" (CACS) group within this laboratory, and more precisely working in the "Methods and Tools, Circuits and Systems" (MOCS) team. Following the European project MORPHEUS, I worked on an area related to modeling and optimization of energy consumption in embedded systems. Within this area of research, I was particularly interested in two case studies: (1) energy consumption and performance of video streaming applications for mobile systems, investigated in collaboration with an Algerian University (see section 2.1) and (2) performance and energy consumption modeling and simulation of embedded flash memory-based storage systems, initiated within the context of the ANR project Open-PEOPLE (Open Platform for Estimation and Optimization Power and Energy Consumption). The common point of these two studies is the approach followed (detailed later) and their positioning at the embedded operating system level. During this period, I also conducted other studies in the area of flash memory storage, thus initiating a new topic of research in our laboratory. This topic has become my main area of research.

In 2011, I extended the scope of my research on storage systems, and more generally to the study of memory hierarchy for different application domains such as Cloud computing (Distributed and energy efficient Cloud infrastructure project at the IRT b<>com), high performance database systems (BaD-Flash project with *Kode Software*[8] company), and Internet of Things (a project with Orange company).

Currently, I spend 20% of my working time for the Institute of Research and Technology (IRT) b<>com, investigating storage systems management in a Cloud context.

In the section that follows, I will describe the research work initiated since I was recruited according to the following three areas of research:

- Research area 1 (2007 - 2008): Introduction to high level synthesis and tools for reconfigurable architectures
- Research area 2 (2009 - present): Embedded operating systems, performance, and energy consumption
- Research area 3 (2012 - present): Optimization of storage systems

Note that this organization mainly follows a chronological order, and is therefore necessarily unbalanced in terms of investment and scientific production. The first research area was an introductory one that I investigated during my first years at UBO, and consequently will not be developed in the rest of the document. Indeed, very early in my career, I initiated a new research topic in my laboratory related to the field of storage system (research area 2 and research area 3) which has become the main focus of my research. **This document focuses on research areas 2 and 3**.

Figure 8 summarizes the main research areas and topics addressed and intersections between them. It also links projects and PhD thesis related to each research area.

[8] Renamed *Koeus* in 2016

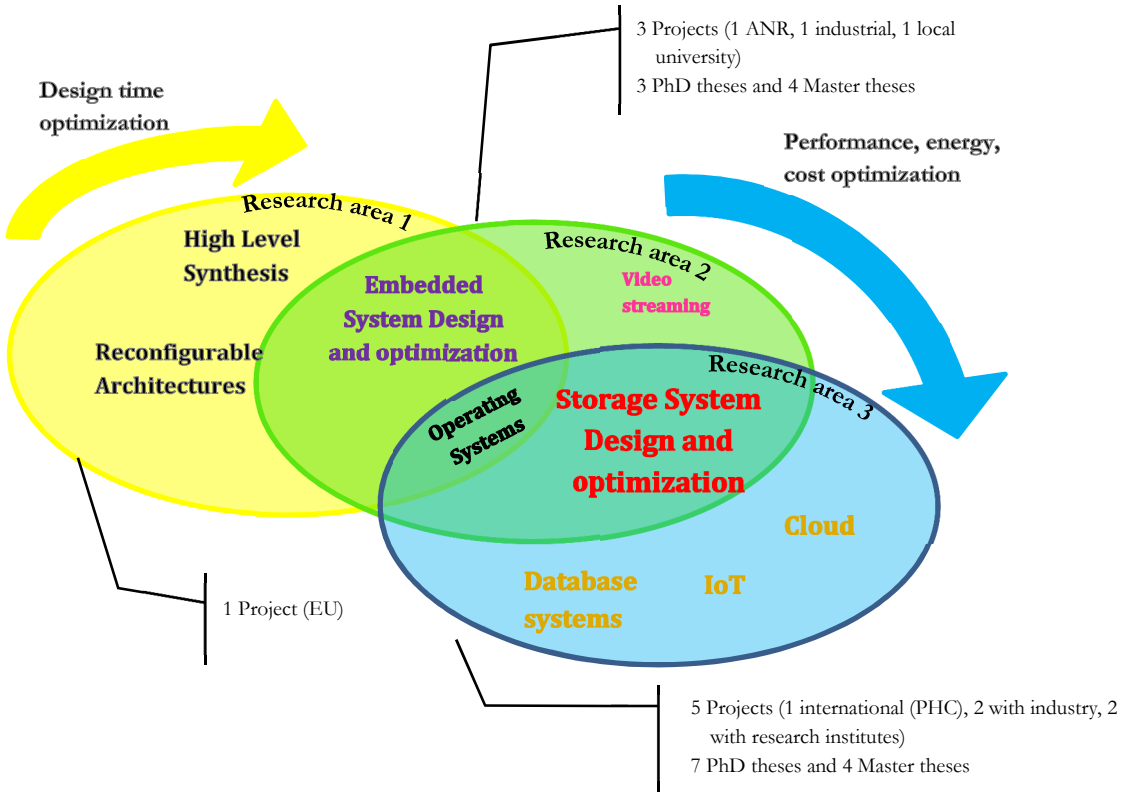


Figure 8. Research topics addressed since my integration at UBO

Table 2 details the research activities carried out since 2007 on different areas of interest. For each area, I listed the research topics, application domains and finally the scientific methodologies and tools used to conduct my research.

Table 2. Time diagram representing the application domains, research topics and research methods and tools used.

	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Application domains	Embedded systems		Embedded systems video streaming			Embedded systems		Cloud	IoT		Big Data
Research topics	Reconfigurable systems High level synthesis		Operating systems Energy consumption Storage systems Flash memory			Operating systems Energy consumption Storage systems Flash memory Database systems		Real-time systems			
Research methodology and tools	Simulation Software engineering Compilers		Simulation Software engineering Modeling (performanceand energy) Performance measures and Benchmarkin Tools			Simulation Software engineering Modeling (performance and energy) Performance measures and Benchmarking Tools		Optimisation tools			

2 RESEARCH AREAS CHRONOLOGY

2.1 Research area 1: Introduction to high level synthesis and tools for reconfigurable architectures

2.1.1 Context

Reconfigurable architectures (such as FPGAs) and Networks on Chip (NoC) brought out challenging research areas that were widely investigated [1]. Current SoC platforms can be very complex and contain different types of computing elements on which a given application can run: general purpose processors (GPP), digital signal processors (DSP), reconfigurable units, etc. Indeed, a given application may have a software or hardware implementation that can be used according to some specific constraints. These may be related to performance or energy consumption, for example. The problem addressed was related to adaptive Hw/Sw co-design on heterogeneous platforms.

Ideally, on a complex SoC with multiple heterogeneous processing units, one can run an application on the most appropriate processing element according to system constraints and state. However, this is quite unrealistic, mainly because of the heterogeneity of the used tools [1]. Indeed, each processor/technology or Intellectual Property (IP) is handled by a different set of tools that are not always interoperable.

The purpose of the integrated European project MORPHEUS (IST 027 342) was, first, to propose a complex heterogeneous SoC that contains multiple processing elements, some of which are reconfigurable and having different granularities. MORPHEUS also aimed to propose a set of integrated and consistent tools for spatial and temporal development on the proposed SoC.

2.1.2 Problem statement

The use of heterogeneous reconfigurable SoC requires new software tools. The major problem that we have considered in the context of this research area was the homogenization and simplification, from user's point of view, of the use of such architectures in order to reduce development time, thus time-to-market. In fact, a tool chain should allow, relying on a homogeneous computing model, to easily migrate from a high-level specification to a hardware resource configuration [2].

In the context of MORPHEUS project, the objective of the A&S LESTER UBO team was to propose a solution to the above-cited problem under the form of: (1) a computing model, (2) a portable data structure that makes possible third-party tools integration in order to provide a complete software toolchain, and (3) a generic low level layer to perform synthesis on heterogeneous reconfigurable architectures. During the first years at UBO, I worked on the second issue with two team members.

2.1.3 Contribution

In this context, we have proposed a high-level synthesis toolchain to program heterogeneous SoCs. Developing a toolchain for reconfigurable and heterogeneous architectures requires the implementation of communicating tools at two levels:

1. An abstract level, or high level allowing, independently from any hardware target, to model an algorithmic representation of the computation including the control flow, the data flow and the program structure in the form of a hierarchical Control and Data Flow Graph (CDFG). ;
2. An implementation level, or low level, that ensures the configuration of a specific hardware target and data transfers, going from an algorithmic representation of the computations.

The high level layer consists of a software toolchain where each tool relies on the definition of the computation CDFG. This software chain is seen as a set of interoperable processes that operate in turn on a definition of the algorithmic representation (see Figure 9). The role of the low level layer tool is to translate the resulting CDFG into a specific architectural target, be it a fine grained (FPGA) or a coarse grained reconfigurable architecture.

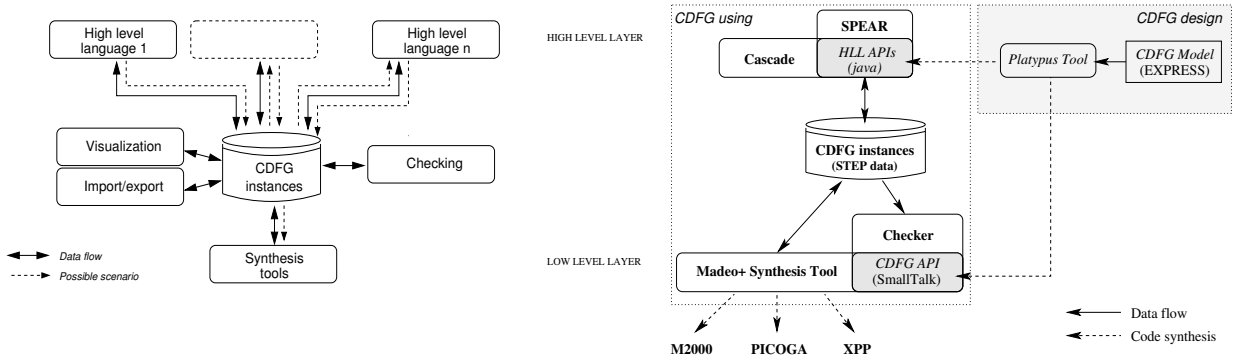


Figure 9. CDFG and tools

We chose to use a hierarchical CDFG as an intermediate representation that makes it possible to capture the control flow, the data flow, and the program structure. This would allow to apply optimizations more easily to the source code. I have worked on defining the CDFG model and on developing part of the associated tools [4][5][6].

In the context of MORPHEUS project, applications were generated from two distinct tools: (1) Cascade tool from Critical Blue that generated computation processes and (2) SPEAR tool [10] from Thales that generated communication processes (see the right part of Figure 9). Both tools used different programming languages; Java and C++. Thanks to Platypus [9], a meta modeling and a programming environment, we generated APIs in these different programming languages in order to reach a homogeneous representation of the CDFG written in STEP format [7] and the meta-model of which was described in EXPRESS [8], as shown in the right part of Figure 9. The CDFG representation was then used by the low level synthesis tool developed by another member of the team (using another programming language, i.e. Smalltalk), always through the use of the developed APIs.

2.1.4 Publications

My work on this introductory research area was achieved over a period of one year and a half. During this period, I had to familiarize myself with the topic, and with the approach and tools from the team I joined. A substantial development work was carried out. This period of familiarization and development work was a good introduction to the field of reconfigurable and embedded systems in general which was used to develop the 2nd research area.

Journals	International conferences	National conferences and workshops
	[ci24]	[cn6] [w12]

2.2 Research area 2: Embedded Operating Systems, performance, and energy consumption

2.2.1 Context

Research conducted on this area started in 2009. During this year, smartphone sales have exploded. This is one of the reasons that gave an extraordinary impetus to the development of various types of embedded systems. This is mainly due to the falling production costs. For instance, the cost of flash memory, which was around 10$/GB in 2007 fell by a factor 5 to reach 1.8 $/GB in 2010 and dropped below the symbolic 1$/GB in late 2011 [12].

Smartphones and embedded systems sales explosion has enabled diversification and complexification. This resulted in an increasing need for performance. Indeed, applications such as video streaming and social networking have led to a growing complexity and diversity of the hardware used: CPU, GPU, DSP or FPGA. It also led to an increase in the volume of multimedia data generated, which requires the use of efficient storage systems.

The complexity of the equipment used in embedded systems, coupled with diversification of applications put more pressure on the used embedded operating systems to manage efficiently hardware resources available in order to ensure an optimal Quality of Service (QoS) for the end user.

Another very crucial problem is that of energy consumption. In fact, the International Technology Road-map for Semiconductors (ITRS) predicted an increase in energy consumption by a factor of 2.5 for the next decade [11]. Unfortunately, the performance increase of batteries empowering these systems does not follow this pace [13].

2.2.2 Problem statement

In this context, optimizing performance and energy consumption of embedded systems necessarily requires a detailed and orthogonal understanding and analysis of the system behavior: **application**, **operating system**, and **hardware** used, but also the interactions between these different elements. In this research area, we focused on three main issues:

- The study of energy consumption in complex embedded system environments;
- The effective integration of flash memory-based storage subsystems in computer systems;
- The use of reconfigurable hardware for optimizing the operating system services (OS).

It should be noted that the last item follows issues covered in Research area 1.

2.2.3 Contributions

In this section, each subsection deals with one of the three issues addressed.

2.2.3.1 Performance and energy consumption of complex embedded systems

The objective of this work was mainly to analyze and understand the energy consumption of embedded systems. We chose two different case studies. The first case study we have considered focuses on the applicative level. Indeed, we considered one of the most used applications in the context of mobile computing which is **video streaming** [14][15]. In fact, video streaming is one of the most compute intensive and energy hungry application. We have also considered another study centered on a specific service of the Operating system (OS), but with a similar approach, which is the storage system management.

Both studies have been the subject of two PhD theses already defended: the one of Mr. Yahia Benmoussa and Mr. Pierre Olivier. The first PhD topic relied on a joint work with the University of Mhamed Bougara in Boumerdes (UMBB), Algeria. The second one is a MESR (Ministry of Higher education and Research) thesis. Both theses have benefited from the support of the ANR project named Open-PEOPLE and were supervised by Dr. Eric Senn from Lab-STICC at UBS, Lorient.

Both studies were conducted with a similar methodological approach summarized in Figure 10. This methodology is based on three steps: (1) an **experimental step** for the study of the performance and energy consumption, (2) a **modeling step**, (3) an **application step** in which the elaborated models were used whether for the purpose of optimization of certain performance metrics or for developing simulation tools.

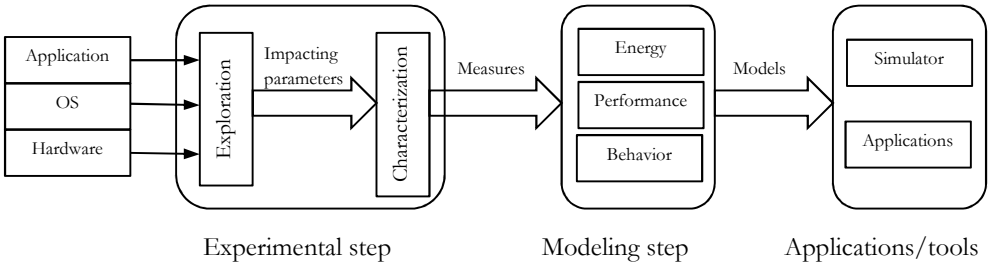


Figure 10. Global approach for performance and energy consumption study.

Energy consumption study of video streaming performance

For this application, the **experimental step** was based on a methodology for assessing comprehensively and at different levels, the performance and energy consumption of video decoding [16]. This experimental step was conducted on embedded platforms on which was tested a wide range of video decoding configurations in terms of resolution, complexity of the video, bit-rate, etc. Concerning system parameters, the energy consumption related to the operating system execution with different frequencies was measured. For the hardware part, we evaluated several embedded platforms in which many types of processors have been tested. This includes general-purpose processors (ARM) and a Digital Signal Processors (DSP). This step emphasized the importance of analyzing various parameters that may be related to different levels of a given system.

The measurements obtained in the experimental step have been used in **the modeling step**. The purpose of this step was to build **performance and energy consumption models** for video decoding, both on an ARM general-purpose processors and DSP. Developed models makes it possible to estimate performance and power consumption with a high accuracy (of the order of $R^2 = 97\%$) according to the video quality and the processor frequency and type [17]. In addition, based on a multi-level characterization and modeling approach using decomposition into sub-models, we showed how developed models, unlike conventional empirical ones, can be generalized to other hardware platforms [18].

In the scope of this study, in **the application step**, we have proposed some possible case studies using the developed models [20]. For instance, in the context of an adaptive video decoding. The idea is to exploit the ability of the proposed model to predict performance decoding time of a given video quality. This is done to better tune the used processing units for energy saving purpose. In addition, we proposed a methodology to extend the developed models to other hardware platforms.

This study is detailed in **Part 3 Chapter 2**.

Energy consumption study of embedded storage systems

For this study, in the **experimental step**, we have explored and identified the various elements of a storage system that impact performance and energy consumption. This includes hardware related parameters: flash memory, CPU and RAM. We have also investigated operating system related parameters corresponding to the I/O software stack of flash storage on Linux [20][21][22]: the Virtual File System (VFS), the file system, and the NAND driver. Of course, the impact of these hardware and system elements depends on the behavior of the executed application (I/O workload) in terms of type of I/O operation, size of I/O requests, access mode, etc.

The **modeling step** corresponds to the representation of the impact of different elements identified in the experimental phase into models [23]. Models of different types were proposed: (1) functional or behavioral models represent storage system management algorithms impacting performance and energy; (2) performance and energy consumption models that describe the profile of different metrics (energy consumption, response time, throughput, etc.) for a given platform; (3) I/O workload model, which is used to represent the workload applied by an application on the storage system; (4) finally, some other models specific to flash hardware components were developed: these are structural models, representing the architecture, and operational models describing the various operations supported by the flash memory. For each performance and consumption model, we associated a parameter extraction methodology and tools so that one can build the profile of a given hardware and software platform [24][25].

In the **application step**, the models were implemented in a simulator [27]. This simulator allows, on the one hand, to obtain estimations of the performance and energy consumption of flash memory-based storage systems, and on the other hand, to provide a framework to prototype new storage management layers. Thus, it can be used for prospection, validation of optimization techniques and comparative studies on performance and energy consumption. The simulator has been validated against real measures and the error rate between the estimations and the measurement remains, in most cases, smaller than 10%.

This study is detailed in **Part 3 Chapter 3**.

2.2.3.2 Flash memory based storage system integration

NAND flash memories are non-volatile EEPROM (Electrically Erasable and Programmable Read Only Memory). They have a hierarchical structure. They are composed of one or more chips. Each chip is divided into multiple planes that are composed of a fixed number of blocks, each of which encloses a fixed number of pages. Three key operations are possible on flash memories: read, write and erase. Read and write operations are performed on pages, whilst erase operations are performed on blocks. Flash Memory constraints can be summarized as follows: (1) erase before write: data can only be written on a page once it has been erased, (2) the asymmetry of the write and erase operations, (3) the limited number erase cycles that can be performed on a given memory cell, and finally (4) the asymmetry of performance between read, write and erase operations [28][29].

To overcome these peculiar issues, three main mechanisms were designed [30][31]: (1) a logical-to-physical mapping scheme is used to keep track of data updates. In fact, due to the first and the second constraints above, it is not

possible to update data in-place, (2) a garbage collection mechanism to recycle invalid pages due to out-of-place data updates, and (3) a mechanism to balance the wear over the memory cells to prevent some cells from wearing out more quickly than the others. These mechanisms are usually located in flash memory controllers (SSD, USB drive, SD card, etc.), in a layer called the FTL (Flash Translation Layer for). Moreover, cache mechanisms that take into account the specificities of flash memories have been proposed [32][33][34][40]. They allow basically two optimizations: (1) to absorb part of the data updates and thus avoid reporting them on flash memory, (2) to buffer data in order to achieve a subsequent reorganization for optimization purposes.

In the context of flash memory integration and optimization, we have proposed three flash specific mechanisms: a cache mechanism named **C-lash** (cache for flash) [35][36], and two FTL mapping schemes, namely **CACH-FTL** (Cache Aware Configurable Hybrid Flash Translation Layer) [37][38] and **MaCACH** (Maximum page-mapped region usage, Cache-Aware, and Configurable Hybrid FTL) [39].

The idea defended in C-lash is that for many I/O workload patterns, it is possible to simplify the design of a flash memory-based storage system by omitting the three services mentioned above and relying principally on a caching mechanism, namely C-lash. C-lash space is partitioned into a block space and a page space. This dual cache system represents, at the cache level, both granularities on which operations can be performed on the flash memory. C-lash is hierarchical as the eviction policy operates from page space to block space, and only then to the flash memory. This optimizes the flash memory eviction at the block space while keeping the most frequently used pages in the cache. In fact, C-lash allows to compromise between the cache effectiveness (keeping the most used data in the cache) and eviction cost (evicting sets of pages with a low impact on performance and lifetime). The conducted experiments have proved the relevance and effectiveness of C-lash for all the I/O workloads with medium to high sequential rates.

Flash memory address mapping can be performed whether in a page or a block granularity. Page mapping gives very good performance as compared to block mapping, but its downside is the huge mapping table size that is generally too large to fit in RAM [28]. Several state-of-the-art studies proposed a hybrid FTL address mapping where both granularities were used [40][41][42]. In this context, we proposed CACH-FTL, a hybrid FTL which partitions flash memory in two regions, a block and a page mapped region. It uses information from the SSD cache to decide about the data placement. When a large number of pages is evicted in a row from the cache, they are written to the block mapped region. On the other hand, if the number of pages is small, they are evicted to the page mapped region so that to reduce latencies. CACH-FTL provides a wide range of configurations. This configurability permits to tune the system offline according to the I/O workload in order to optimize the performance.

MaCACH FTL is an adaptive upgrade of CACH-FTL in which parameter setting of the FTL is updated online based on a PID feedback control (Proportional Integral Derivative). MaCACH therefore automatically adapts to the I/O workload and the state of the flash memory in order to maximize performance by using more effectively the page mapped region. Tests conducted on MaCACH have shown very good performance in terms of average response time and number of erase operations as compared to some tested state-of-the-art FTLs [34][43][44]. In addition, it offers a large configuration space.

These contributions are detailed in **Part 3 Chapter 3**.

2.2.3.3 Reconfigurable architectures and operating system service optimization

In the continuity of Research area 1 and in the context of developing tools for the integration of reconfigurable architectures in complex embedded systems, we have studied the possibility of coupling software component architecture [45][47][48], and reconfigurable hardware architectures. The software component architecture allows designing the software as a collection of interconnected components. This gives the ability for a system, once deployed, to reconsider the software structure. The reconfigurable hardware architecture helps, for its part, in accelerating software parts. The emergence of partial and dynamic reconfigurability [49][50] coupled with the computing power of current reconfigurable system, makes it relevant to consider operating system (OS) service implementations on reconfigurable FPGA circuits.

The objective of the study was to use the benefits of component software architectures to isolate OS services and interface them in a standardized way with reconfigurable circuits for flexible and scalable management of OS services.

The aim was to lay the ground for such an approach, and we selected time management service in a real-time OS (in our case μCOSII) as a case study [46].

This topic is not detailed in this document, the interested reader may consult [ci19].

2.2.4 Publications

My work on this research area began in late 2009 with a significant development of the storage system activity. The work on this area led to the supervision of two PhD students: Mr Pierre Olivier and Mr. Yahia Benmoussa.

Scientific production: 14 papers in national, international journals, or in chapters, 10 papers in international conferences, 4 papers in national conferences, and 5 workshop papers.

Journals	International conferences	National conferences and workshops
[ri3][ri5][ri6][ri7][ri8][ri9][ri10][ri11][ri13][ri14][ri15][rn1][rn2][ch2]	[ci10][ci14][ci15][ci16][ci17][ci18][ci19][ci20][ci21][ci22][ci23]	[cn2][cn3][cn4][cn5][w8][w9][w10][w11]

2.3 Research area 3: Optimization of storage systems

2.3.1 Context

The amount of data of different forms generated today is growing exponentially. In fact, it is growing faster than Moore's law (although the latter is not intended to represent this metric). For instance, online data indexed by Google increased by more than a 56 factor from 2002 to 2009 (5 to 280 Exabyte) [52]. This trend is not limited to web data, enterprise data volume is also growing very fast, it observed a cumulative growth rate of 173% [52][53].

Digital data are heterogeneous by nature and are processed and managed using different techniques and methods [51]: capturing, analyzing, processing, classifying, archiving, etc. This heterogeneity as well as large volumes are putting more and more pressure on storage subsystems and all the memory hierarchy, pushing the scientific community to think about new technologies and methods to optimize the performance of these systems.

2.3.2 Problem statement

In this data centric context, it is becoming crucial to optimize the performance of I/O storage systems. Indeed, it is the historical bottleneck because of continuing growth of the performance gap with processing units.

Flash memory-based storage systems came to reduce the performance gap between processing units and storage system [45]. Flash memory has been integrated in current computer systems in three different ways [45] [28]: (1) as an extension of the main memory, 2) as a storage system accelerator, in this case the flash memory is considered as a cache that would store frequently accessed data in order to reduce disk access, 3) as an alternative storage device, or complementary to traditional hard disk drives (HDD) storage.

Flash memories have clear advantages in terms of performance and energy consumption [54]. However, their cost and the world's high demand for storage let us think that hard drives are not expected to disappear in the mean term. It seems therefore relevant to consider hybrid (or heterogeneous) storage systems containing the two types of storage: HDDs and SSDs that can be architected in different ways.

It is in this context that I initiated my work on the research area 3. The major topic tackled in this area is **how to take profit of storage system heterogeneity for different types of applications**.

I mainly worked on two application domains in this research area:

1. Cloud computing
2. Database systems.

2.3.3 Contribution

In this part, each subsection deals with one of the issues addressed in this research area.

2.3.3.1 Integration of flash memories in the Cloud infrastructure

I started my work on the integration of flash memories in Cloud infrastructures for three different applications. Each one was subject to an industrial project.

IaaS Cloud: My work on the integration of flash memory in an infrastructure as a service Cloud (IaaS) has started within the Institute of Research and Technology (IRT) b <> com. The aim of our participation in this project was to provide the Cloud administrator (see Figure 11) with a VM placement framework for hybrid storage system performance optimization and energy efficiency. In the context of an IaaS Cloud, the customer is allocated a virtualized environment according to the terms of the contracted Service Level Agreement (SLA) with the Cloud provider. This environment consists mainly of virtualized CPU, memory, storage and network. The customer applications are run through virtual machines (VM) in the context of the IRT b<>com project.

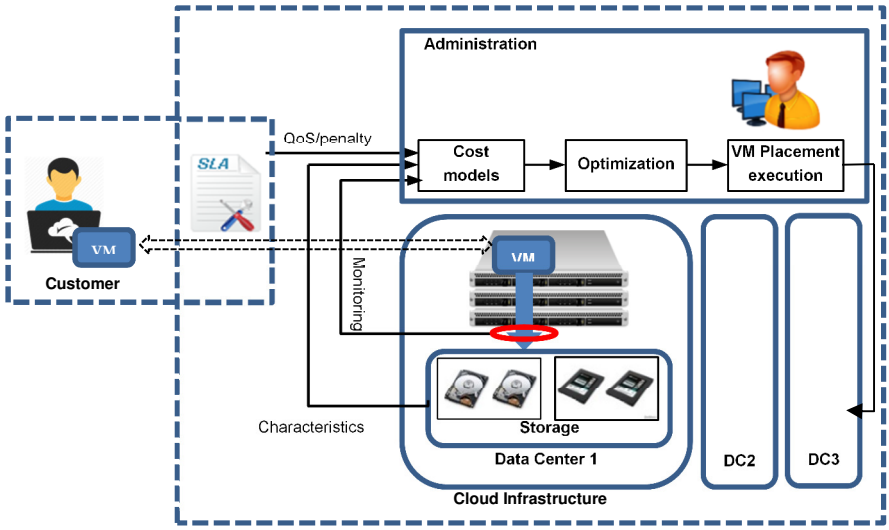


Figure 11. Considered Cloud architecture

In order to perform Cloud resource provisioning and optimize the infrastructure operation costs, Cloud providers must, on the one hand, accurately assess the cost of resource provisioning [55] and on the other hand, evaluate the compliance of the resource allocation with the customers QoS metrics contracted in the terms of the SLA [56].

The overall approach followed in the context of this project obeys the MAPE-K (Monitor, Analyze, Plan, Execute - Knowledge) autonomic loop reference model [239]. So, in our context, we used the following steps summarized in Figure 11): (1) **Monitoring** the I/O behavior and **analyzing** the VMs workloads (M and A) (2) **Planning** VMs storage by first evaluating the cost for the execution of a virtual machine (VM) on a given storage system [58][59], and then optimizing storage allocation according to administrator constraints (energy, QoS, performance, etc.), and (3) finally **Executing the data placement** plan issued in the previous step.

As for modeling the overall cost of running a VM, we took into account the characteristics of the storage system, the I/O workload execution of the VM and SLA with penalty models. The objective of the optimization step is to design a methodology for VM placement in hybrid storage systems to reduce costs for the Cloud Service Provider (CSP), based on the defined cost models.

DBaaS Cloud: My work on the integration of flash memory for a DataBase as a Service (DBaaS) Cloud is realized in collaboration with the University of Science and Technology of Algiers (USTHB). A project funding from "Partenariat Hubert Curien" (PHC) has been obtained in 2016.

The methodological approach used in this project is the same as in the previous one from the administrator's point of view (see Figure 11). The main difference lies in: (1) the used application: in fact, we no longer consider VM placement but user's database objects (e.g. a table or an index). (2) A direct consequence of managing objects rather than VMs is

the number of elements to place, one might need to use different optimization techniques. (3) The type of SLA considered will take into account the application context (Algerian specific context which is under study[9]).

Cloud and IoT: This work is done in the context of a project with Orange Labs which began in October 2015. The aim of this project is to propose a content centric approach for data management of IoT applications in a Cloud context. In the context of Orange Labs, the storage architecture can take advantage of the network architecture by controlling data location throughout the overall network architecture to optimize latencies and I/O throughput. Data location can be managed from end to end in a rather accurate way between the object and the Cloud data center. In fact, we relied on the **Fog** computing architecture, i.e. the extension of the Cloud architecture with components between the objects generating data and Cloud data center (e.g. gateways, local and regional point of presence).

Two major issues are addressed in this project: (1) the automatic data placement in a fog tiered storage system based on the data access pattern. This is performed by considering the whole storage system from end (object) to end (Cloud data center) [61]. (2) Providing mechanisms for switching between different levels of consistency within the proposed storage architecture. We assume that the read and write performance depend on the chosen type of consistency. This is an ongoing work that will not be detailed in this document.

Contributions on Cloud applications are detailed in **Part 4, Chapter 2**.

2.3.3.2 Integration of flash memory Database systems

Concerning database applications, our purpose is also to take into account the heterogeneity of storage systems to optimize I/O performance. To do this, we are working on three aspects: (1) studying and modeling the cost of query execution on flash-based storage systems [62][63], (2) revisiting the sorting algorithms in view of their optimization for flash based storage systems for high performance databases [64], and 3) studying the implications of OS I/O software stacks for high performance databases [65].

The first issue was partly tackled in the PhD thesis of Dr. Salmi (see Section 2.10.). This same issue, with issues 2 and 3 are addressed with Kode Sotfware company, this collaboration started in late 2014.

This work is detailed in **Part 4, Chapter 3**.

2.3.4 Publications

My work on this research area began in late 2012 and is still under progress.

Scientific production: 4 papers in international journals and book chapters, 10 international conference papers and 3 workshop papers.

Journals	International conferences	National conferences and workshops
[ri1][ri2][ri12][ch1]	[ci1][ci2][ci4][ci5][ci6][ci9][ci12][ci13][ci 20]	[w1][w4][w6]

[9] This will be provided by the CERIST (Centre de Recherche sur l'Information Scientifique et Technique), who is a very important Cloud providers in Algeria.

3 DOCUMENT OUTLINE

In this thesis document, I will focus on research areas 2 and 3, which gather around 6 years of activity.

The rest of the document is divided into two parts

- Part 3 is dedicated to the achieved work on research area 2
- Part 4 describes ongoing work on research area 3 and concludes on a discussion about future work.

Table 3. table showing publications related to research topics covered in research areas 2 and 3. For the other publications, please see Publication section on page 169).

	2011	2012	2013	2014	2015	2016	2017	Total
Books							1	**1**
Int. journals	1	2	1	5	2	3	1	**15**
Nat. journals			1					**1**
Chapters			2					**2**
Int. conf.	3	2	5	5	1	4	2	**22**
Nat. conf	3				2			**5**
Worksops		2	2	2	1	4		**11**
Total	***7***	***6***	***11***	***12***	***6***	***11***	***4***	***57***

Part 3: This part discusses the achieved work within the research area 2. This includes activities related to the study of energy in complex embedded systems environment; and the effective integration of flash memory-based storage systems in computer systems. Activities related to reconfigurable computing will not be detailed. This part is subdivided into three chapters:

- **Chapter 1** details our contribution on performance and energy consumption modeling and optimization of video streaming for embedded platforms. This chapter summarizes the three steps of our methodology as discussed earlier: performance and energy characterization, energy modeling, and some applications. This chapter is mainly based on the PhD thesis of Yahia Benmoussa and is inspired from the paper published in the **Elsevier Journal of System Architecture** [ri6].
- **Chapter 2** describes our work on the performance and energy consumption study of embedded storage systems based on flash memory. This chapter summarizes the three steps of our methodology as discussed earlier: the experimental exploration of performance and energy consumption of I/O storage stack, the performance and energy consumption modeling, and simulation step. This chapter is mainly based on the PhD thesis of Pierre Olivier and is inspired from the paper published in the **ACM Transactions on Embedded Systems Computing** [ri3].
- **Chapter 3** explores 3 different contributions on flash memory based storage system integration. Three of our contributions are described: **C-lash**, a cache system for flash memory storage systems, **CACH FTL**, a hybrid mapping scheme that relies on flash specific cache mechanisms, and **MaCACH FTL**, an adaptive hybrid mapping scheme based on CACH FTL. This chapter is mainly inspired from three papers [ri5][ri9][ci21], one published in the **Elsevier Journal of System Architecture**, one in **MDPI Computer Journal**, and one in **SPECTS** conference.

Part 4: This part discusses ongoing and future work. Concerning the ongoing work, I will mainly describe our research on flash memory integration in Cloud infrastructure and database systems. Both topics were carried out in a collaborative context with industrial or research institutions. Concerning future work, I will focus on challenges related to Non-Volatile Memory (NVM) integration in computer systems.

- **Chapter 1** summarizes the current state of our work on flash memory storage integration in Cloud environments. This chapter mainly focuses on two applications, IaaS and DBaaS. The objective of this work is to design a solution to optimize data placement on hybrid storage architecture to reduce the overall system cost while respecting customers SLA. This chapter is mainly based on the PhD theses of Hamza Ouarnoughi

and Djillali Boukhelef and is inspired from papers [ci2][ci6][w6], published in **IEEE/ACM CCGrid**, **Euromicro PDP** conferences, and **IEEE REACTION** workshop, respectively.

- **Chapter 2** describes the current state of our work on flash memory storage integration for database applications. Is described in this chapter some of our contributions about cost modeling in case of database management systems (DBMS) for embedded systems. I also detail our work on data sorting on flash based storage systems and finally discuss some system optimization for database performance enhancement. This chapter is partly based on the PhD thesis of Arezki Laga and is inspired from papers [ri1][ri12][ci5], published in **IEEE Trans. On Computers**, **Springer Design Automation for Embedded Systems**, and **IEEE NVMSA** conference.
- **Chapter 3** gives some perspectives for our future work. It mainly focuses on NVM integration for different applications and operating system support.

Part 3: Achieved work, Research Area 2

1 INTRODUCTION

This chapter summarizes the work conducted on Research area 2: Embedded Operating Systems, performance, and energy consumption. This work began in 2009. It mainly focuses on optimizing performance and energy consumption of embedded systems. This necessarily requires a detailed understanding and analysis of the system behavior: application, operating system, and hardware used, it also requires apprehending the interactions between these elements. In this section, I will focus on the two following issues:

1. Investigating energy consumption in complex embedded system environments: the objective of this work is mainly to analyze and understand the energy consumption of embedded systems. We chose two different case studies. The first case study we considered focuses on the applicative level. Indeed, we investigated one of the most used applications in the context of mobile computing which is **video streaming**. In fact, video streaming is one of the most compute intensive and thus energy hungry applications. The second considered case study with a similar approach is centered on a specific service of the Operating System (OS), that is the **embedded storage system management**.

   Both studies were the subject of two PhD theses already defended that of Dr. Yahia Benmoussa and the one of Dr. Pierre Olivier. The first PhD work was achieved in joint collaboration with the University of Mhamed Bougara in Boumerdes, Algeria. The second one was funded by the MESR (Ministry of higher education and research). Both PhD thesis have benefited from the support of the ANR project Open-PEOPLE and were co-supervised by Dr. Eric Senn from Lab-STICC in Lorient.

   Both studies were conducted with a similar methodological approach. This methodology is based on three steps: (1) an **experimental step** for the study of the performance and energy consumption, (2) a **modeling step**, and (3) an **application step** in which the elaborated models were used whether for the purpose of optimization of certain performance metrics, or for developing simulation tools.

2. The Effective integration of flash memory-based storage in specific mechanisms: a cache mechanism named **C-lash** (cache for flash) [ci21], and two FTL mapping scheme mechanisms, namely **CACH-FTL** (Cache Aware Configurable Hybrid Flash Translation Layer) [ci17] [ri9] and **MaCACH** (*Maximum page-mapped region usage, Cache-Aware, and Configurable Hybrid FTL*) [ri5].

This part is divided into three chapters:

- **Chapter 1** details our contribution on performance and energy consumption modeling and optimization of video streaming for embedded platforms.
- **Chapter 2** describes our work on performance and energy consumption study of embedded storage systems based on flash memory.
- **Chapter 3** explores three different contributions on flash memory based storage system integration.

2 PERFORMANCE AND ENERGY CONSUMPTION OF VIDEO DECODING ON HETEROGENEOUS SOC

2.1 Summary

This chapter summarizes our work on performance and energy consumption modeling and optimization of video streaming for embedded platforms. It describes the three steps of our methodology as discussed earlier: performance and energy characterization of video streaming on two processor types, GPP and DSP, energy modeling of video streaming throughout a comprehensive semi empirical modeling methodology, and some applications related to the use of our models. This section is mainly based on the PhD thesis of Mr. Yahia Benmoussa.

2.2 Context

Mobile devices such as smartphones and tablets are more and more used in everyday life. One of the most popular applications running on these devices is video playback. This is due to the growing use of video-sharing platforms (e.g. YouTube, Netflix, Dailymotion), social networks (e.g. Facebook, Twitter), mobile IPTV, video-conferencing, etc. According to a recent (2013) study achieved on 200 million mobile users [15], the average video watching time is 52 min per day. In addition, it is expected that video data will represent 70% of the overall Internet mobile traffic in the next few years [14].

These new trends in video application use combined with the market explosion of multimedia consumer electronics raise new challenges for mobile device architecture designers. Indeed, to fit the important processing requirement and real-time constraints of video applications, processing resources embedded in these devices tend to be more and more powerful and complex. One important issue resulting from these new trends in hardware architecture is a drastic increase in the power consumption of these devices.

In fact, according to [68][69], when playing a video, the processing resources are responsible of more than 60% of the power consumption. This leads to a dramatic decrease in mobile devices autonomy as lithium battery technologies are not evolving fast enough to absorb the ever-growing energy requirements of such mobile architectures [13].

2.3 Problem statement

For the abovementionned reasons, energy saving considerations became at the center of modern microprocessors design. Although tremendous advances were achieved in this field, the energy optimization efforts are still insufficient. In effect, due to the limitation of the microprocessor fabrication technologies, it is expected that only 20% of the energy saving will be achieved at this level in the next few years [11]. Thus, one should consider the overall system including both the hardware and the software platforms at multiple levels in order to cope with the energy saving issue. However, to take full advantage of these multi-level energy saving opportunities, mobile system designers should deal with the increasing system complexity and heterogeneity.

In the case of mobile video applications, heterogeneous processing resources, such as GPP and DSP (among others), are at the heart of the video decoding process. They interact with memory and I/O system to execute software components such as the operating system and video codecs. Understanding the interactions between all these elements is necessary when considering the overall energy consumption balance in order to design energy-efficient techniques. Low-level estimation of the impact of all these parameters is time consuming and very difficult in a context of increasingly complex hardware and applications. On the other hand, high-level methodologies are easier to develop but still hit a wall when it comes to provide users with comprehensive models describing the energy consumption behavior.

In our opinion, **one should find a middle ground and achieve a balance between an abstract high level model and a detailed complex low level one**. This can be achieved by bridging the gap between a comprehensive performance and energy characterization methodology based on exhaustive measures and a high-level modeling methodology.

2.4 Approach

In this perspective, we proposed **an end-to-end methodology to characterize and model the energy consumption of processing resources in the context of video decoding for embedded heterogeneous platforms** containing both a GPP and a DSP.

We aim to model the energy consumption of video decoding executed on complex embedded systems including embedded operating systems and heterogeneous processing elements. Accordingly, we proposed a high level modeling approach considering a set of parameters at the **application**, **operating system,** and **architectural** levels.

- At the **application level**, we considered the impact of the video quality (bit-rate and resolution) and the scene complexity on the energy consumption of video decoding. The considered video codec is H.264/AVC, a widely used compression video standard.
- At the **operating system level**, we focused on studying the impact on the energy efficiency of the processor clock frequency and the inter-processor communication mechanism implemented by the OS for scheduling video decoding task on heterogeneous processors.
- At the **architectural level**, we studied the impact of different processing configuration available on heterogeneous SoC on energy consumption. We particularly focused on modeling the energy consumption of video decoding on two widely used architectures: GPP and DSP. We explored also the energy efficiency of video decoding using parallel core and hardware accelerated codecs.

To model the energy consumption at the considered levels, we chose to use a high level methodology based on extensive experimental power measurements achieved on real embedded platforms. This is motivated by our desire to build fast energy models which represent real life scenarios.

High level models may be hard to generalize to other architecture since they do not consider low level details. However, we believe that one can find the **middle ground and achieve a balance between the advantages of high level empirical approaches and low level based ones**. In our opinion, we could make the experimental based energy models more portable using a deep characterization methodology at all the considered levels to map the developed model with comprehensive relevant parameters.

The work presented in this section was extracted from Yahia Benmoussa's PhD thesis and the following publications: [ri6][ri7][ri10][ci10][ci14][ci15][w8][w10].

2.5 Background

We describe hereafter some elementary background related to power consumption in electronic circuits. Then, we discuss how the performance and energy consumption of video decoding applications depend on architectural, operating system and applicative levels in a video decoding system.

In CMOS digital circuits, the total power consumption is the sum of the static and dynamic powers:

$$P_{tot} = P_{static} + P_{dyn} \quad (1)$$

Where $\mathbf{P_{static}}$ and $\mathbf{P_{dyn}}$ are defined as:

$$P_{static} = I_{leak} \cdot V \quad (2)$$

$$P_{dyn} = C_{eff} \cdot V^2 \cdot f \quad (3)$$

$\mathbf{I_{leak}}$ is the leakage current, **V** is the supply voltage associated to the clock frequency **f** and $\mathbf{C_{eff}}$ is the circuit effective capacitance [70].

The static power is related to the circuit fabrication technology and does not depend on its activity. Below 65-nm circuits feature size, it becomes significant and poses new low-power design challenges [71]. On the other hand, the dynamic power is related to the circuit activity. For example, in case of a microprocessor, the dynamic power depends on the type of instruction executed and on the data accessed [72]. In Eq. (1), this is represented by the $\mathbf{C_{eff}}$ parameter defined as C_{eff} = A.C, where **C** is the circuit capacitance and **A** the switching probability representing the activity factor.

Since the dynamic power is based on the circuit activity, it highly depends on the system and application layers. This can be illustrated by the following example dealing with the particular case of video decoding application.

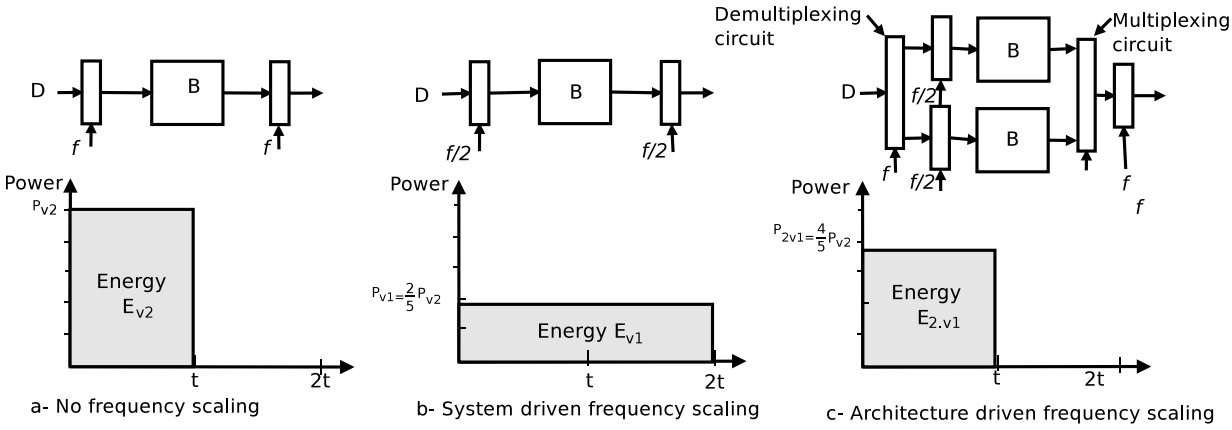


Figure 12. System and architecture driven frequency scaling example.

Figure 12 illustrates a simplified representation of a CMOS circuit which processes a set of sequential data **D** (encoded video frames) using a block **B** (video decoder). The block **B** operates at frequencies **f/2** and **f,** corresponding to the supply voltage levelsV_1=1.06 V and V_2=1.2 V respectively[10]. If **t** is the processing time when **B** operates at a frequency **f** (Figure 12-a), then the energy consumption is $E_{V_2} = P_{V_2}.t$ where $P_{V_2} = C_{eff}.V_2^2$. If we suppose the processing time at frequency f/2 (Figure 12-b) is doubled, then the ratio between the energy E_{V_1} consumed by the circuit at the frequency f/2 with V_1=1.06 V, and E_{V_2} is:

$$\frac{E_{V_1}}{E_{V_2}} = \frac{C_{eff}.V_1^2.{}^{f}/_{2}.2.t}{C_{eff}.V_2^2.f.t} = \frac{V_1^2}{V_2^2} \cong \frac{4}{5}$$

In this case, scaling down the voltage and the frequency decreases the power consumption to $P_{V_1} = C_{eff}.V_1^2.{}^{f}/_{2} = \frac{2}{5}.P_{V_2}$ which leads to 20% energy saving at the cost of a decreased performance. This may represent a scenario where the operating system scales down dynamically the processor frequency at runtime when it detects a load decrease. This illustrates a system-driven voltage scaling.

In order to save energy without sacrificing performance, an architectural-driven voltage scaling [73] can be achieved by using two **B** blocks which are both clocked at a frequency **f/2** and supplied with a voltage $\mathbf{V_1}$ as described in Figure 12-c. P_{2V_1} and E_{2V_1} refer to the power and the energy consumption associated to this configuration. Since the two blocks are operating in parallel, the execution time does not decrease and the ratio between E_{2V_1} and E_{V_2} is:

$$\frac{E_{2V_1}}{E_{V_2}} = \frac{C_{eff}.V_1^2.{}^{f}/_{2}.t + C_{eff}.V_1^2.{}^{f}/_{2}.t}{C_{eff}.V_2^2.f.t} = \frac{V_1^2}{V_2^2} \cong \frac{4}{5}$$

In this configuration, the total power consumption P_{2V_1} is the sum of the power consumptions of the two blocks, which is equal to $2.C_{eff}.V_1^2.{}^{f}/_{2} = \frac{4}{5}.P_{V_2}$. The energy saving is equal to 20% without sacrificing the performance but at the cost of an additional circuit area and thus additional static power.

Theoretically, this type of parallelism provides better performance and energy efficiency for multimedia data processing applications [73]. However, it generates an energy overhead in both architectural and system levels. For the architectural level, the use of additional processing and multiplexing/de-multiplexing circuits consumes more static power [74][71]. On the other hand, at the system level, when this type of parallelism is implemented in external

[10] V_1 and V_2 are associated to the frequencies 500 MHz and 250 MHz of the Cortex A8 processor used in our experiments.

specialized processors such as DSPs, the GPP/DSP inter-processor communication may generate a substantial system overhead [75], and thus, it may induce more energy consumption.

As an illustration of such system overheads, Figure 13 describes the steps of a typical DSP video decoding process controlled by a GPP for which the video frames are located in an input buffer in the memory:

1. The GPP writes the frame from its cache to a shared memory to be accessed by the DSP.
2. The GPP sends the frame parameters (frame location in the memory) to the DSP codec via a GPP/DSP hardware bus.
3. The DSP cache invalidates the entries in its cache corresponding to a frame buffer in the shared memory.
4. The DSP decodes and transfers the frame to the output buffer.
5. The DSP sends the return status to the GPP.

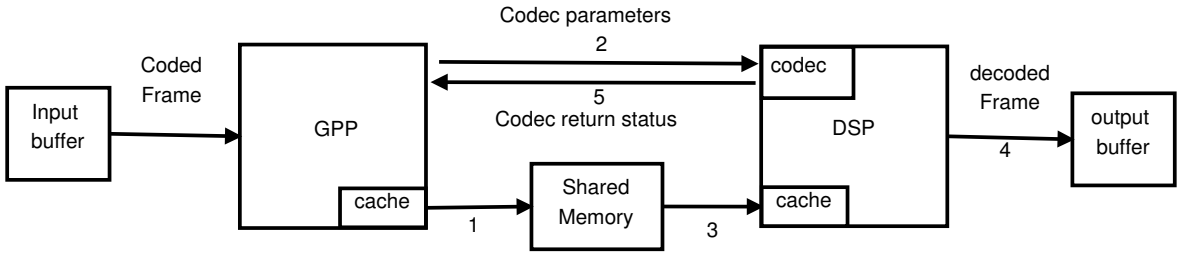


Figure 13. DSP video decoding

The fact that both the DSP and the GPP have their own cache memory and communicate using a shared memory imposes the cache to manage the coherency each time data are shared between the DSP and the GPP. Additionally, from the operating system point of view, the GPP/DSP communication is managed by a driver through a system call. A frame decoding is considered as an I/O operation generating a system latency caused by entering the idle state and handling the hardware interrupt. In addition to these system implications, the energy consumption may rely on the application level such as the video decoder design and video properties. For example, if the decoder is multi-threaded, it induces additional synchronization processing and energy consumption. On the other hand, the video bitrate, resolution and complexity have an impact on the amount of buffer memory transfers, the decoding time and the energy consumption [76].

The energy consumption of video decoding depends, then, on a combination of fabrication technology, architecture, system and application parameters. Understanding and estimating the impact of these parameters on the energy consumption of the video decoding on a heterogeneous SoCs is the main objective of this study.

2.6 Related work

We present some work related to the investigated topic in two categories: first, we describe some studies on energy consumption at the architectural level, then we present different methodologies proposed for energy consumption estimation and modeling in general and in the case of video decoding in particular.

2.6.1 Energy consumption of video decoding according to processor architecture

For the architectural level, the advantages in terms of performance and energy consumption of H.264/AVC video decoding using ASICs (Application Specific Integrated Circuit) are highlighted in [77]. In the same way, a more general study [78] investigated the reasons behind energy inefficiency of GPPs and proposed guidelines to reduce the energy break-down as compared to ASICs. However, video standards evolve quickly and ASICs do not provide flexibility to adapt to those changes [79]. For example, hardware accelerators for the new MPEG HEVC (High Efficiency Video Coding) standards were not available on mobile devices by 2015.

DSP-based solutions aim to conciliate the flexibility of GPPs and the energy efficiency of ASICs. In [73][74], the authors focus on the performance and energy efficiency of DSPs due to the use of pipelines and parallelism. The benefit of using them in energy constrained mobile devices was highlighted in [80], especially for video decoding [81].

Although there were some major advances achieved in the architecture of GPPs and DSPs, their performance tends to stall due to frequency and power wall limitations [82]. To overcome these barriers, modern architectures use SoCs

integrating more and more heterogeneous processor cores. For example, the big.LITTLE ARM architecture contains four Cortex A7 and four Cortex A15 processors [83]. Several studies investigated the impact of task scheduling on multicore architecture for energy efficiency [85][84]. In case of video decoding, the authors of [86] proposed a slice-based parallel H.264/AVC decoding on multi-core processor. The frequency of each core is set according to the expected slice workload. However, this requires a fine grained workload complexity model to adjust the core frequency accordingly.

In [87], the authors propose a task level scheduling strategy for multimedia application running on multi-core processors. Their idea consisted in assigning the lowest frequency to all the cores. Then, the tasks are load-balanced on the cores. If all the cores were at full load, the frequency of one of them was increased so that it can run more tasks. The experimentation tests considered a set of heterogeneous workloads inducing video decoding and encoding and audio decoding.

In our study, we analyzed the energy efficiency at a task level (decoding of a short video sequence) on a heterogeneous SoC including GPP and DSP processors. Furthermore, we took into consideration the video quality parameter and propose guidelines for energy aware scheduling of video decoding on GPP and DSP in case of adaptive video decoding according to experimental results.

2.6.2 Energy consumption modeling and estimation

To evaluate the energy consumption of embedded systems including complex microprocessors, operating systems and applications, two major approaches are used: **low level behavioral simulation** and **high level empirical modeling** based on measurements.

Low level simulators make it possible to estimate the energy consumption of a running application without using physical measurement tools. For example, Wattch [88], based on SimplerScalar processor simulator framework [89], uses a suite of parameterizable power models for different hardware structures. It is based on a per-cycle resource usage count generated through cycle accurate simulations. In the same way and more recently, the McPAT simulator [90] is proposed to estimate the energy consumption of multi-core architectures. These simulators allow hardware architects to explore the energy efficiency of processor architectures early by testing the impact of different hardware configurations. However, they are hard to build and require a very deep knowledge of the target microprocessor micro-architecture [91].

On the other hand, **high level empirical modeling** approaches help in building energy models based on physical power measurements. They can be implemented rapidly on any available platform using power measurement tools. This is achieved to focus the effort on building high-level models according to system and application parameters. In [92], H.264/AVC decoding performance is characterized on different GPP architectures at cycle-level. Based on this approach, a cycle-accurate modeling of the energy consumption of different H.264/AVC decoder phases is proposed in [93]. The obtained model was used to develop an energy-aware video decoding policy for ARM processor supporting the Dynamic Voltage and Frequency Scaling (DVFS) feature. In [94], the authors proposed an ARM processor energy model considering the variation of the video bit-rate. The obtained model was used to develop an energy-aware video decoder for scalable video coding (H.264/SVC)[95]. However, in this proposed model, the authors did not consider a realistic memory system by assuming a linear and uniform scaling of the decoding performance when varying the processor frequency which is not always true due to the memory off-chip access latency.

The impact of off-chip memory access on performance scalability using DVFS was addressed in some other studies. For example, in [96][97], the authors propose an online performance model for video decoding. First, they observed that, in MPEG video decoding applications, the execution time of the memory bound instructions tend to be constant from a frame to another. They propose to separate a frame decoding time into two parts: a frame-dependent part and a frame-independent part. The frame-independent part remains constant regardless of the frame type and tends to correspond to the memory-bound instructions. On the other hand, the frame-dependent part corresponds to the CPU bound instruction. The amount of memory-bound executed instruction is calculated based on the number of L2 cache miss information provided by a processor event counter. The execution time of the CPU-bound instruction is estimated using a moving average filter. The combination of both information is used to estimate the frame decoding time and to adjust the processor frequency accordingly. Such an approach relies on the existence of some hardware

counters in the execution platform. Besides, they are based on online monitored information and the proposed models do not consider the impact of video quality on performance scaling.

High level empirical models obtained using measured energy data can provide good estimation precision but are hard to apprehend. Even if this type of models is necessary for performance comparison, it fails to describe the relation between the energy consumption behavior and the architecture or video related parameters.

In our study presented hereafter, we have proposed a modeling methodology to build a comprehensive energy model achieving a balance between an abstract high level model and a lower level one.

2.7 Contribution

2.7.1 Experimental characterization and modeling methodology description

In this study, we consider a video decoding process on a given energy-constrained mobile device. The hardware architecture of this mobile device contains a GPP and a DSP within a SoC. Each processor supports different clock frequencies **f**. A video sequence **v** consists of a set of video frames. It is characterized by a displaying-rate **d** (expressed in Frames/s), a bit-rate **r** (expressed in Kb/s), a spatial resolution **s** (size of a frame in term of number of pixels) and a complexity **c**, which reflects temporal and spatial complexity characteristics of the video.

We define a video decoding configuration as a set of constant and variable parameters. The constant parameters are related to the processor architecture, the video resolution and the complexity. For a given video sequence and a mobile device, these parameters are not supposed to change. On the other hand, the variable parameters are the video bit-rate **r** and the processor clock frequency **f**. The bit-rate may vary depending on the network bandwidth capabilities and the processor frequency is driven by system frequency scaling policy.

In this context, we focus on characterization and modeling of energy consumption of processing resources when decoding a video as a function of the above-cited constant and variable parameters. The objective of the characterization part is to evaluate and explore energy consumption variation. It consists in executing and measuring the energy consumption of different video decoding configurations. The modeling part, which depends on the measurement results obtained from the characterization phase, aims at estimating energy consumption. The proposed methodology, described in details in the next subsections, is thus an end-to-end approach to evaluate, analyze and estimate the energy consumption of video decoding on a SoC.

As shown in Figure 14 our methodology is based on (1) an **experimental** methodology for characterization and (2) a **modeling** methodology. This methodology is described in the rest of the section. The **application** step will be discussed later in this chapter.

2.7.1.1 Experimental methodology for characterization

In the process of energy characterization of video decoding, we focus on the processing elements. Phases which are not part of the actual GPP and/or DSP video decoding were discarded.

As described in Figure 14 (left side), the characterization is divided into 4 steps: (1) video complexity characterization, (2) operating system level characterization, (3) video frame level characterization, and (4) video sequence level characterization. The first step is achieved at the video encoding phase while the remaining ones are executed during video decoding (on an embedded hardware platform).

1. **Video complexity**

This step is executed in the video encoding phase to prepare the tested video sequences for our experimentations. We used H.264/AVC, a mature and widely adopted standard [99].

A set of representative raw video sequences (with different video complexities and resolutions) were selected and encoded with different constant bit-rates using the H.264/AVC encoder. To extract the video complexity information, we kept trace of the mapping between each bit-rate and the average quantization parameters $\mathbf{qp_{avg}}$ used by the encoder to encode the videos. The quantization parameter, which is defined in the H.264/AVC standards, is adjusted dynamically by the encoder for each video frame or macro-block to fit a target bit-rate. The more a video is complex,

the higher is the value of the used quantization parameters. Therefore, the information about the video complexity can be provided by the couple **c=(r, qp_{avg}).**

As illustrated in Figure 14, the output of this step (block (1)) served to build a rate model which describes the variation of the bit-rate as a function of **qp_{avg}**. This model helps to build an energy analytical model as a function of the bit-rate and some video complexity related parameters. The next characterization steps are those dealing with the actual decoding process executed on an embedded platform.

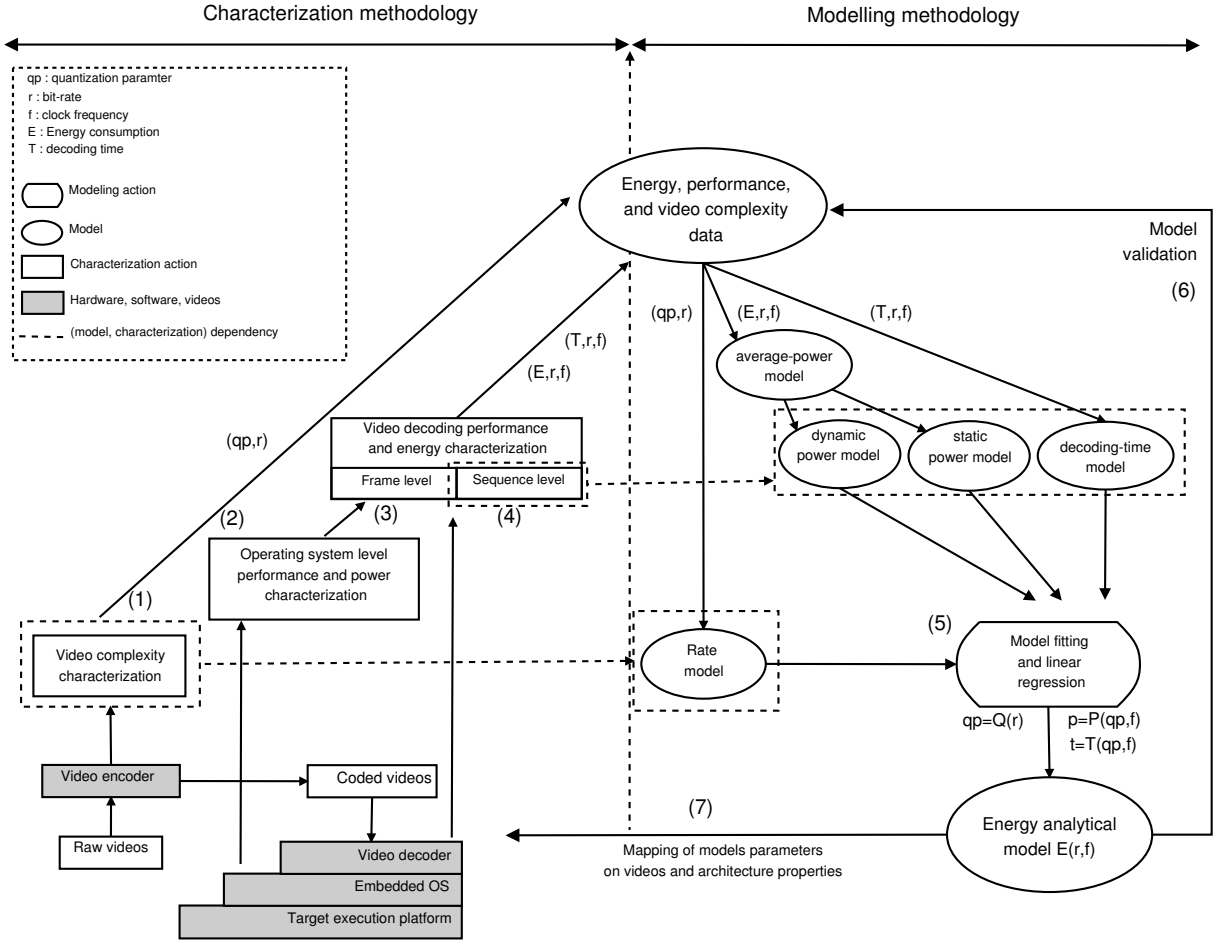


Figure 14. Characterization and Modeling methodology description

2. Operating system level

In this step of the methodology, the power consumption of both GPP and DSP in idle and active states are measured for different clock frequencies regardless of any video decoding process. The objective of this level is to extract a set of reference power consumption values that help to understand and quantify the performance and the energy consumption of the different video decoding process phases at a frame granularity.

3. Video frame level

This level relies on the preceding one. Its objective is to understand how the processing elements are used and where goes the energy when decoding a single frame in order to explain the global performance and energy consumption for both GPP and DSP decoding. For this purpose, elementary video frame decoding is characterized in terms of system metrics such as the amount of buffer transfers, GPP/DSP communication latency and cache coherency maintenance.

4. Video sequence level

In this step, the average performance (in terms of number of decoded frames per second) of H.264/AVC decoding of the overall video sequence is evaluated depending on the video bit-rate and resolution, and the processor clock frequency. Both GPP and DSP were tested. The considered performance evaluation metric is the video displaying rate

of the decoded video. In fact we considered that a decoding rate which is lower than the displaying rate of the coded video is not sufficient with respect to real-time constraints.

The overall energy consumption is then calculated by multiplying the sum of the elementary measured power values by the decoding time. The average energy per frame (mJ/frame) is inferred by dividing the overall energy by the total frame number.

Each video was decoded using GPP and DSP processors. This decoding was repeated for all available clock frequencies. For each bit-rate, resolution, and clock frequency, the decoding time and the energy consumption were measured. As shown in Figure 14 (block 4), the result of this phase are triplet data **(T, r, f)** and **(E, r, f)** describing the decoding time (**T**) and energy consumption (**E**) in terms of **r** and **f**. These data are used in the modeling phase, described hereafter, to build a power and a performance model for video decoding

2.7.1.2 Modeling methodology

In our modeling methodology, we used a top-down model decomposition approach in which the energy model is decomposed as a function of two sub-models: a decoding-time model **T** and an average power model $\mathbf{P_{avg}}$.

$$\boldsymbol{E(r,f) = P_{avg}(r,f) . T(r,f)} \quad \textbf{(4)}$$

$\mathbf{P_{avg}}$ is then decomposed into a dynamic $\mathbf{P_{avgdyn}}$ and static $\mathbf{P_{avgstat}}$ power models:

$$\boldsymbol{P_{avg}(r,f) = P_{avgdyn}(r,f) + P_{avgstat}(r,f)} \quad \textbf{(5)}$$

We propose to use the mapping between the bit-rate **r** and the average quantization parameter $\mathbf{qp_{avg}}$ to add the video complexity information to our model. For this purpose, we first use the video $\mathbf{qp_{avg}}$ instead of the bit-rate **r** as a model parameter for the power and performance models. Therefore, the model described in Eq. (5) giving the energy in terms of the clock frequency **f** and the bit-rate **r** becomes:

$$\boldsymbol{P_{avg}(qp_{avg}, f) = P_{avgdyn}(qp_{avg}, f) + P_{avgstat}(qp_{avg}, f)} \quad \textbf{(6)}$$

In order to obtain an energy model as a function of the clock frequency and the bit-rate, we used a rate model **Q** which describes $\mathbf{qp_{avg}}$ in terms of the bit-rate **r** and some video complexity related coefficients.

$$\boldsymbol{E(r,f) = (P_{avgdyn}(Q(r),f) + P_{avgstat}(Q(r),f)) . T(Q(r),f)} \quad \textbf{(7)}$$

The energy model described in Eq. (4) is thus decomposed into: (1) a rate model, (2) a static/dynamic power model and (3) a time model (see Figure 14). Based on these sub models, we can start the energy model construction phase. A model fitting is performed on the video complexity characterization results to develop a rate model. Then, a regression analysis of the video sequence level characterization results is used to develop the power and the decoding time analytical sub-models. Details on sub-models, experimental measurement and development of the sub-models are described below.

Video rate sub-model

To express the video bit-rate as a function of the quantization parameters, we used the model proposed in [76]. The authors of this study assume that the video bit-rate can be described as follows:

$$\boldsymbol{r = \left(\frac{q}{q_{\min}}\right)^{-a} . r_{max}} \quad \textbf{(8)}$$

Were **q** is the step size for the quantization which is defined as follows [99]:

$$\boldsymbol{q = 2^{(qp_{avg}-4)/6}} \quad \textbf{(9)}$$

a is an exponent which represents how fast the video rate changes in terms of the step size parameter. $\mathbf{q_{min}}$ is the lowest step size parameter used to encode the video with the highest bit-rate $\mathbf{r_{max}}$.

By substituting Eq. (9) in (8) we obtain the following rate model:

$$\boldsymbol{qp_{avg} = 4 + 6 . ln_2(q_{min} . (\frac{r}{r_{max}})^{-1/a})} \quad \textbf{(10)}$$

Using the values ($r;qp_{avg}$) returned by the encoder, the parameters **qmin**, $\mathbf{r_{max}}$ and **a** can be calculated using model fitting for each coded video.

Power sub-model

The static power corresponding to each voltage level is modeled using the processor data-sheet [100]. To model the dynamic power consumption, $\mathbf{C_{eff}}$ parameter value is obtained by fitting the measured dynamic power (i.e. the measured total power minus the measured static power) with the model described in Eq. (3).

Decoding-time sub-model

To develop the video decoding time model **T**, we used an observation on the experimental results which reveals a linear relation between **1/t** (where **t** is the decoding time) and both clock frequency **f** and quantization parameter $\mathbf{qp_{avg}}$. This linear relation was validated using a multi-linear regression of **1/t** in terms of **f** and $\mathbf{qp_{avg}}$.

2.7.2 Experimental setup

The above discussed methodology is independent from the underlying hardware and software platforms. In this section, we describe the application of this methodology on an OMAP hardware platform on which is ran a Linux kernel and GStreamer multimedia framework.

2.7.2.1 Hardware setup

The power measurements were conducted on the OMAP3530EVM board containing the low-power OMAP 3530 SoC. This SoC is based on 65-nm technology and consists of a Cortex A8 ARM processor supporting ARMv7 instruction set and a TMS320C64x DSP. The OMAP3530 supports six operating frequencies ranging from 125 MHz to 720 MHz for the ARM and from 90 MHz to 520 MHz for the DSP. The power consumptions of the DSP and the ARM processors were measured using the Open-PEOPLE framework [98], a multi-user and multi-target power and energy estimation and optimization platform.

The OMAP3530EVM board provides a single jumper for measuring both of the DSP and the ARM processor power consumptions. In case of ARM video decoding, the measured power represents the ARM dynamic consumption plus the ARM and DSP static power. In case of DSP video decoding, both ARM and DSP are involved. In fact, the ARM controls the DSP which executes the actual video decoding process. The measured power is thus the sum of the static and the dynamic power of both ARM and DSP. In what follows, $\mathbf{P_{static}}$, is the sum of ARM and DSP static power. ARM and DSP dynamic powers are noted $P_{dyn_{arm}}$ and $P_{dyn_{dsp}}$, respectively. The total power consumption of the ARM and the DSP is noted P_{tot}.

2.7.3 Software setup

On this hardware platform, Linux kernel version 2.6.32 was used with cpufreq [101] enabled to drive the ARM and the DSP frequency scaling. The DSP was running the DSPBios operating system and was driven from the Linux/GPP side using a specific driver.

On this system, H.264/AVC video decoding was achieved using GStreamer [102], a multimedia development framework. GStreamer allows an accurate GPP/DSP decoding comparison thanks to its modular design allowing to plug and execute both GPP and DSP video decoders into the same software environment.

The used hardware and software configurations are summarized in Table 4.

For video decoding characterization we encoded 3 well-known raw video sequences: City, Soccer and Harbor in H.264/AVC base-line profile. These sequences represent respectively a low, a medium and a high video complexity. Each video is available in three resolutions: qcif (176x144), cif (352x288) and 4cif (704x576). Each video was encoded into 13 different bit-rates (from 64 Kb/s to 5120 Kb/s) using x264encoder [103]. After each coding operation, the values of **qp** used in the encoding process were extracted from the encoder logs and the average quantization parameter $\mathbf{qp_{avg}}$ was then calculated accordingly. This step served to characterize the video complexity and to build a linear performance model as described.

Table 4. Hardware and Software configurations

Applications	GStreamer	ARM plug-in	ffdec_h264
		DSP plug-in	TIViddec
	Operating System	GPP	Linux 2.6.32
		ARM	DSPBios
		ARM/DSP DVFS driver	cpufreq
		DSP driver	DSPLink
		DSP power management driver	LPM
MistralEVA3530	DSP	Model	TMS320C64x
		Frequencies (MHz)	90, 180, 360, 400, 430, 520
	ARM	Model	Cortex A8 + NEON
		Frequencies (MHz)	125, 250, 500, 550, 600, 720
	SoC	Model	OMAP3530
		Voltages levels	0.975, 1.05, 1.2, 1.27 , 1.35, 1.35

2.7.4 Experimental results of video decoding characterization

In this section, we discuss results obtained after the execution of the characterization methodology discussed above.

2.7.4.1 Video complexity

Figure 15 shows the mapping between resolution, bit-rate, and quantization parameters extracted for the considered sequences. The plotted data show that for the same resolution and bit-rate, the average quantization parameter $\mathbf{qp_{avg}}$ of the City, Soccer and Harbor are in an increasing order. This is explained by the fact that the Harbor video sequence is more complex than Soccer which is, in turn, more complex than City video.

	Average quantization parameters qp_{avg}								
	City			Soccer			Harbor		
Bit-rate(Kb/s)	qcif	cif	4cif	qcif	cif	4cif	qcif	cif	4cif
64	32.81	42.74	51.00	38,22	48,81	51	37,78	45,84	51
128	27.69	36.79	49.38	31,65	41,55	50,99	33,94	41,56	50,7
256	23.06	31.83	41.52	25,57	35,43	45,27	30,07	37,78	45,5
512	18.49	27.28	35.76	19,84	29,76	39,09	25,83	34,12	41,26
1024	13.51	23.07	31.27	14,22	24,38	33,75	20,66	30,21	37,48
1536	10.43	20.71	29.14	10,89	21,47	30,87	16,84	27,64	35,18
2048	7.97	18.92	27.83	8,25	19,35	29,01	13,72	25,65	33,53
2560	5.69	17.50	26.86	5,87	17,77	27,61	11,08	23,98	32,21
3072	3.51	16.33	26.09	3,59	16,49	26,5	8,75	22,55	31,11
3584	1.31	15.34	25.45	1,8	15,41	25,6	6,51	21,27	30,15
4096	0.44	14.48	24.89	1,05	14,45	24,85	4,37	20,11	29,31
4608	0.22	13.70	24.40	0,91	13,6	24,17	2,31	19,03	28,56
5120	0.14	12.99	23.94	0,82	12,83	23,57	0,95	18,01	27,88

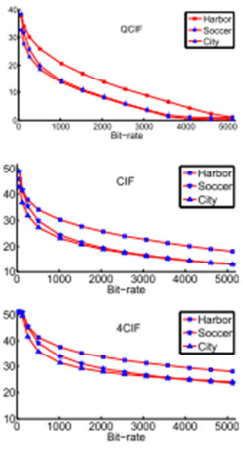


Figure 15. Mapping between video $\mathbf{qp_{avg}}$ and bit-rates.

2.7.4.2 Operating system level

Table 5 summarizes the measured power characteristics of the Cortex A8 processor and the TMS320C64x DSP. The $P_{act_{arm}}$ and $P_{idle_{arm}}$ represent the active and idle power consumption of the ARM processor corresponding to the six available clock frequencies. The $P_{idle_{dsp}}$ shows the DSP idle power consumption at different frequency levels. The idle state corresponds to the state where the DSP is activated without executing any instruction. The values of the static power corresponding to each voltage level are taken from [100].

The power consumption levels measured in this step provide information on how the energy is consumed in different processor states. The amount of time spent in each state is one of the parameters which impact the overall energy consumption. This is discussed when analyzing video decoding at a frame level in the following section.

Table 5. OMAP3530 power consumption levels

V_{dd}	f_{arm}	f_{dsp}	$P_{act_{arm}}$	$P_{idle_{arm}}$	$P_{idle_{dsp}}$	P_{static}
V	MHz		W			
1.35	720	520	0.5965	0.4342	0.2312	0.01975
1.35	600	430	0.4997	0.3801	0.1778	0.01975
1.27	550	400	0.4087	0.3089	0.1490	0.01527
1.2	500	360	0.3276	0.2476	0.1217	0.01135
1.05	250	180	0.1238	0.0913	0.0498	0.00716
0.975	125	90	0.0421	0.0275	0.0224	0.00516

2.7.4.3 Video-frame level

The objective of this step is to understand where goes the consumed energy at a microscopic level with a frame granularity. The objective was to observe the different energy phases (memory vs CPU vs communication such as in [104]). We restricted our experimentations on 10 frames extracted from the Harbor sequence. These frames are coded in 4cif (4 Mb/s), cif (1 Mb/s) and qcif (128 Kb/s) resolutions. We used a 720 MHz clock frequency for both ARM and DSP processors.

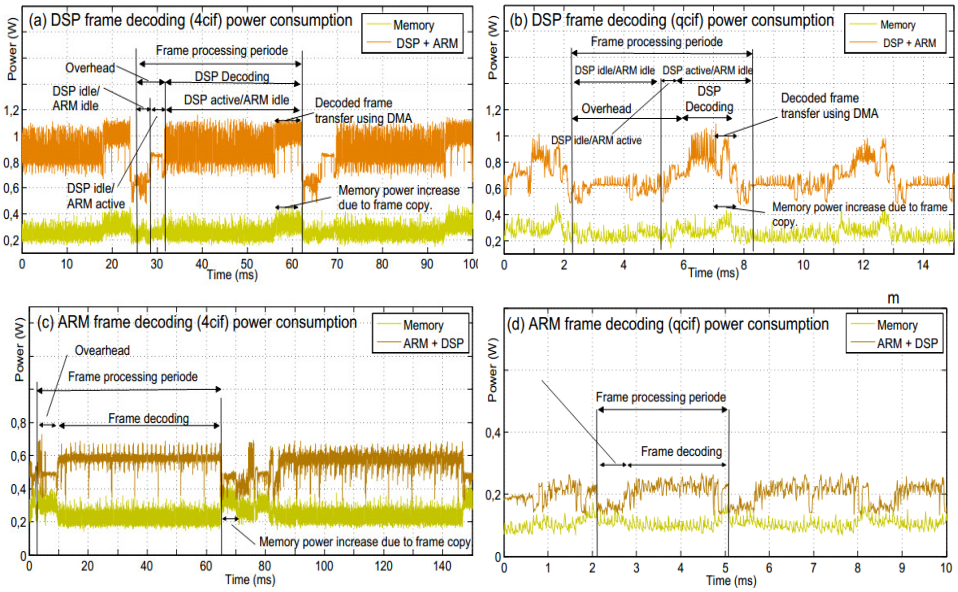


Figure 16. ARM and DSP frames decoding

Figure 16-a and Figure 16-b show the power consumption levels of 4cif and qcif DSP video decoding. The DSP frame decoding phase is represented by the strip varying between 0.7 W and 1.1 W corresponding to [32 ms, 62 ms] (see Figure 16-a) and [6.2 ms, 7.5 ms] (see Figure 16-b) intervals. This phase is terminated by a burst of DMA transfers of the decoded frame macro-blocks from the DSP cache to the shared memory. This phase corresponds to the intervals [56 ms, 62 ms] (see Figure 16-a) and [7.2 ms, 7.5 ms] (see Figure 16-b) and is illustrated by an increase in memory power consumption. When the DSP terminates the frame decoding, it returns to the GPP the execution status and enters the idle state. This event occurs, for example, at 25 ms in Figure 16-a. The ARM wake-up latency is represented by the power level of 0.66 W which is the sum of the power consumption of both ARM and DSP in idle state (0.43 W + 0.23 W) as described in Table 5. The ARM wake-up is represented by the power transition to 0.83 W level which is the sum of the ARM active state (0.59 W) and the DSP idle state (0.23 W). Then, the ARM sends the parameters (the next frame to decode) to the DSP codecs and triggers a DSP decoding function.

Figure 16-c and Figure 16-d show the power consumption variation in case of 4cif and qcif ARM decoding. Like the DSP decoding, the frame decoding phase is characterized by an increase in the power consumption. The decoded frame copy does not appear clearly as in the case of DSP decoding since the frames are decoded in the ARM cache and evicted when no space is left in the cache.

One can observe that the amount of time spent in frame decoding as compared to the total video decoding time varies according to the video resolution. The complete measured time and energy overhead are given in Table 6.

Table 6. ARM and DSP decoding times and energy overhead.

Resolution	ARM						DSP					
	Time (ms/frame)			Energy (mJ/frame)			Time (ms/frame)			Energy (mJ/frame)		
	Decoding	Total	Overhead (%)	Decoding	Total	Overhead (%)	Decoding	Total	Overhead (%)	Decoding	Total	Overhead (%)
qcif (128 kb/s)	2.19	2.87	10.04	1.20	1.54	22.07	1.97	4.16	31.01	1.71	2.63	34.98
cif (1024 kb/s)	10.85	12.04	9.88	6.18	6.87	10.04	6.016	8.36	28.11	5.35	6.72	20.38
4cif (5120 kb/s)	47.23	52.39	9.86	27.39	28.4	3.55	23.73	25.93	8.48	21.59	22.16	2.5

The time overhead percentages are 31%, 28% and 8% of the total frame decoding time in case of qcif, cif and 4cif for DSP decoding. On the other hand, it is almost constant (10%) in case of ARM decoding. We notice that the overhead is not negligible. For example, the total qcif DSP decoding time is higher than the one of ARM although the frames are processed faster by the DSP.

On the other hand, the energy overhead is higher in case of DSP decoding. In fact, the ARM/DSP communication contributes to a significant part of the energy consumption especially in case of low video resolutions. We can also notice that keeping the DSP in idle state waiting to receive a decoding request from the ARM participates to the energy overhead. In fact, the DSP is not deactivated when it is waiting to receive the next frame. The idle interval is so short that entering in a deeper sleep mode would have a negative impact on performance. During this timeout, the DSP consumes about 0.23 Watts at 520 MHz without executing any task.

In the case of DSP video decoding, most of the time, the ARM processor is in idle state waiting for the DSP to decode the video frames. This explains the increase of idle time when increasing video resolution. Like in the ARM decoding, the frames with higher resolution are decoded by the DSP in a longer time than low resolution leading to a longer ARM idle time.

Regardless of the time spent in each one of these actions, the consumed total power on ARM and DSP decoding is described by Eqs. (11) and (12) respectively,

$$P_{tot_{arm}} = C_{eff_{arm}}.V^2.f_{arm} + P_{static} \quad (11)$$

$$P_{tot_{dsp}} = C_{eff_{arm}}.V^2.f_{arm} + C_{eff_{dsp}}.V^2.f_{dsp} + P_{static} \quad (12)$$

In case of DSP video decoding, both ARM and DSP processors are involved in the decoding process. At a given time, the power consumption is determined by the effective capacitance of both ARM and DSP as described in Eq. (12). This equation can be simplified using the linear relation between the ARM clock frequencies and those of the DSP (this is specific to the tested hardware platform). In fact, from Table 5 we can notice that f_{dsp}=0.72. f_{arm}. Therefore, Eq (12) becomes:

$$P_{tot_{dsp}} = (C_{eff_{arm}} + 0.72.C_{eff_{arm}}).V^2.f_{arm} + P_{static} \quad (13)$$

We can conclude from this section that the time and the energy overhead are not negligible, especially in the case of DSP decoding of qcif resolution. In addition, when decoding a frame, the power consumption varies depending on the processor state and the executed instruction type but should follow a model $\alpha.V^2.f + P_{static}$ for both DSP and ARM video decoding cases. These results are used in the next section to explain and model the overall performance and energy consumption of a whole video sequence decoding.

2.7.4.4 Video sequence level

Decoding time

We analyzed the video decoding performance in term of number of frames decoded per second (frames/s) which is equal to N/t. **N** is the total number of frames (300 in our tests) and **t** is the decoding time. Figure 17 shows a comparison between ARM and DSP video decoding performance in case of 4cif, cif and qcif resolutions for the considered video sequences. The dark (red) flat surface represents the acceptable reference video displaying rate (30 frames/s).

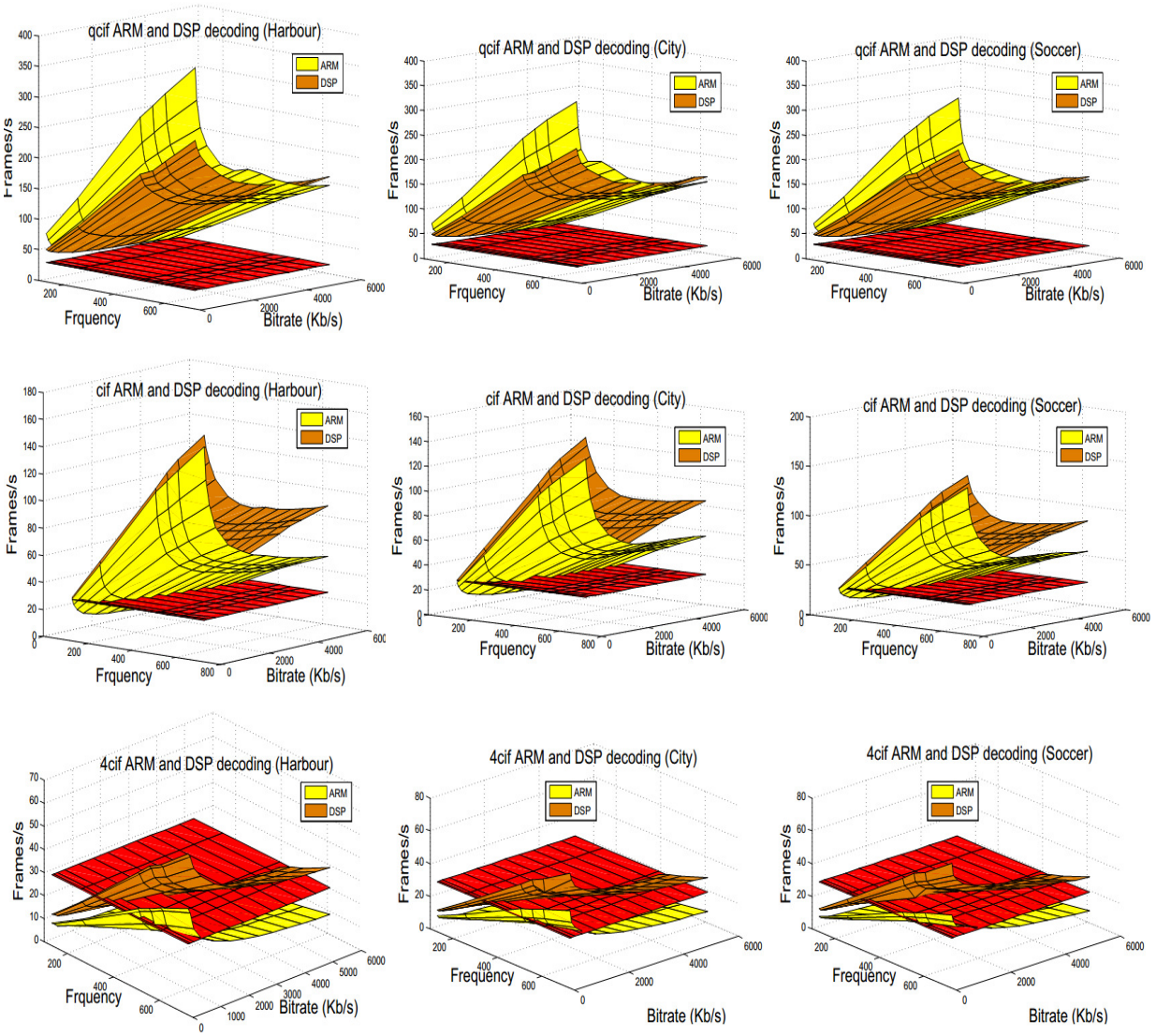


Figure 17. ARM and DSP video decoding performance

The first observation which can be made, in the case of qcif and cif resolutions is that, generally, the video is decoded at a higher rate than the displaying rate even for low clock frequencies (for qcif) regardless of the video bit-rate and the processor type. The ratio between the actual decoding speed and the displaying rate increases for high clock frequencies and low bit-rates.

In the case of 4cif resolution, a decoding rate higher than 30 Frames/s can be achieved by the DSP starting from 180 MHz frequency (i.e. 250 MHz ARM frequency in the x axis) for low bit-rates and starting from 430 MHz frequency (i.e. 600 MHz ARM frequency) for high bit-rates.

The performance of the ARM processor and the DSP are almost equivalent in case of qcif resolution. However, the ARM decoding speed is 43% higher than DSP's in case of 64 Kb/s bit-rate while the DSP decoding speed is 14% higher than the ARM for 5120 Kb/s. For cif and 4cif resolutions, the DSP decoding is almost 50 % faster than the ARM's in case of cif resolution and 100% in case of 4cif. This ratio decreases drastically for low bit-rates.

Power consumption

Figure 18 illustrates the variation of the average power consumption of ARM and DSP video decoding according to video resolution and bit-rate in case of the Harbor video (the Soccer and City video sequence gave similar results). We notice that power consumption depends mainly on the clock frequency, which is explained by the dominance of the

dynamic power model on the static one. For example, at 720 MHz, the static power is 19.75 mW which represents 3.4% and 2.8% of qcif video ARM and DSP decoding power consumption (540 mW and 700 mW respectively).

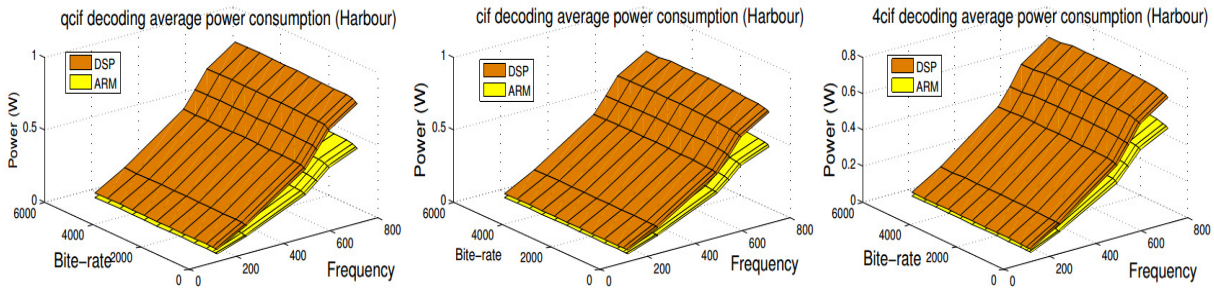


Figure 18. ARM and DSP video decoding power consumption

Energy consumption

Previous results showed significant variations of DSP/ARM performance and power consumption. These variations depend on the clock frequency, the video bit-rate and the resolution. The energy consumption is the combination of the power consumption and the decoding time properties. Figure 19 shows the energy consumption of the ARM and the DSP video decoding (mJ/Frame) in case of 4cif, cif and qcif resolutions for Soccer, Harbor and City video sequences.

The DSP qcif video decoding consumes 100% more energy than the ARM in case of low bit-rate and 20% for high bit-rates. This is explained by a lower performance and a higher power consumption of the DSP decoding as compared to the ARM because of the system overhead (see Table 6). On the other hand, The DSP 4cif video decoding consumes less energy than the ARM although it consumes 60% more power. This is due to a better DSP decoding performance, which can be 100% higher than the one of the ARM.

In case of cif resolution, we noticed a crossing between ARM and DSP energy consumption levels. In fact, for low bit-rate starting from 1 Mb/s, the decoding on ARM consumes less energy than the DSP. The opposite is true for high bit-rates videos.

Analysis of video sequence level results shows that performance and energy consumption of ARM and DSP video decoding highly depend on video quality. In fact, from the results obtained from the frame level characterization step in section 2.7.4.3, we have noticed that in case of low video qualities, DSP performance and energy efficiency tend to drop as compared to those of ARM processor. This is explained by the non-negligible overhead due to the ARM-DSP communication. This overhead is independent from the video quality and is responsible of a significant part of the processing time and the consumed energy.

2.7.5 Modeling step for video decoding characterization results

Based on performance and energy consumption measurement data from the previous section, we describe hereafter the construction of the energy analytical model based on the sub-models decomposition methodology described in Section 2.7.1.2

2.7.5.1 Video rate sub-model

In this step (see block 1 of Figure 14), a model fitting is performed on the **(qp, r)** values obtained from the encoder (see Figure 15) to approximate **a**, $\mathbf{r}_{rmax}$, and $\mathbf{q}_{min}$ parameters of the model proposed in [76] and described in Eq. (8). Table 7 shows the model fitting results. This model has a very good precision especially for high resolution videos (R^2 values around 97%).

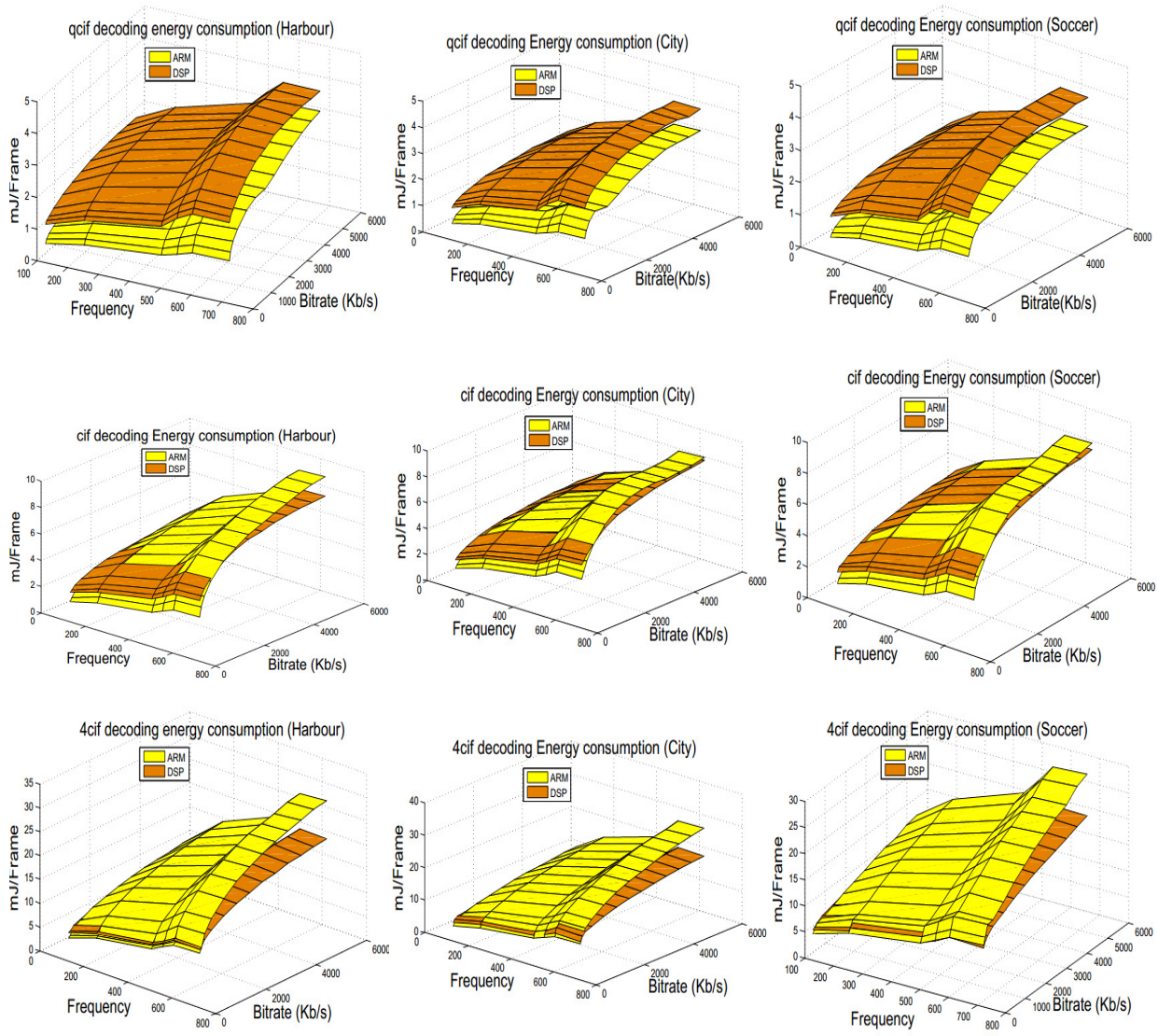


Figure 19. ARM vs DSP decoding energy consumption of H.264/AVC video

Table 7. Model fitting results of the bit-rate model.

Resolution	Video	a	r_{max} (Kb/s)	q_{min}	R^2
qcif	Harbor	1.031	475.5	10.15	0.9615
	Soccer	0.993	177.5	18.14	0.9978
	City	1.026	75.72	54.74	0.9726
cif	Harbor	1.393	1422	14.54	0.9937
	Soccer	1.084	327.9	32.84	0.996
	City	1.353	106.3	47.53	0.9972
4cif	Harbor	1.538	644.3	63.24	0.9913
	Soccer	1.25	552.7	54.06	0.9855
	City	1.34	115	147	0.9805

2.7.5.2 Power sub-model

In this section, we show the model of the average power consumption for ARM and DSP video decoding in terms of video resolution and bit-rate (see Figure 18). We first describe the estimation of static power then we model the dynamic power.

Static power sub-model

In order to build the energy model, we used the static power model provided by the OMAP3530 data-sheet [100] (see Table 5).

Dynamic power sub-model

One can notice that the dynamic power represents the major part of the total power in case of high frequencies. For example, the static power represents 0.02 W of the 0.6 W ARM total power consumption at 720 MHz (3.33%). On the other hand, it represents 0.00516 W of the 0.05 W total power at 125 MHz frequency (10%). Thus, considering the static power is important especially for low-frequencies.

Based on the frame level power consumption analysis in Section 2.7.4.3, we know that for both ARM and DSP video decoding, the dynamic power consumption should follow the model described in Eq. (3). In addition, as it can be noticed in Figure 18, it appears that the power consumption does not depend on the bit-rate **r**. Therefore, we can suppose that:

$$\frac{P_{dyn}}{V^2} = \frac{P_{tot} - P_{static}}{V^2} = \alpha . f$$

Where $\alpha = C_{eff_{arm}}$ in case of ARM decoding and $\alpha = C_{eff_{arm}} + 0.72. C_{eff_{arm}}$ in case of DSP decoding, see Eq. (11) and (13).

For a validation purpose, we executed a multi-linear regression for $\frac{P_{tot} - P_{static}}{V^2}$ in terms of **f** and **r**. The coefficients of the **r** variable are almost nil, which confirms that the average power is independent from the bit-rate. On the other hand, the coefficients of α are shown in Table 8 for Harbor, Soccer and City video sequences. We observe that the regression coefficient representing the $\mathbf{C_{eff}}$ dynamic power model parameter is almost constant for a given video resolution and increases in case of a higher resolution. Column $C_{eff_{arm}}$ represents the effective capacitance of the ARM processor in case of ARM video decoding. On the other hand, the column $C_{eff_{arm-dsp}}$ represents the effective capacitance of both ARM processor and DSP when using the DSP to decode a video. We observe that $C_{eff_{arm-dsp}}$ increases with the video resolution unlike $C_{eff_{arm}}$ which is almost constant.

Table 8. Model fitting results of the dynamic power model.

Resolution	Video	$C_{eff_{arm}}$	$C_{eff_{arm-dsp}}$
qcif	Harbor	4.20E−007	5.48E−007
	Soccer	4.19E−007	5.53E−007
	City	4.17E−007	5.51E−007
cif	Harbor	4.28E−007	6.03E−007
	Soccer	4.23E−007	6.05E−007
	City	4.21E−007	6.04E−007
4cif	Harbor	4.18E−007	6.47E−007
	Soccer	4.17E−007	6.46E−007
	City	4.15E−007	6.44E−007

Decoding time sub-model

In this step, we exploit a linear relation that we observed between $(\mathbf{1/t})$ and both **f** and $\mathbf{qp_{avg}}$ as illustrated in Figure 20. A formal multi-linear regression analysis confirmed the observation. The results of this regression showed that the decoding time can be described by Eq. (14). The values of coefficients α_0, α_1, α_2,and α_3, obtained by the multi-linear regression analysis, are shown in Table 9.

$$\frac{1}{t} = \alpha_0 + \alpha_1 . f + \alpha_2 . qp_{avg} + \alpha_3 . qp_{avg} . f \quad (14)$$

The obtained results, in Table 9, show the accuracy of the proposed model as it can be seen from the R^2 values, which are calculated for each test defined by a video sequence, a resolution and a processor type (ARM or DSP). To have a decoding-time model as a function of the bit-rate, the qp parameter in the above model can be expressed in terms of bit-rate using the rate-model in Eq. (10). Eq. (14) becomes:

$$\frac{1}{t} = \alpha_0 + \alpha_1 . f + (\alpha_2 + \alpha_3 . f) . (4 + 6. \ln_2(q_{min} . \left(\frac{r}{r_{max}}\right)^{-1/a})) \quad (15)$$

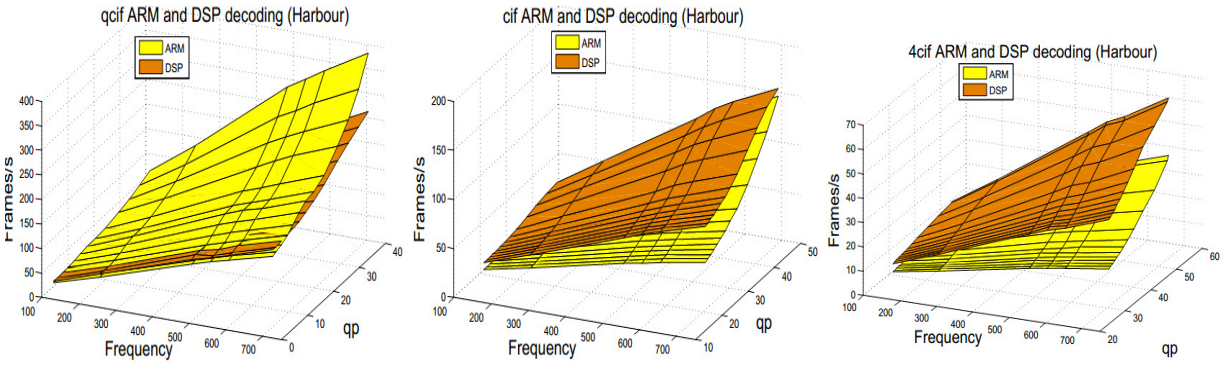


Figure 20.ARM and DSP video decoding time in terms of f and qp

Table 9.Multi-linear regression of 1/t in terms of f and qp

Resolution	Video	Processor	α_0	α_1	α_2	α_3	R^2
qcif	Harbor	ARM	2.040e+00	1.895e−04	2.166e−01	7.994e−06	0.9739
		DSP	2.756e+00	2.178e−04	6.841e−02	4.021e−06	0.9915
	Soccer	ARM	3.482e+00	2.165e−04	1.733e−01	7.352e−06	0.9929
		DSP	2.795e+00	2.300e−04	4.703e−02	3.795e−06	0.9961
	City	ARM	3.332e+00	1.195e−04	1.365e−01	6.325e−06	0.9895
		DSP	2.385e+00	1.908e−04	3.573e−02	2.911e−06	0.9913
cif	Harbor	ARM	−1.680e+00	−1.943e−05	1.395e−01	4.825e−06	0.9782
		DSP	−1.945e−01	5.243e−05	9.438e−02	3.877e−06	0.994
	Soccer	ARM	3.356e−02	3.622e−05	9.569e−02	3.342e−06	0.9919
		DSP	1.181e+00	8.893e−05	3.783e−02	2.950e−06	0.9961
	City	ARM	3.128e−02	3.021e−05	8.775e−02	3.229e−06	0.9901
		DSP	1.021e+00	6.320e−05	2.803e−02	3.120e−06	0.9913
4cif	Harbor	ARM	−1.078e+00	−1.995e−05	5.405e−02	1.559e−06	0.9911
		DSP	−4.915e−01	1.458e−06	2.906e−02	1.850e−06	0.9985
	Soccer	ARM	−1.749e−01	6.355e−06	3.101e−02	9.765e−07	0.9023
		DSP	6.244e−03	3.984e−05	1.502e−02	9.909e−07	0.8651
	City	ARM	−1.591e−01	5.505e−06	2.913e−02	6.564e−07	0.9523
		DSP	5.564e−03	7.654e−05	2.277e−02	6.097e−07	0.92651

Energy model

Based on dynamic power model, static power model and decoding time model described respectively in Eqs. (1), (3) and (15), video decoding energy consumption can be calculated as follows:

$$E = \frac{C_{eff}.V^2.f + P_{static}}{\alpha_0+\alpha_1.f+(\alpha_2+\alpha_3.f).(4+6.\ \ln_2(q_{min}.\left(\frac{r}{r_{max}}\right)^{-1/a}))} \quad (16)$$

This model describes the energy consumption in terms of clock frequency **f** and bit-rate **r** in addition to the constant parameters $\mathbf{C_{eff}}$, α_0, α_1, α_2, α_3, $\mathbf{q_{min}}$, $\mathbf{r_{max}}$ and **a**. These parameters are discussed in the next section.

2.7.5.3 Extracted model parameters discussion

We briefly discuss in this section our semi empirical model parameters. In effect, even though our designed model was built empirically, we could relate each parameter to architecture, system, or applicative behavior.

Architecture related parameters

Constant parameters extracted when building the performance sub-model (see Eq.(14)) are related to memory hierarchy architecture. In fact, we can observe from the analytical model described in Eq.(14) that **1/t** depends on the frequency **f**, **qp**, and the correlation existing between **f** and **qp** weighted with the coefficient α_3. A non-null $\alpha_3$value means that the decoding speed-up, when scaling the clock frequency **f,** depends on **qp** parameters. This can be explained by the fact that: (1) The memory access rate increases for high quality video (low qp value) [105][106][107]. (2) Unlike cpu-bound instructions, memory-bound instruction execution times do not scale when varying the clock frequency [108] due to the memory wall problem [97][109]. This is illustrated graphically by a twisted surface around **qp** and **f** axis in Figure 20.

At a constant **qp**, **1/t** varies by a factor of $(\alpha_1 + \alpha_3.qp_{avg})$ in terms of **f**. This factor reaches its minimum value when **qp**=0, which corresponds to a lossless H.264/AVC coding. On the other hand, at a constant f, 1/t varies by a

factor $(\alpha_2 + \alpha_3.f)$ in terms of **qp**. This factor reaches its minimum value when f is small. This is because when **f** decreases, the difference between the processor frequency and the memory frequency decreases. Consequently, the decoding time is not impacted considerably when the memory-bound instruction rate varies. Theoretically, this factor can be null (which means that the decoding time is independent from **qp**) in one of these cases: (1) the size of the cache memory is large enough to hold the entire video sequence or (2) the processor is clocked at the same frequency as the memory. However, these configurations are not realistic. Thus, the decoding time depends on the combination of α_1, α_2, and α_3.

System related parameters

From some system profiling results [18] obtained thanks to Oprofile tool [112], we have noticed that the time spent by DSP/ARM processors in active or idle state highly depends on the decoded video resolution. In the developed energy model, this is reflected by a different value of the $\mathbf{C_{eff}}$ parameter (see Table 8). The more the processor spends time in idle state, the smaller its effective capacitance. However, this does not mean higher energy efficiency. Indeed, in idle state, the processor does not perform any useful task. To increase its energy efficiency, one may use lower clock frequency with DVFS [101] or enter deeper low-power modes using Dynamic Power Management (DPM) [113].

Video related parameters

The parameters extracted from the video complexity characterization phase are only related to the video. The parameter **a** is an exponent which controls how fast the video rate changes in terms of the step size parameter. On the other hand, $\mathbf{r_{max}}$ and $\mathbf{q_{min}}$ depend on the video complexity. The higher is the complexity of a video, the higher is its corresponding $\mathbf{r_{max}}$ parameter and the lower is $\mathbf{q_{min}}$ [76]. Indeed, one can observe from Table 7 that the higher is the complexity of the decoded video (Harbor is the more complex and City is the less complex one), the higher is the value of $\mathbf{r_{max}}$ and the lower is the value of $\mathbf{q_{min}}$. The value of the parameter **a** seems to be independent from video complexity.

2.7.6 Models validation

We have analyzed, in the previous sections, the accuracy of each developed sub-model (rate, time and power). The objective of this section is to analyze the accuracy of the performance and energy analytical models (Eqs. (15) and (16)) obtained from the combination of sub-models in (Eqs.(1), (3) (10) and (14)).

2.7.6.1 Decoding time model

The Table on Figure 21 shows the accuracy of the performance model described in Eq. (15) (Frames/s in terms of **f** and the bit-rate) obtained from the combination of the rate sub-model described in Eq. (10) ($\mathbf{qp_{avg}}$ in term of the bit-rate) and the multi-linear time sub-model described in Eq. (14) (Frames/s in terms off and $\mathbf{qp_{avg}}$). The calculated R^2 coefficients are about 98%.

The right part of Figure 21 shows an example on the comparison between the predicted performance values and the measured ones. One can notice that the combination of the same rate model in Eq.(10), which depends exclusively on the video properties, with the multi-linear time sub-model in Eq. (14), depending on the execution platform, allows to build an accurate performance model for both ARM and DSP processors.

Video	Processor	Energy model R^2		
		qcif	cif	4cif
Harbour	ARM	0.9851	0.9840	0.9886
	DSP	0.9750	0.9789	0.9787
Soccer	ARM	0.9753	0.9813	0.9811
	DSP	0.9699	0.9701	0.9687
City	ARM	0.9749	0.9803	0.9801
	DSP	0.9699	0.9689	0.9797

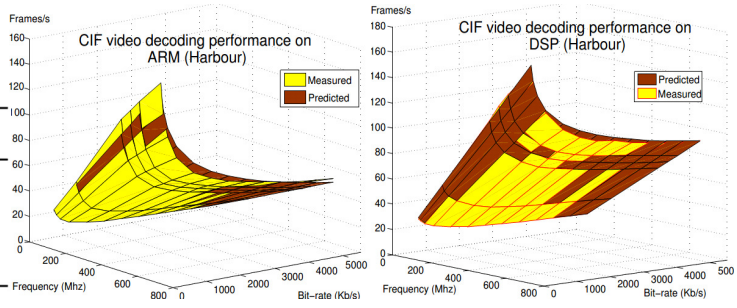


Figure 21. validation of the video decoding performance model (OMAP3530)

2.7.6.2 Energy model

The table on Figure 22 shows the calculated R^2 values of the energy model in Eq. (16) as compared to the measures energy consumption. It is around 97% for almost all the video sequences for both ARM and DSP (see right part of Figure 22 for an illustration).

Video	Processor	Energy model R^2		
		qcif	cif	4cif
Harbour	ARM	0.9851	0.9840	0.9886
	DSP	0.9750	0.9789	0.9787
Soccer	ARM	0.9753	0.9813	0.9811
	DSP	0.9699	0.9701	0.9687
City	ARM	0.9749	0.9803	0.9801
	DSP	0.9699	0.9689	0.9797

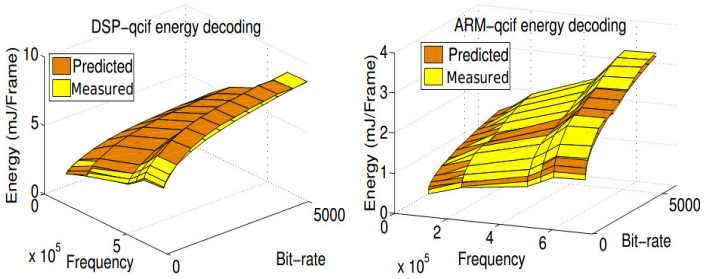


Figure 22. validation of the video decoding energy model (OMAP3530)

2.7.7 Applications

We give, in this section, different ways to use the models proposed in this study. Before doing so, we present a motivational example where the developed models can be used.

2.7.7.1 Motivational example

To cope with the network bandwidth fluctuation and the heterogeneous mobile device capabilities, more and more video content providers such as Youtube and Netfilix, support adaptive video streaming. As illustrated in Figure 23, in adaptive video streaming [114], a video stream **S** is divided into sequential and independent elementary segments $\mathbf{S_1}$, $\mathbf{S_2}$,…$\mathbf{S_n}$. Each segment $\mathbf{S_i}$ represents a few seconds video sequence and is coded into different video qualities $\mathbf{Q_1}$, $\mathbf{Q_2}$, $\mathbf{Q_m}$. A video segment $\mathbf{S_i}$ having the quality $\mathbf{Q_j}$ is represented by a chunk $\mathbf{S_{ij}}$. For example, a two minutes video sequence may be divided into 60 segments of 2 s duration each. Each segment may be coded into 512 Kb/s, 1 Mb/s, 2 Mb and 4 Mb/s bitrate video qualities. The video decoder may then switch dynamically between different video qualities according to network bandwidth variation or battery level.

In this context, adaptive video streaming poses a new challenge to energy aware video decoding. In fact, the video decoder should take into consideration the runtime dynamic variation of the video quality in dimensioning its processing resource to save energy.

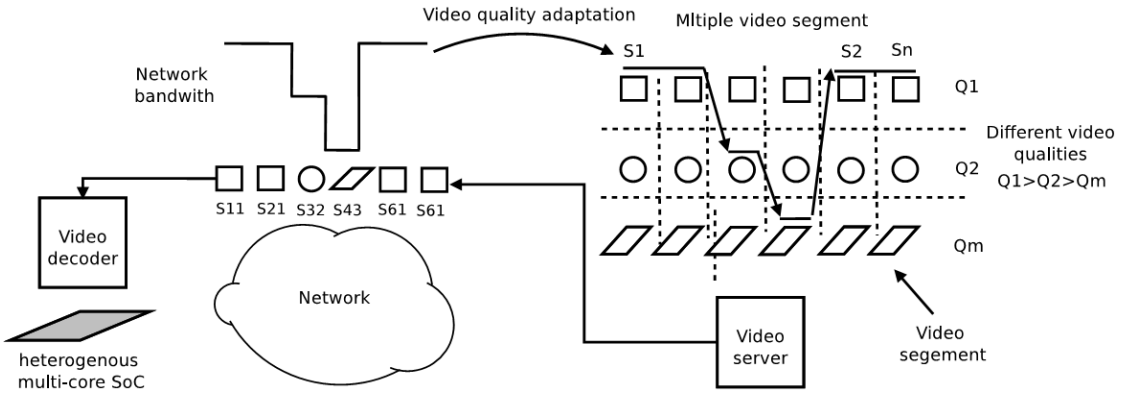


Figure 23. Adaptive streaming

2.7.7.2 Energy aware scheduling of video decoding on heterogeneous multi-core SoCs

It was highlighted in Section 2.7.4.4 that the energy efficiency of DSP video decoding as compared to that of GPP highly depends on the video quality. As illustrated in Figure 24-a, the combination of energy models developed for GPP and DSP may result in an optimal energy model if low video qualities are decoded on the GPP and high video

qualities on the DSP. This is particularly interesting in the above introduced adaptive video decoding context. As illustrated in Figure 24-b, when a video decoder adapts dynamically the quality of the video, it may decide accordingly on which processor it is decoded depending on the video quality. More details on this application can be found in [19].

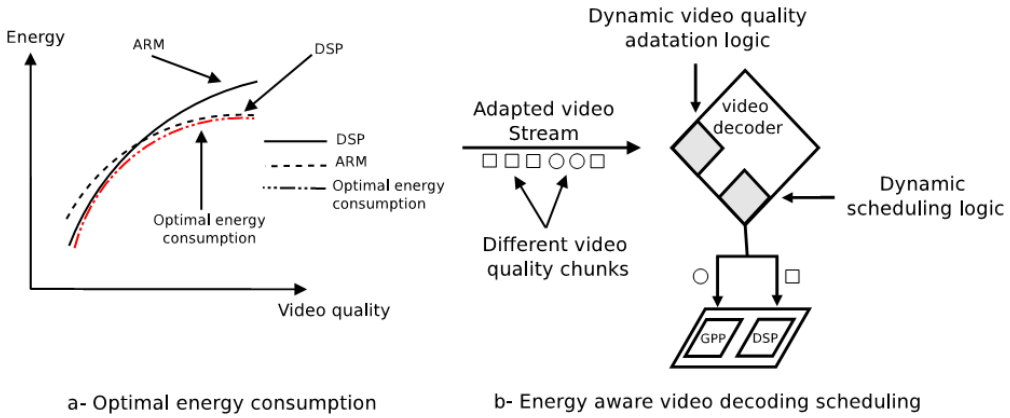


Figure 24.Video-quality/Energy Aware Video Decoding on SoCs.

2.7.7.3 Video-quality aware dynamic voltage and frequency scaling

As shown in Section 2.7.4.4, the video decoding performance highly depends on the video quality. In the context of DVFS, an adaptive video decoder should react to a video quality change to adjust processor frequency accordingly. If the video related parameters a, q_{min}, R_{max} are sent for each video segment, then, using the rate model, the video decoder may calculate qp_{avg} corresponding to the selected chunk. This would avoid to parse all the frames of the video chunk to extract the quantization parameter and calculate the average value. Thus, as illustrated in Figure 25, assuming the multi-linear performance model (Frame/s in terms of f and qp_{avg}), the optimal processor clock frequency f_{optim} allowing a decoding speed as close as possible to the displaying rate can be calculated proactively for a given video quality.

One can highlight that the performance model makes it possible to predict the average decoding performance. To decouple the constant frames displaying speed from the frame-to-frame workload variation, a buffer may be introduced between the decoder and the displaying device as suggested in [115].

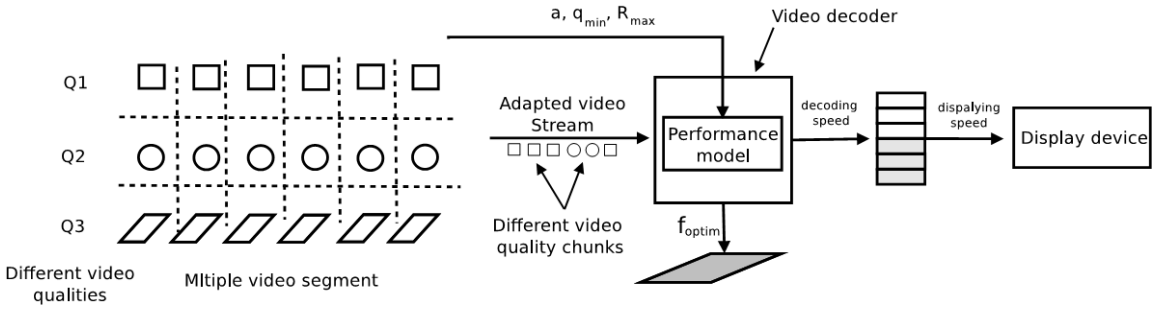


Figure 25. Video-quality/Energy Aware Video Decoding on SoCs.

2.7.7.4 Video-quality/energy trade-off

One of the advantages of the energy model described in Eq. (16) is that it separates between the architecture/system related parameters and the video parameters. This allows to evaluate, for a given video, the impact of the quality change on the consumed energy by changing the value of the bit-rate only. If we evaluate the video quality corresponding to each bit-rate (using PSNR or any QoE metric), we can select the Video-quality/Energy trade-off for a given energy-constrained configuration. This can be used for example in a multi-objective (Video-quality maximization, Energy minimization) video decoding technique [116].

2.8 Conclusion

This chapter presents an end-to-end approach for performance and energy consumption characterization and modeling of GPP and DSP for video decoding application. The experimental step of this study allowed to understand the impact of architectural and system design on the overall performance and energy consumption when using different video qualities. Indeed, obtained results revealed that the best performance-energy trade-off highly depends on the required video bit-rate and resolution. For instance, the GPP can be the best choice in many cases due to a significant overhead in DSP decoding, this overhead may represent more than 30% of the total decoding energy.

In the modeling step, using a sub-model decomposition approach, performance and energy consumption results obtained in the characterization phase allowed to build an analytical model in terms of video bit-rate and clock frequency variable parameters in addition to a set of comprehensive constant parameters related to video complexity and underlying architecture. The developed model is accurate (97% R^2 value) for both GPP and DSP video decoding. Moreover, it was shown that the combination of these different sub-models allows to build an accurate high level performance and energy model for video decoding.

Reusing the developed models is very important and so we investigated a methodology to extract the model's parameters online for different videos and different architectures. Concerning video related parameters, they can be extracted online. $\mathbf{a}$, $\mathbf{q}_{\mathbf{min}}$, and $\mathbf{r}_{\mathbf{max}}$ can be calculated from the video encoding phase and sent to the decoder. So the video decoder can calculate the $\mathbf{qp}$ parameter according to Eq. (10). Thus, assuming the multi-linear model described in Eq. (14), online performance model building becomes easier. For example, an adaptive linear filtering technique [118] may be used to adjust α_0, α_1, α_2, and $\alpha_3$parameters online.

The **applications** of the developed models are numerous, some of them have been described in [18][84][119]. The simplest application one can think of is the switching between DSP and GPP for video decoding according to video quality [ri6]. Other applications have been investigated in the context of adaptive video streaming. Adaptive video streaming poses a new challenge to energy aware video decoding. In fact, the video decoder should take into consideration the runtime dynamic variation of the video quality in dimensioning its processing resource to save energy, some examples are provided in [18].

My work on this topic has significantly reduced in late 2015 apart from a study we recently achieved on adaptive DVFS algorithm for energy efficient video decoding using metadata (normalized by the MPEG *Green Metadata* standard) [117] and for which my participation was modest. In fact, I currently focus on storage related topics as previously mentioned.

2.9 Outcome

Advised PhD student	**Project**	**Collaborations**	**Publications**		
			Journals	**Int conf.**	**Others**
Dr. Yahia Benmoussa	ANR OpenPeople*	UBS Lorient (E. Senn), UMBB Algeria (D. Benazzouz)	3	4	2

3 PERFORMANCE AND ENERGY CONSUMPTION OF STORAGE IN EMBEDDED SYSTEMS

3.1 Summary

This section describes our work on performance and energy consumption of embedded storage systems based on flash memory. This chapter summarizes the three steps of our methodology: **experimental step** for exploring performance and energy consumption of I/O storage stack in which we highlight the major components of the OS that influence those metrics, **modeling** I/O performance and energy consumption through sub model decomposition, and **application step** that allowed building an accurate simulator for performance and energy exploration of embedded storage systems. This section is mainly based on the PhD thesis of Pierre Olivier.

3.2 Context

The increasing complexity of embedded systems pushes designers to compose with a very large design space. Estimation of metrics such as performance and power consumption is critical in this design phase to make fast and sound design choices. Indeed, as previously discussed, performance and power consumption are crucial in embedded systems. For instance, multimedia applications running on complex embedded systems require high performance to offer a suitable quality of service. Moreover, embedded systems are often battery backed. Their power consumption impacts their battery operating time and lifetime. In addition, estimation techniques are also useful in many cases when performing measures is not possible due to time or financial constraints, for example: prototyping and new system design, and optimization.

Storage systems are a well-known performance bottleneck, and embedded storage is not an exception [143]. NAND flash memory is the main storage media in embedded systems. This type of memory exhibits specific constraints, and relies on some specific management mechanisms. These mechanisms play a major role in flash storage system performance and power consumption. In an embedded system, the power consumption of a flash memory chip might seem relatively low as compared to other components that are CPU and RAM. Nevertheless, it has been shown that the flash management layer has a high impact on power consumption in mobile devices [148].

Flash management can be implemented by a hardware component, the Flash Translation Layer (FTL). It can also be implemented purely in software through some dedicated Flash File Systems (FFS). Today, the FTL is used in SSD, USB drives, SD/MMC cards, eMMC chips, etc. The abstraction layer brought by the FTL raises issues, as evidenced by the need to introduce FTL specific commands (TRIM) or FTL specific file systems such as F2FS [135]. On the other hand, FFS are widely used in embedded systems[11]. This is due to multiple reasons: they are cheaper, less complex than FTL, and constitute a direct link between the applicative layer and the flash storage media [170][167]. Most popular FFS are Open Source.

3.3 Problem statement

Very few state-of-the-art studies propose characterization methods, models or simulators for FFS performance and power consumption. Performance and power consumption metrics are impacted by several factors belonging to a complex software / hardware environment in which an FFS evolves. This large number of factors brings the need for sub-model decomposition to hierarchically represent the impact on performance and power consumption of the different software / hardware components. To build up these models, one should provide a detailed analysis of these metrics for FFS based storage systems.

3.4 Approach

In this context, we proposed a methodology to estimate FFS storage performance and power consumption. Estimations are obtained through system models execution. We have built those models by taking into account all the layers of a standard FFS management stack, from the application down to the flash storage media. We provided a set

[11] They were when we started this study and their use declined with the large adoption of eMMC interface in embedded systems.

of models, integrated in a stand-alone simulator, allowing to obtain estimations for performance and power consumption based on a system description and application I/O workload.

Our methodology is divided into three successive steps. A first step is to understand the performance and power consumption behavior of FFS based storage systems. This is the role of the **experimental and exploration phase**. In this phase, we identified the main components of FFS storage systems impacting performance and power consumption. In the **modeling phase** the impact of those elements is formalized into models. Those models describe the performance and power consumption behavior for the elements identified in the exploration phase. To obtain estimations, the models need an execution environment. This is achieved in the last phase, the **validation and simulation phase.** In this phase models are implemented in a simulator, namely OpenFlash. OpenFlash provides performance and power consumption estimations. It was also used to validate the models.

In this chapter, we describe each phase of the methodology. In addition, we present an application of the methodology as a case study. It concerns the performance and power consumption of the JFFS2 FFS, executed with Linux on an embedded board. Linux is selected because it is one of the most used embedded kernels [171], and it supports most popular FFS.

The contributions presented in this study are the following:

- An analysis and identification of all the elements of FFS-based storage systems impacting performance and power consumption. It is done through performance and power consumption measurements with micro-benchmarks ;
- A set of models formalizing this impact, and their concrete application to the JFFS2 FFS with embedded Linux. We chose JFFS2 because it is a popular and mature FFS;
- A simulator named OpenFlash, implementing the models and giving estimation for FFS storage systems executing a given I/O workload.

The work presented in this section was extracted from Pierre Olivier's PhD thesis and the following publications: [ri3][ri8][ri11][ri13][ci8][ci16][ci18][ch2].

3.5 Background

3.5.1 NAND flash memory basics: architecture, operations, and constraints

A NAND flash chip architecture is organized hierarchically. Its internal structure is depicted in Figure 26. A chip contains one or several planes. Planes are composed of blocks, and a block contains several pages. Current page sizes vary from 2 KB to 16 KB. Generally, a block contains 64 or 128 pages [131]. A flash chip contains one, two or four planes. NAND flash memory is an EEPROM and so supports three operations: the usual read and write (or program) operations are performed on a page and the erase operation is achieved on a whole block. In terms of performance, read latencies range from 20 to 100 µs, while write latencies range from 200 to 300 µs, and erase latencies from 500 to 1500 µs. The average read power varies from 10 to 20 mW, and 20 to 50 mW for write and erase operations [131].

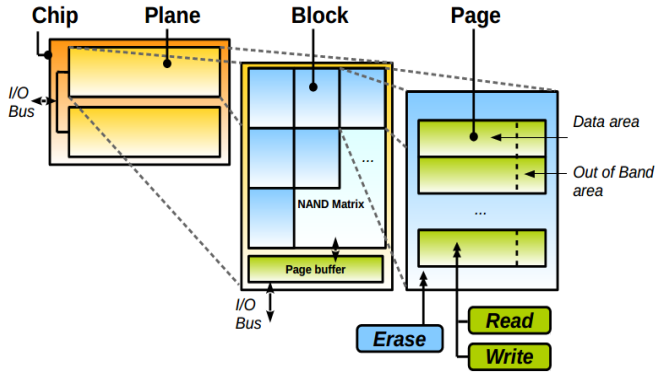


Figure 26. Simplified view of a NAND flash chip architecture

There are specific constraints related to the use of this memory. The first constraint is the **erase-before-write** rule. Due to the technology and architecture of a flash memory chip, it is not possible to write in a page that already contains data (in-place data updates). Before being able to write in a page containing data, the block containing this page must be erased first. The limitations brought by this constraint are reinforced by the asymmetry between write and erase operation granularities: page versus block. Moreover, there is a large difference in terms of execution times for these two operations. A second constraint is the **wear-out**. A flash memory cell can only sustain a limited number of erase operations. After a given threshold is reached, the cell becomes unusable. Blocks containing worn-out cells are called bad blocks and must not be used by the host OS. The number of write / erase cycles that can be sustained by a flash memory cell varies between 5 000 and 100 000 according to the flash chip technology used. A third and last constraint is the low **reliability** of flash storage and operations. During flash operations, bit flips may occur. To reduce these errors, writing pages sequentially within a block may be necessary.

Because of the aforementioned constraints, specific flash management mechanisms are used to make flash memory usable in today's computer systems. These mechanisms can have a significant impact on the performance and power consumption of the storage system.

3.5.2 Flash memory constraints management

Flash storage system management mechanisms cope with the erase-before-write limitation through the implementation of **logical to physical address mapping** mechanisms. This makes out-of-place data updates possible at the flash level. Logical addresses, viewed by the host OS, are mapped to physical addresses on flash memory. After a data update, the page containing the old data version is not erased straightaway. Block erasures are performed asynchronously by the **Garbage Collector** (GC). The goal of the GC is to recycle invalid flash space into free space. The GC implementation details depend on the considered flash management system. Nevertheless, GC work can be summarized as follows: (1) selection of a victim block(s), (2) copy of potential still valid pages from the victim block to other locations, and (3) erasure of the victim block (s). GC execution can be triggered online with an upcoming write request when the amount of free flash space reaches a given threshold (synchronous GC). It can also be executed in background during I/O timeouts (asynchronous GC).

Concerning flash wear out, flash management mechanisms try to evenly distribute write and erase operations over the whole flash memory space to maximize memory lifetime. This concept is called **wear leveling**. Wear leveling can be taken into account in the write strategy of flash management mechanisms, and also in the GC victim block selection policy. Moreover, some specific mechanisms dedicated to wear leveling have been proposed [144]. To cope with the low reliability of NAND flash memory, flash management mechanisms implement Error Correcting Code. Some examples are Hamming, Reed-Solomon or Bose-Chaudhuri-Hocquenghem codes.

3.5.3 Flash integration in computer systems, a focus on Flash File Systems

Flash management systems implement the previously presented concepts to cope with flash constraints. There are two types of flash management systems: **Flash Translation Layer** (FTL), and dedicated **Flash File Systems** (FFS). The FTL [126][130] is a hardware and software layer located in the controller of flash based storage peripherals. In addition to flash management mechanisms presented earlier, the FTL emulates a block based device. It allows a transparent integration for the flash storage system at the host OS level. FTL devices are generally used with a standard (hard disk) file system: FAT, NTFS, Ext4, etc.

On the other hand, with dedicated FFS, flash management algorithms are implemented at the file system level. The file system is integrated in the OS. FFS are fully aware of the intricacies of the managed flash media. FFS are widely used in the embedded system domain to manage raw flash chips. In the study presented in this chapter, we focus on FFS. FTL will be discussed in the next chapter.

One can identify three main roles for FFS. First, they must manage flash constraints: FFS implement address mapping, GC and wear leveling policies. Generally speaking, address mapping mechanisms map fragments of logical data to on-flash physical pages. These fragments correspond to files and part of files. Second, FFS must manage embedded systems related constraints, as FFS are mainly used in this domain. Limited processing power and relatively low main memory footprint are traditional embedded constraints. Moreover, as most embedded systems run on battery power, FFS must be resilient to sudden power cuts (unclean unmount). This is achieved through techniques such as journaling

or atomic operations. Finally, FFS play the role of standard file systems, they define how data are written / retrieved to / from the storage media, and how data are organized as a set of files and directories. Moreover, FFS may provide additional functionalities such as data compression [140] or encryption [165].

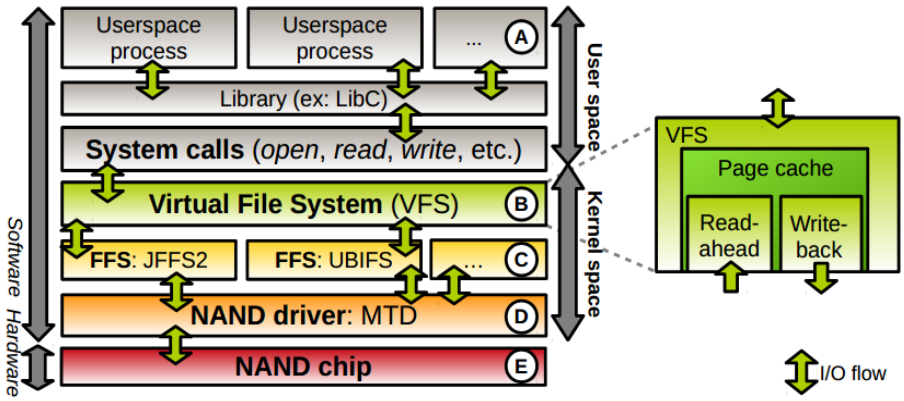


Figure 27. FFS storage management stack with Linux.

Linux is one of the most popular embedded kernels [170], it supports the three main FFS: JFFS2 [172], YAFFS2 [174] and UBIFS [134]. FFS storage management with Linux can be viewed as a software stack, depicted in Figure 27. At the top of this stack, user-space processes (A on Figure 27) access files using system calls such as open, read or write. These system calls are received in the kernel by the Linux Virtual File System (VFS, B on Figure 27). VFS is an abstraction layer for all the file systems supported by Linux. VFS forwards file operations to the actual file system, FFS in our case (C). Finally, a FFS uses the NAND driver (D) to access the flash chip (E). With Linux, all supported raw flash chip models are used with a generic driver called the Memory Technology Device (MTD). One can notice from Figure 27 that by using MTD layer, the FFS does not rely on the I/O block layer, neither does it use the I/O scheduler. As a consequence, all file I/O operations going through MTD are synchronous. Linux maintains at the VFS level a RAM based cache named the **page cache**. The page cache uses all the RAM that is not utilized by the kernel or the running programs to buffer file data as long possible. Conversely, if there is some need to recycle RAM space, the *pdflush* kernel thread is used; it discards clean pages and evicts dirty pages if needed. There are two optimization mechanisms associated with the page cache. Firstly, **read-ahead** is a prefetching algorithm executed during read operations. According to heuristics, read-ahead loads from the file system in the page cache file data that are likely to be accessed in a near future. Secondly, the **write-back** feature allows buffering written data into the page cache before writing them to the file system. This is done to take benefit from temporal and spatial localities exhibited by write operations. From the kernel high level perspective, files are loaded into pages. The standard Linux page size is 4 KB. To avoid confusion, OS pages will be referred to as Linux pages and flash memory pages will be referred to as flash pages in this section.

3.6 Related work

3.6.1 Performance exploration

Exploration is a first and necessary step in the process of modeling system performance and power consumption profiles. Its goal is to identify the main elements of the system impacting these metrics. The modeling effort should subsequently be focused on these elements. Exploration is done through measurements of various performance and power consumption metrics. Because of a lack of standard storage macro-benchmarks in the embedded systems domain, measurements are generally realized under ad-hoc micro-benchmarks.

In the literature, most FFS performance evaluations are comparative studies. Some work compare existing systems in order to determine which one performs the best [151][132][168][159]. Others propose a new FFS [140] [162][150][142][163][138][164][144][135][146] or some optimizations [165][149] and compare their proposition to existing FFS. These are comparative studies and are not realized for the purpose of modeling FFS performance. Thus,

the focus is not set on characterizing the observed performance profiles. Evaluated metrics include performance metrics such as execution times, file system related metrics such as mount time, and wear leveling related metrics.

Concerning power consumption, there are two main metrics: the power consumed by a component at a given time, and the energy, consumed during a period of time. In the literature, one can find studies exploring the power consumption at the level of a flash chip [131][152]. Results sometimes show a difference compared to the values reported on the chip data sheet. It indicates the importance of measurements in real environments. Several studies also focus on the power consumption of SSD [166][175] compared with HDDs [147]. In embedded systems domain, most flash storage power consumption measures aim to validate a model or prove the efficiency of a new proposition. Few work target characterization through measurements. Authors of [68] analyze the power consumption of multiple hardware components in a smartphone. As the power consumption of the flash chip is relatively low, authors show that the power consumed by the components executing the flash management layer (CPU and RAM) is not negligible.

There is no in-depth performance or power consumption exploration study for FFS storage systems. We tackle this problem in the first step of our methodology. In this exploration phase, we identify the main elements of FFS storage systems that impact performance and power consumption. This is done using real measurements and micro-benchmarking.

3.6.2 Performance and power consumption modeling

Several studies propose models targeting flash storage performance and power consumption. Models can be classified according to the level of modeling granularity.

The **micro-architectural level** is the most detailed modeling granularity. FlashPower [156] or NVSim [129] model the performance and power consumption at the cell and circuit level. NandFlashSim [139] is a NAND cycle accurate performance and power consumption model targeting the chip level. Other chip level models are presented in [128][137]. The chip level is very close to the micro-architectural level as they both focus in modeling performance and power consumption of flash operations. In the context of our work, these levels are not sufficient because they do not consider the flash management layer. At the **flash + management layer level**, the management mechanisms are taken into account. Because these algorithms have a high impact on performance and power consumption, several models target this level. Some are ad-hoc, created for use in a specific context [141][124][161][169]. They are used for example to estimate the performance and power consumption of a proposed system. Generic models are also proposed [123][142]. We observe that in this level, both ad-hoc and generic models focus on FTL and do not consider FFS.

The last level is the **OS level**. It considers the flash and management layer components, as well as some components of the host OS. For example, EagleTree models SSD, I/O scheduler, and the behavior of multiple threads generating the I/O workload. Concerning FFS with Linux, related models can be considered to be at the OS level. Indeed, the FFS is part of the OS. The FFS is also strongly coupled with other OS components such as VFS or the NAND driver. To the best of our knowledge, there is no study proposing detailed FFS performance and power consumption modeling. In [125], the authors present a simple macro-model of power consumption for JFFS2. A simple empirical equation is provided for each file system related kernel primitive to estimate the energy consumed by each system call. The energy is modeled for as a linear function of the amount of transferred data. This model is very simple and abstracts flash management layers complexity and system state. These factors cannot be ignored because they have a large impact on FFS performance and power consumption.

There is a lack of performance and power consumption models for FFS. Most of the existing models focus on FTL, or are too abstract. Concerning the positioning of our work on the modeling granularity scale, it is firstly located at the OS level because we consider elements such as system calls and VFS. Moreover, we provide detailed models concerning the management layer (FFS) and the flash chip level.

3.6.3 Flash storage systems simulators

Some of the previously presented models are implemented in simulators. These software tools provide an environment to execute the models and gather performance and power consumption estimations. Several flash simulators exist [154][145][133][139][127].

Most of them focus on FTL, or simulate only the flash memory component at the circuit level. The majority of these tools are discrete events simulators. Their main feature is to allow exploring the design space of flash based storage systems. Moreover, one can identify interesting additional features:

- Power consumption estimations (in addition to performance) [139] ;
- Simulation of advanced flash architectures such as multi plane / LUNs / channels, and the related advanced flash operations [154][145][133][127] ;
- Definition of an initial state for the system before a simulation, for example through a warm-up phase [145][127] ;
- Synthetic I/O workload generation [145][127] ;
- Support for defining new / custom FTL systems [145] ;
- Background operations (GC, wear leveling) support [127].

To the best of our knowledge, there is no simulator dedicated to FFS. In this study we propose a simulator, OpenFlash, targeting FFS systems performance and power consumption to fill the state-of-the-art gap on this subject. Moreover, OpenFlash is built taking into consideration the salient features of existing flash simulators. Power consumption and advanced architecture / commands are supported by OpenFlash. The simulator also supports background operation modeling and FFS prototyping.

3.7 Contribution

3.7.1 Experimental step: performance exploration

3.7.1.1 Overview

The goal of this phase is to investigate the main elements impacting performance and power consumption in the context of a FFS management stack processing I/O workloads. To achieve this, we performed both functional exploration, and performance / power consumption measures. **Functional exploration** consists in understanding the way an I/O workload is processed by the FFS management stack. The output consists in a detailed description of the concerned elements and their impact, aiming to be formalized into models (in the modeling phase). Since our goal is to have estimations at the applicative level, we explored the entire flash storage management stack, composed of Linux VFS, FFS, and driver levels. We chose mainly to focus on file read and write operations. Those operations can be whether initiated by read and write system calls or through memory mapping with *mmap* .

I/O management algorithms have a significant impact on performance and power consumption, so functional exploration is necessary. Functional exploration can be static, or dynamic. **Static exploration** consists in studying the literature and technical documentation concerning FFS [172][173][134], Linux kernel [122], and NAND driver [157]. Studying source code is also possible and sometimes mandatory when one needs to understand some implementations details. **Dynamic functional exploration** consists in tracing at runtime the I/O flow of in real embedded platforms. Tools such as GDB were used. We also developed a set of tools for monitoring FFS events and performance, they are briefly presented below

Measures were also realized to investigate the impact of various elements on performance and power consumption. This is a relatively fine-grained analysis focusing on each one of the FFS management stack levels. We used ad-hoc micro-benchmarks to generate the I/O workloads, and a set of tools we developed [24][25] to extract events and performance metrics. Concerning power consumption measures, we relied on the Open-PEOPLE [98] platform.

3.7.2 Exploration phase

The exploration phase was applied on the Mistral Omap3evm [155] embedded board. It contains a TI Omap3530 CPU with an ARM core clocked at 720 MHz, 256 MB of RAM, and 256 MB of Micron SLC NAND flash [153]. This flash chip contains 64 pages (2KB) per block. Linux kernel 2.6.37 was used. Power measures were obtained with a NI PXI-4472B digitizer [158], with a sampling frequency of 1 KHz. We measured power on the CPU and memory (RAM + flash) power rails.

3.7.2.1 Functional exploration

We explored the kernel algorithms involved in processing read and write system calls, as well as JFFS2 GC behavior. We present hereunder a brief description of the main operations realized by these system calls.

Read operation. When an application executes the read system call, it ends up at the VFS level by a call to *vfs_read*. This function iterates on all Linux pages concerned by the read request, and performs the following for each one of them: if the Linux page is present in the page cache, calling the file system is unnecessary. If not, the FFS (JFFS2 in our case) is called through the function *jffs2_readpage*. This function copies the Linux page from flash to the page cache. JFFS2 first determines the set of nodes containing the data belonging to the Linux page. Next, flash pages containing these nodes are read, with the help of the NAND driver flash page read function. A small buffer is present at the NAND driver level, keeping in RAM the last flash page read. The Linux page is then placed in the page cache. When activated, the page cache read-ahead feature allows the kernel to prefetch data from the file system. Prefetching decisions are based on sequentiality estimation of the workload based on some heuristics [160].

Write operation. A write system call triggers the execution of several high level kernel functions. It leads to a call to VFS *generic_perform_write* function. This function iterates on all Linux pages concerned by the write request, according to the target file offset and the amount of bytes written. Next, for each concerned Linux page, two functions of the file system are called: ***jffs2_write_begin***, and ***jffs2_write_end***. JFFS2 does not use the page cache write back feature and writes are fully synchronous. This is because of potential data loss due power cuts in embedded systems. In ***jffs2_write_begin***, JFFS2 checks if the written Linux page is present in the page cache. Otherwise, the Linux page is read from flash. Flash space reservation is also performed. In ***jffs2_write_end***, the written data is packed into one or several nodes, which are appended into JFFS2 write buffer. Its goal is to buffer nodes with size smaller than a flash page, to avoid internal fragmentation in flash space. The buffer has the same size as one flash page. When the buffer is full, a flash page is written with the buffer content through a call to the NAND driver page write function. When *jffs2_write_end* returns, data is actually written on flash.

Garbage Collection. Garbage collection occurs in JFFS2 (1) when data need to be written and there is not enough clean flash space available and (2) in background through a kernel thread. We call the first execution mode **online** (or synchronous) GC, and the second one **offline** (or asynchronous) GC. Both execution modes use the same entry function named *jffs2_garbage_collect_pass*. The GC selects a victim block, copies the potential still valid nodes to another location, and erases the victim block. The selected victim block is most of the time a block with a large amount of invalid data. Nevertheless, one time out of 100, JFFS2 selects a fully valid block to be reclaimed. This is done for wear leveling purpose to prevent cold data isolation.

3.7.2.2 Performance and power consumption measures

Driver Level. We measured the execution time and the energy consumed for each basic flash operation (read / write / erase) at the MTD NAND driver level. We used micro-benchmarks executed through kernel modules, directly accessing the flash chip. We found no variations in the execution time of each operation when varying the number of operations performed and the access pattern [23]. Our latency measures were higher than the numbers on the data sheets. This overhead was nearly constant for each operation, it represents the driver overhead. Concerning power consumption, the power measured on both CPU and memory chip when performing one type of flash operation was stable. We found that the CPU was active during flash I/O operations, and generated 70 to 80% of the energy due to flash operations. The rest of this energy consumption was due to the memory chip (Flash+ RAM). We also measured the impact of the MTD read buffer by reading the same flash page several times in a row. Because the read operations were satisfied from the buffer (RAM) instead of flash, execution times were reduced by a factor 35, and energy by a factor 17. Latencies and energy consumption for flash operations will be presented in the modeling section.

FFS Level - JFFS2. At the FFS level we measured the execution time and power consumption of *jffs2_readpage*, *jffs2_write_begin* and *jffs2_write_end*. Our measures showed that most of the time and energy for these functions were dominated by flash operations. Generally speaking, flash I/Os represent 90% to 95% of the execution time of *jffs2_write_end*, and 75% to 99% of the execution time of *jffs2_readpage*. This phenomenon is illustrated on the left part of Figure 28. This figure shows a micro-benchmark execution of sequential write operations of 4 KB in a file. The figure presents the execution time and the number of flash page write operations during each call to *jffs2_write_end*. We

can see that the execution time is directly correlated to the number of pages written. Concerning *jffs2_write_begin*, this function does not perform any flash write operations so its execution time and energy are very small as compared to *jffs2_write_end*. Nevertheless, if the Linux page hit by a write request is not in the page cache, it is read during *jffs2_write_begin*. Thus, *jffs2_write_begin* execution time might include the time for a call to *jffs2_readpage*. The number of flash writes realized during the execution of *jffs2_write_end* is related to the volume of data to write. As this amount has a maximum value of one Linux page (4 KB) at this level, the number of flash writes for one execution of this function stays relatively small.

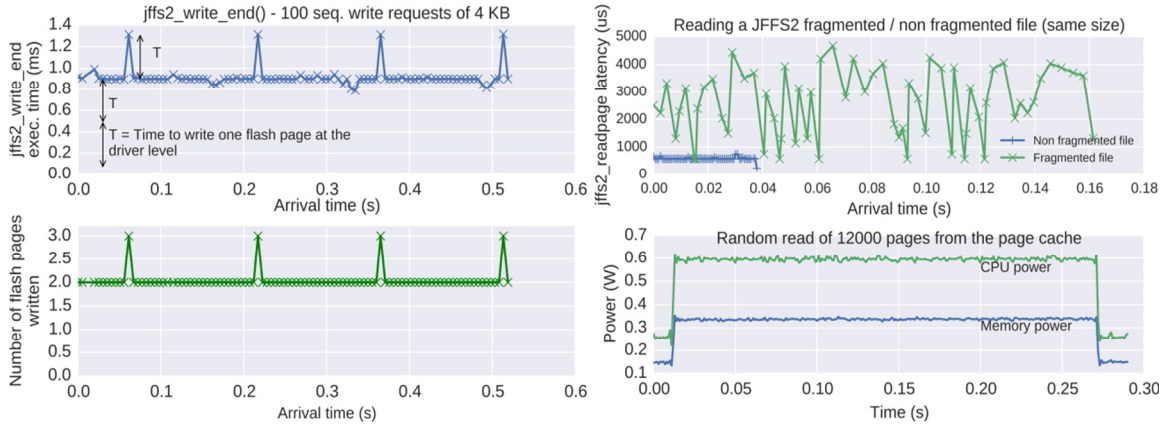


Figure 28. JFFS2 write performance (left), JFFS2 on-flash fragmentation read impact (top right), page cache read power consumptiom (bottom right).

Concerning *jffs2_readpage*, the number of flash pages read can be highly variable. It depends on the on-flash alignment and fragmentation[12] of the JFFS2 nodes that need to be read. It also depends on the cache effect produced by the MTD read buffer. On the right part of Figure 28, one can see the execution time of each call to *jffs2_readpage* when sequentially reading a fragmented and a non-fragmented file. Linux pages belonging to the fragmented file are thus composed of multiple small nodes that are highly scattered on flash space. Read execution times on the fragmented file are significantly large and highly variable because of the number of flash pages read. The total execution time is significantly larger as compared to reading the non-fragmented file.

We also explored the performance and power consumption impact of the GC [22]. JFFS2 online (synchronous) GC impacts write performance when the amount of free space is low. Offline (asynchronous) GC can reduce this impact when the time between the application requests is long enough. Offline GC can also generate some energy consumption when running in background after a write intensive workload producing some invalid data. Flash memory state, in terms of amount of valid / invalid / free pages, defines the GC behavior and thus can have a significant impact on performance and power consumption.

VFS Level. We observed a small overhead for the read and write functions at the VFS level. When the page cache is not involved, flash operations still represent most of read or write operation execution time and energy consumption. The page cache can significantly improve these metrics on read operations. Our experiments showed that when satisfied from the page cache (RAM), applicative read operations are 20 times faster compared to flash. Energy consumption is also reduced by a factor 10. On the bottom right of Figure 28, one can observe the power measured on CPU and RAM chips when randomly reading pages from the page cache. We also explored the impact of read-ahead [20]. Read-ahead was proved to be inefficient on raw flash storage. This is because the MTD NAND driver does not support asynchronous I/O. Nevertheless, read-ahead is active with JFFS2 and in some cases may degrade read performance. Disabling read-ahead at the JFFS2 level can lead to an improvement of up to 70% in performance and power consumption. Read-ahead may highly impact read performance and is enabled with JFFS2, so its functional behavior is modeled in the modeling phase. Concerning the page cache write-back feature, it is only supported by the

[12] We talk about node fragmentation, for instance, when a 2KB of data is spread on 2 nodes in the flash memory rather than written in a single node in one flash page. This is not traditional external fragmentation.

UBIFS FFS. Our experiments showed an increase in performance by a factor 10 to 50 as compared to JFFS2 which does not support this feature.

3.7.2.3 Exploration phase application results

The output of this phase is a list of elements impacting performance and power consumption in an FFS storage system, and a description of how those elements impact these metrics. This knowledge was formalized into models in the modeling phase.

These elements fall into the following categories:

1. The characteristics of hardware components involved in flash storage management (flash memory, CPU, RAM). Flash operation latencies and operating power define the performance and energy consumption of flash operations. CPU and RAM characteristics impact these metrics for OS management algorithms ;
2. The algorithms of I/O software management stack, and multiple RAM caches in this stack. Algorithms and caches are present in various levels: **at the driver level**, MTD allows to perform basic flash operations. A read buffer at this level can impact performance / power consumption in case of high spatial locality. **At the FFS level**, we focused on JFFS2. Linux pages belonging to files are divided into nodes. The on-flash node fragmentation can have a significant impact on read operations performance and power consumption. These metrics are also impacted by GC execution. Small writes are absorbed by a write buffer. **At the VFS level**, the page cache plays an important role as it buffers all file I/O operations and may speed-up future read operations. Related optimization mechanisms also impact performance and power consumption: the read-ahead and write-back features ;
3. The characteristics of applicative I/O workloads: for example, metrics such as spatial and temporal locality determine the cache effect in the management layer, and thus the performance and power consumption of the I/O workload ;
4. The state of the system: It consists in the state of the management stack (content of the RAM caches), as well as the state of the flash memory, in terms of amount of valid / invalid / free space.

3.7.3 Modeling step

The goal of this phase is to formalize the impact on performance and power consumption of previously identified elements into models. In particular, abstract performance and power consumption models are defined then specialized for a given platform through parameters extraction. These models were implemented in a simulator described in the next section.

3.7.3.1 Modeling step overview

The main model types defined are **functional**, **performance** and **power consumption** models. Their interactions are illustrated in Figure 29.

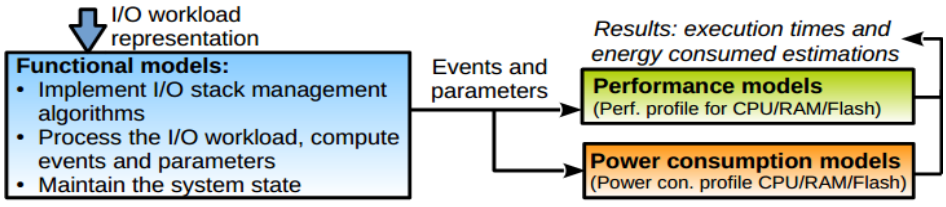


Figure 29. Interactions between different models.

Functional Models. Functional models are a formalized and simplified representation of how applicative I/O workload is processed by the I/O software stack. We provided algorithms for all the relevant mechanisms on the read and write system calls paths, and for all the levels of the I/O stack. We can summarize functional model roles as follows:

- They compute the number of events and related parameters. Events represent important operations during system call processing. An operation can be initiated from one of the I/O stack levels. For example a flash read operation at the driver level has one parameter that is the physical address of the page to read. When an event

occurs, the functional model calls the related performance and power consumption models to obtain an estimation of the execution time and energy consumed ;

- They work on a set of variables and maintain the system state. The system state represents the I/O management stack data structures, in particular the content of the various RAM caches in different levels of the I/O stack ;
- They also compute the number of flash read, write and erase operations realized on the flash component. Operations are computed according to the I/O workload, management algorithms and the system state.

Performance and Power Consumption Models. These models are called by the functional models to effectively compute the execution time and energy consumed for different events. We modeled performance and power consumption for events occurring when processing read and write system calls. This is done for all the levels of the I/O management stack. As part of the write operation processing, GC behavior triggering flash erase operations is also modeled. Processing read and write system calls was modeled as a call stack as depicted in Figure 30. Each level includes some overhead due to system processing in addition to the time taken by flash unitary operations. We can use the same illustration for energy consumed as it denotes the integral of power consumption over a given duration. Thus, we define the total execution time of an operation at a given level as the sum of (1) the execution time of lower levels and (2) an overhead in current level. This statement is also true for the consumed energy.

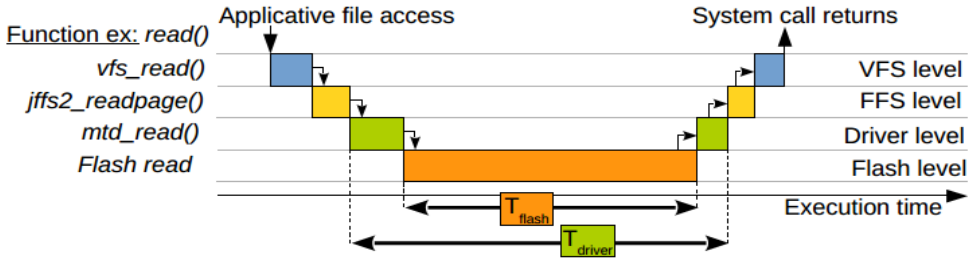


Figure 30. Modeling execution time of a read system call in the I/O stack.

Measuring latencies and energy for flash operations as well as overheads for the upper levels allows to build a performance and power consumption profile for a given hardware/software platform. It is achieved through micro-benchmarks. We call this process **parameters extraction**.

The model decomposition presented on Figure 30 is relatively straightforward for execution times. Concerning power consumption, components such as RAM and CPU do consume energy during the flash operations. In particular, we have noticed in the exploration phase that the CPU is still active during flash operations. Thus, when measuring the energy for the lowest level during parameter extraction, measurements were done not only on the memory chip (RAM + flash on our platform) but also on the CPU.

Of course, functional execution in the I/O stack is more complex than it appears in Figure 30. For example, a cache hit in a given level might not trigger access to the underlying level. Loops might be performed at one level and initiate several operations in a row to the underlying level. To represent this complexity, performance and power consumption models include additional parameters and terms that are presented in the application part of this section.

Other Model Types. A flash **structural model** was built for representing a flash chip. It describes various architectural parameters: the number of pages per block $\mathbf{N_{PagesPerBlock}}$, the total number of blocks $\mathbf{N_{Blocks}}$, the page size $\mathbf{S_{Page}}$, the OOB area size $\mathbf{S_{OOB}}$ and the I/O bus width $\mathbf{S_{IO}}$. For example, a 100 MB partition on the flash chip of our test platform [153] will be represented as: $N_{PagesPerBlock}$= 64, N_{Blocks}= 800, S_{Page}= 2048, S_{OOB} = 64 and S_{IO}= 8.

We also represent in a **flash operational model** the various operations supported by the flash chip(s) and the related constraints. The operational model also maintains the state of each flash page (free or not). Moreover it describes how write and erase operations modify the state of each page. In general, with embedded raw flash storage, only one chip is used, and only legacy (read / write / erase) operations are performed. Thus, our structural and operational models reflect this simplicity. Nevertheless, our model supports complex flash architectures and advanced operations.

Finally, the I/O **workload model** represents one or several applications accessing a set of files. It is a series of timely ordered system calls that the modeled storage system takes as input. In practice, it is a trace file. System calls are modeled with parameters. For example a read system call has the following parameters: a file identifier, a target

address and a size of data to read. The main system calls modeled are read and write. We also modeled open, close, creat, remove, rename, truncate, and sync.

3.7.3.2 Building functional, performance and power consumption models

Building functional models. functional models are built using the information gathered during functional exploration. These are algorithms representing how storage management stack processes an applicative I/O workload. We modeled the way I/O flow is processed in all levels of the stack: subdivision of requests at the Linux page level by VFS, read-ahead algorithm, subdivision of Linux pages in nodes at the JFFS2 level, and flash operations realized through the MTD driver. We also modeled the content and management algorithms for the RAM I/O related caches and their behavior. This includes page cache, JFFS2 write buffer, and MTD read buffer. The page cache is modeled as a simple cache with a configurable size and a LRU eviction policy. The read-ahead mechanism was fully modeled. The JFFS2 GC occurrences and behavior were modeled.

Performance and power consumption models and parameter extraction. In this subsection we describe the performance and power consumption models. They are used to compute the execution time and power consumption for the various events we considered. Concerning power consumption, our embedded platform offers two power measurement points. They are located on the CPU and memory (RAM + flash) chips power rails. Thus, we provide two sets of parameter extraction results, one related to the CPU, and the other to the memory chip.

Driver Level. The MTD driver level is the lowest level considered in our work. We modeled basic flash operations at this level, using constant values for the flash page write and block erase operations execution times and energy [23]. These constants are $\mathbf{T_{MtdWrite}}$, $\mathbf{T_{MtdErase}}$, $\mathbf{E_{MtdWrite}}$ and $\mathbf{E_{MtdErase}}$. For flash page read operation, we used a simple equation to consider MTD read buffer:

$$T_{MtdRead}\,(CacheMiss\,) = T_{MtdReadMiss}\,.\,CacheMiss + T_{MtdReadHit}.(1 - CacheMiss\,) \quad (17)$$

$\mathbf{T_{MtdReadHit}}$ is the time taken by a read operation at the driver level in the case of a cache hit in the MTD read buffer. $\mathbf{T_{MtdReadMiss}}$ represents this time in the case of a cache miss. **CacheMiss** is a Boolean variable that can take one of two values: 0 in the case of a cache hit, and 1 on a cache miss. The value of **CacheMiss** is computed by the functional models that maintain the content of the cache and call the equation each time a read in the driver level occurs. A similar equation was defined for energy.

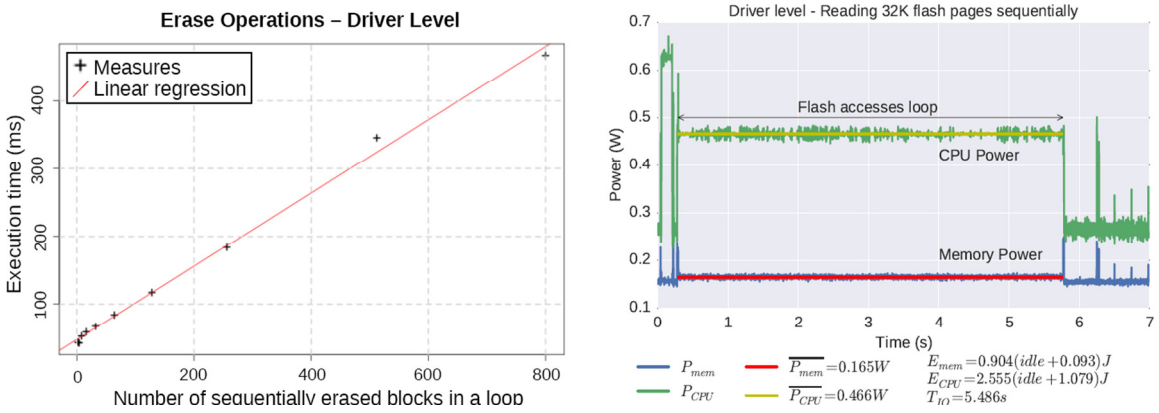


Figure 31. Linear regression applied to find the execution time of one erase operation (left), and mean power consumption of read operations (right)

Parameter extraction for driver level is done with micro-benchmarks. They perform multiple flash operations of the same type in a loop. The execution time of the whole operation is measured. We applied linear regression on the results to obtain latencies for a single flash operation. An example for the erase operation is illustrated on the left part of Figure 31. As one can see, the execution time increases linearly with the number of operations performed. Read and write operations showed the same behavior. As for the power consumption, we measured the mean power observed during each type of flash operation. An example about read operations is illustrated on the right part of Figure 31. The power measured is very stable during flash operations. We used these results to compute the energy cost of a single call

to each type of flash operation for driver level. Parameter extraction results are depicted in Table 10. In this table, one can observe that write and erase operation times are stable, indeed, the GC is implemented at a higher (FFS) level.

Table 10. Driver level parameters extraction results

Performance			Power consumption			
Parameter	**Value**	**R^2**	**Parameter**	**CPU**	**Memory**	**Total**
$T_{MtdReadMiss}$	185.065 μs	0.998	$E_{MtdReadMiss}$	34.6 μJ	2.2 μJ	36.8 μJ
$T_{MtdReadHit}$	4.8 μs	0.999	$E_{MtdReadHit}$	1.4 μJ	0.6 μJ	2.0 μJ
$T_{MtdWrite}$	407.6 μs	0.998	$E_{MtdWrite}$	74.6 μJ	6.3 μJ	81 μJ
$T_{MtdErase}$	536.4 μs	0.999	$E_{MtdErase}$	97.5 μJ	8.5 μJ	106 μJ

FFS (JFFS2) Level. Models for execution time and energy for *jffs2_readpage* are:

$$T_{Jffs2ReadPage}(N_{FlashPages}) = N_{FlashPages}.T_{MtdRead} + T_{Jffs2ReadPageOverhead} \quad (18)$$

$$E_{Jffs2ReadPage}(N_{FlashPages}) = N_{FlashPages}.E_{MtdRead} + E_{Jffs2ReadPageOverhead} \quad (19)$$

$N_{FlashPages}$ is the number of flash pages read during the execution of *jffs2_readpage*. It is computed by the functional models maintaining the state of the system. Functional models also inject the correct value for time and energy of a page read based on the MTD read buffer contents. As one can see, execution time and energy equations have a similar form, thus for the next models we only give the performance equations. Concerning the write functions at the FFS level, the models are:

$$T_{Jffs2WriteBegin}(PcMiss) = PcMiss.T_{Jffs2ReadPage} + T_{Jffs2WriteBeginOverhead} \quad (20)$$

$$T_{Jffs2WriteEnd}(N_{FlashPages}) = N_{FlashPages}.T_{MtdWrite} + GC.T_{Jffs2GcPass} + T_{Jffs2WriteendOverhead} \quad (21)$$

The model for *jffs2_write_begin* is composed of an overhead, $T_{Jffs2WriteBeginOverhead}$, the time taken to read the Linux page if it is not present in the page cache (PcMiss can be 0 or 1, and is computed by the functional model). Concerning *jffs2_write_end*, its execution time is modeled as the sum of an overhead $T_{Jffs2WriteendOverhead}$, the time taken by the flash write operations, and a potential pass of the online GC. $N_{FlashPages}$ is the number of flash pages that need to be written. A value for GC of 1 represents the fact that the GC is launched because there is not enough free flash space to satisfy the current write request. The GC is modeled as follows:

$$T_{Jffs2GcPass}(N_r, N_w, N_e) = N_r.\ T_{MtdRead} + N_w.T_{MtdWrite} + N_e.T_{MtdErase} + T_{Jffs2GcPassOverhead} \quad (22)$$

Here, N_r, N_w, N_e are respectively the number of flash pages read and written, and blocks erased during the GC pass. They are computed by functional models. At the FFS level, parameter extraction consists in measuring the overhead values. We launched micro-benchmarks, performing large amounts of each concerned operation and measured their execution times at the FFS level. For each call to a FFS function, we traced the flash operations and subtracted the time taken at the driver level to obtain the overhead at the FFS level. For each operation type we computed the overhead for several executions and took the mean value. Results on the experimental platform are presented on Table 11. The standard deviation shows that some overheads are stable while others exhibit more significant variations amongst the measurements. Nevertheless, these overheads are very small as compared to flash operation latencies, so, we did not investigate them in detail.

Table 11. FFS & VFS parameter extraction results

FFS performance overhead				
Operation	**value (st. dev.)**	**Operation**	**Value (st. dev.)**	
JFFS2 Write Begin	5.70 (0.7) μs	JFFS2 Readpage	46.80 (14.8) μs	
JFFS2 Write End	54.60 (17.4) μs	JFFS2 GC Pass	348.6 (75.2) μs	
VFS performance overhead		**VFS power consumption overhead**		
Operation	**value (st. dev.)**	**CPU**	**Memory**	**Total**
VFS Read (page cache hit)	39.74 (0.6) μs	6.16 (0.07) μJ	4.37 (0.07) μJ	10.53 μJ
VFS Read (page cache miss)	55.48 (8.3) μs	18.16 (0.7) μJ	11.91 (0.4) μJ	30.07 μJ
VFS Write	29.97 (1.9) μs	16.5(4.1) μJ	8.28 (3.4) μJ	24.78 μJ

In practice, we did not extract parameters concerning power consumption at the FFS level because of technical issues (see [27]). Instead, the VFS energy overhead measured is composed of the overheads of VFS and JFFS2. The overhead of the file system is included in the overhead of VFS, as presented below.

VFS Level. Models for VFS are the following

$$T_{VfsRead}(N_{PHit}, N_{PMiss}) = N_{PMiss} . (T_{FfsRead} + T_{VfsReadMissOverhead}) + N_{PHit} . T_{VfsReadHitOverhead} \quad (23)$$

$$T_{VfsWrite}(N_P) = N_P . (T_{FfsWriteBegin} + T_{FfsWriteEnd} + T_{VfsWriteOverhead}) \quad (24)$$

Read and write functions at the VFS level mainly iterate on Linux pages concerned by I/O requests. Thus, we modeled one overhead per Linux page accessed. N_P is the number of Linux pages concerned by a write request. N_{PHit} and N_{PMiss} are respectively the number of page cache hits and misses during a read request. We modeled two overheads for the read function: one occurring on a page cache hit, and the other on a page cache miss. Indeed, the actions taken are rather different in those two cases. Note that variables such as N_{PHit} and N_{PMiss} are computed by functional models that maintain an internal representation of the page cache content. As for the FFS level, parameter extraction is performed by subtraction. Results are presented in Table 11. Note also that concerning power consumption, the overhead presented is composed of VFS and FFS level overheads.

3.7.4 Simulation and validation phase

We developed a simulator, named OpenFlash, as an execution environment for the previously described models. One of the main goals of OpenFlash is to provide a FFS simulation support. The simulator implements models representing the entire I/O stack of Linux kernel, as well as the hardware components involved. Furthermore, OpenFlash implements the salient features one can find in state of the art flash simulators, such as flash management layers prototyping, background events modeling, FTL systems support, advanced flash architectures and operations. In this section we first give a presentation of OpenFlash. Next we explain how the models and the simulator were validated.

3.7.4.1 Overview of OpenFlash

OpenFlash is a trace-driven, discrete event simulator [136] written in C++. The architecture of OpenFlash is illustrated in Figure 32.

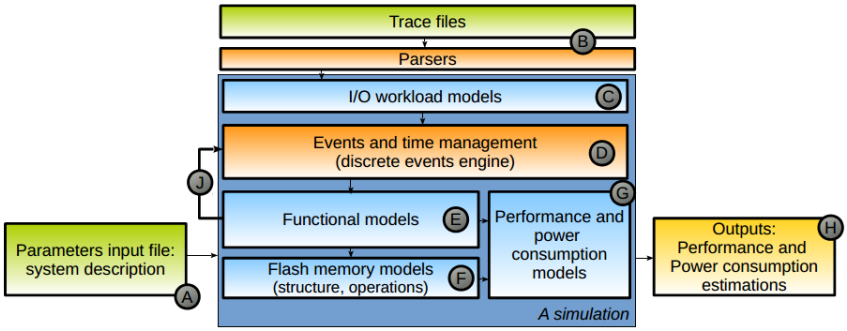


Figure 32. Global architecture of OpenFlash

To run a simulation, the tool first needs a description of the simulated system (A on Figure 32). It also needs a description of the I/O workload applied to the system (B). During a simulation, the trace file is parsed on-the-fly, and events are created according to the I/O workload model (C). The events are processed by a discrete event engine. Events represent applicative system calls, and they are processed by the functional models (E). The functional models compute flash operations (F) and update the system state. The functional models also call the performance and power consumption models to obtain estimations from the multiple sub-operations triggered when processing an I/O event. Finally, various statistics concerning performance and power consumption are provided (H).

I/O events can come from two sources: the input trace, and also the functional models. The ability for the functional layer (FFS or FTL) to inject events between I/O requests coming from the trace pushed toward the design of a simulator that supports background operations such as garbage collection or wear leveling asynchronous operations. For example, JFFS2 background GC is modeled as a periodic event that is regularly triggered and checks for the presence of invalid data. If such invalid data are found, a GC pass is executed. The discrete event engine can be viewed as a simple scheduler deciding about when I/O requests related events and background events are processed.

3.7.4.2 Models and simulator validation

Validation methodology. We validated model estimations at each level of the I/O stack. We used the following methodology: we defined several validation scenarios, descriptions of programs performing I/Os on files. Various parameters of the I/O workload were varied amongst scenarios. These scenarios were translated into (1) a C program compiled and ran on our embedded platform and (2) traces simulated with OpenFlash. We compared real measurements with the simulation outputs in order to determine the estimation error.

We validated power consumption from the higher applicative level (above VFS), and performance from FFS level to validate JFFS behavior.

In addition to micro-benchmarks, we validated performance for a real application. We used SQLite, one of the most popular embedded DBMS, to create a database reproducing the Android OS contact database, as done in [143]. We measured, simulated and validated performance when (1) filling the database (SQL INSERT) and (2) sequentially reading all the previously inserted records (SELECT).

Validation: Power Consumption at the VFS Level. We defined several validation scenarios. We performed read operations on a large JFFS2 file, created on an empty partition. Several parameters impacting power consumption were varied: number of I/O requests, access patterns, activation / deactivation of read-ahead, and data presence in the page cache. The request size was set to 4 KB, the size of a Linux page. This is because read requests are rounded to this granularity in the VFS level. Concerning write operations, we varied the number and size of requests, and the access pattern: appending sequential data to a newly created file and random data updates to an existing file. We measured the total I/O energy consumption for each scenario, and compared it to the simulation estimations.

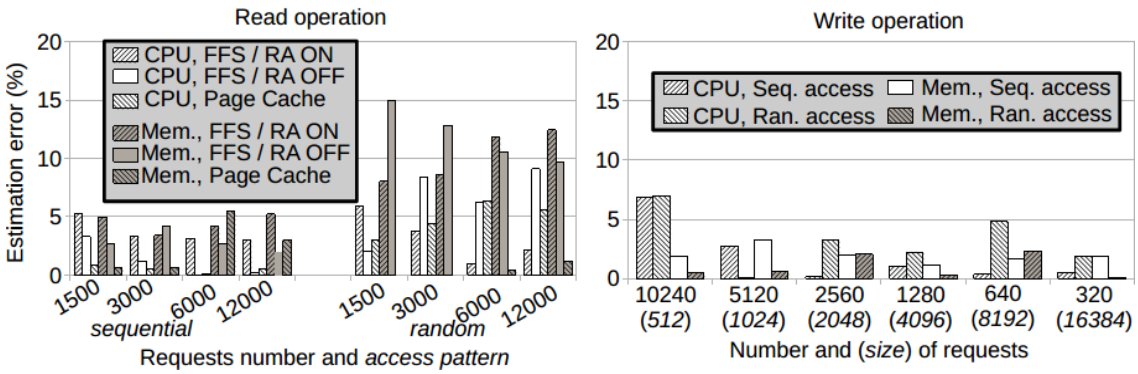


Figure 33. Power consumption estimation error at the VFS level, for read (left) and write (right) operations. Concerning the read results, RA means Read-ahead. Page Cache indicates read operation satisfied from the page cache, and FFS denotes reads served from the FFS.

Results are presented in Figure 33. Concerning read operations, the mean error for all experiments is 4.4%. For sequential operations we consider the estimation precision as being good as all errors are below or around 5%. For random operations, most of the errors are below 10% but a small number of experiments show errors between 10 and 15%. These results concern the memory component, in particular when reading from the page cache. In this case, the energy consumed is very low as compared to the energy consumed when actually using the FFS. Concerning write operations, the mean error for all experiments is around 2.0%. Most of the experiments show errors below 5%, except the experiment with a request size of 512 bytes which has an error of 7%.

Validation: Performance at the FFS and application (VFS) levels. Concerning read operations, we measured and simulated read operations on a large JFFS2 file. We varied the main factors impacting performance of read operations: fragmentation and access pattern (sequential and random). Concerning write operations, we defined 6 validation scenarios [27] by varying: the number and size of requests, access pattern, presence of accessed data in the page cache, and request alignment on Linux pages.

We measured and simulated the execution times of the main read and write functions at the FFS level: respectively *jffs2_readpage* and *jffs2_write_end*. For each test, we compared the real and simulated values for (1) the mean value for *jffs2_readpage* / *jffs2_write_end* execution times and (2) their standard deviation within a test. For write operations, we also traced and simulated the number of flash page write operations realized during each execution scenario, and

compared these values. Results are presented on the left of Figure 34. Note that the standard deviation error presented on the graphs represents the error between the standard deviation of real measures and the one of the simulations.

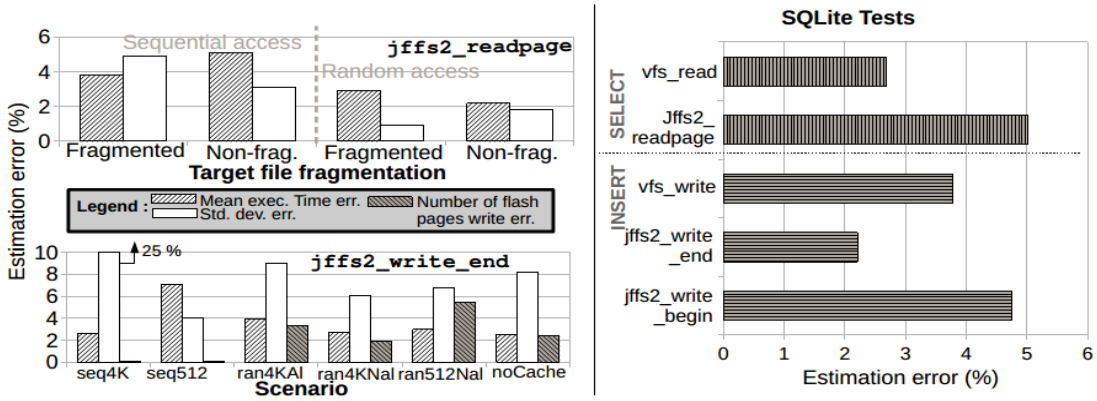


Figure 34. FFS and SQLite tests performance estimation errors.

For read operations, the mean error for all experiments is 3.1%. We consider the precision as very good as all error rates are below 5%. Concerning write operations, the mean error rate for all write experiments is 3.6%. Most of the errors for the mean execution time are below 5%. The errors on the standard deviation are between 5% and 10%, apart from the Seq4K scenario where this error is 25%. For this small request size scenario, the error is large because the standard deviation of the simulation is practically nil, while for the measures it is not. Finally one can notice that the number of flash page write operations estimated by the models is rather precise. Indeed, the error for this value is very small.

Figure 34 shows the results concerning SQLite tests. We measured and computed the estimation error rate for the execution time of the main FFS and VFS function, for both INSERT (write dominant) and SELECT (read dominant) experiments. As one can see, the simulator precision is preserved on real applications as the error stays equal to or below 5%. Note that the INSERT experiments trigger overwrites to the database file, creating invalid data and causing GC execution. The good precision concerning these results also validates our GC model.

3.8 Conclusion

In this study, we presented a methodology for estimating embedded FFS performance and power consumption. The methodology is composed of three consecutive phases. In the **performance exploration phase**, we point out the main elements impacting performance and power consumption in FFS storage systems: characteristics of hardware components (CPU, RAM, flash), I/O processing algorithms, state of the system and attributes of the I/O workload. In the **modeling phase**, this impact is formalized into models of various types. Parameter extraction techniques were developed to build a performance and power consumption profile for a given hardware / software platform. As a case study, we presented a set of models built to represent the entire Linux flash storage management stack using the JFFS2 FFS on an embedded board. In the **simulation phase**, developed models were implemented in a simulator. OpenFlash was validated against performance and power consumption measures in real environment. The average estimation error remains below 10%.

Reusing the developed models is very important and many contributions of the presented work can be used either separately or together. For instance, implementing existing or new FFS algorithms into the simulator should be relatively fast, as a large part of the already developed models could be reused (performance, power consumption, and functional models for VFS and the NAND driver). If one needs to use an FTL rather than an FFS, the interface to do so already exists and a support for FTL integration is integrated in the simulator. In this case, NAND structural models could be reused.

The **applications** of the developed models are numerous; our simulator allows to obtain estimations (performance, energy) about a given workload on a Linux I/O stack with a FFS, but also prototype new flash management layers such as new FFS (for instance UBIFS) or FTL, or test some optimizations to tune the I/O layer.

3.9 Outcome

Advised PhD student	Project	Collaborations	Publications		
			Journals	Int conf.	Others
Dr. Pierre Olivier	ANR OpenPeople*	UBS Lorient (E. Senn)	4	3	5

* The PhD thesis was not financed by the ANR project, but have benefited from the platform and contributed to its development.

4 INTEGRATION OF FLASH MEMORY BASED STORAGE SYSTEMS

4.1 Summary

This section describes our contributions on flash memory based storage system integration. Three subsystems designed are detailed: **C-lash**, a cache system for flash memory storage, **CACH FTL**, a hybrid mapping scheme that relies on flash specific cache mechanisms, and **MaCACH FTL**, an adaptive hybrid mapping scheme.

4.2 Context

Flash memory adoption in different domains is due to many well-known attractive features such as good I/O performance, energy efficiency, shock resistance, size and weight. These attractive features of flash memory come with some limitations designers must deal with in order to maximize both lifetime and I/O performance. Those limitations and constraints have been discussed in the previous section.

The Flash Translation Layer (FTL) is a hardware/software layer intended to overcome the aforementioned limitations, just like previously discussed FFS, it implements the following services: (1) a logical-to-physical mapping scheme, (2) a garbage collector to recycle blocks enclosing invalid pages, (3) wear levelling techniques to balance the wear over the memory cells in order to prevent some memory cells wearing out more quickly than the others. FTLs implement other functionalities such as power-off recovery, bad block management and error correction codes. Buffering systems have also been designed on top of FTLs. They are mainly used to reorganize non-sequential request streams and absorb write operations which contributes in minimizing the number of erase operations in addition to buffering data for future use.

4.3 Problem statement

Even though the objective of both FTL mapping schemes and flash memory buffering mechanisms should be complementary, they have been most of the times designed and built separately. In fact, the objective of studies about FTL mapping schemes is to provide a flash based storage system with good performance in terms of response times while extending its lifetime. This is achieved by minimizing the number of erase operations due to internal FTL mapping scheme management while decreasing the mapping table size.

The buffering system, instead, tries to absorb data updates coming from the host and attempts to group page writes from the same block before sending them to the flash memory. Absorbing data updates allows to minimize the number of write operations as seen by the FTL and so could decrease the number of erase operations thus enhancing performance and lifetime. Grouping pages minimizes the merge operation cost (reading still valid pages from the flash and merging them with new pages) when evicting the group of pages. This generally helps optimizing the overall performance.

The information gap between FTLs and flash memory buffer can lead to performance loss, but also to inadequate problem definition. Indeed, most state-of-the-art FTLs suppose an I/O workload coming from the host while all SSDs contain a cache layer that operates on the host I/O workload and so modifies the I/O pattern before transferring I/Os to the FTL.

4.4 Approach

In this context, we have explored both cache design for flash based storage systems and FTL mapping scheme design to contribute in filling the information gap between the cache and the FTL mapping scheme.

We chose to discuss three of our contributions in the domain of flash memory integration:

- **C-lash**: a cache for flash mechanism with the objective to replace FTL services by a unique cache system that would allow to optimize the performance and enhance the lifetime for some specific sets of I/O workloads.
- **CACH-FTL**: as C-lash was not well performing for all I/O workload patterns, we developed CACH-FTL, a Cache-Aware Configurable Hybrid FTL that uses information about data sent from the cache to perform address mapping efficiently.

– **MaCACH-FTL**: the main issue with CACH-FTL is that parameters were defined offline and thus one given configuration of CACH-FTL could not fit all I/O workload patterns. MaCACH is an adaptive cache-aware hybrid FTL mapping Scheme using feedback control to tune its parameters online according to I/O workload patterns.

The work presented in this section was extracted from the following publications: [ri5][ri9][ri15][rn1][ci17][ci21][ci22].

4.5 Background

FTL mapping schemes depend on the granularity with which mapping is managed. They can be classified into three categories: page, block and hybrid mappings (see Figure 35). (1) Page-mapping scheme maps each logical page into a physical page independently from the other pages from the same block. It is a very flexible scheme and it gives the optimal performance, but requires a mapping table too large to fit into common flash devices embedded RAM. (2) Block-mapping scheme considers the granularity of a block. A logical page address is composed of the logical block number and a fixed page offset inside the block that is not modified by the mapping process. The block-mapping scheme is more feasible in terms of mapping table size; however, its main drawback is that a page update systematically triggers a whole block erase operation and several valid page copies to another block. (3) Hybrid-mapping scheme was proposed to overcome the abovementioned shortcomings by combining both types of mapping: a (small) subset of flash memory space is mapped by page, and the rest is mapped by block. With this scheme, designers try to get as close as possible to the performance of page-mapping scheme with a mapping table footprint close to the block-mapping scheme's.

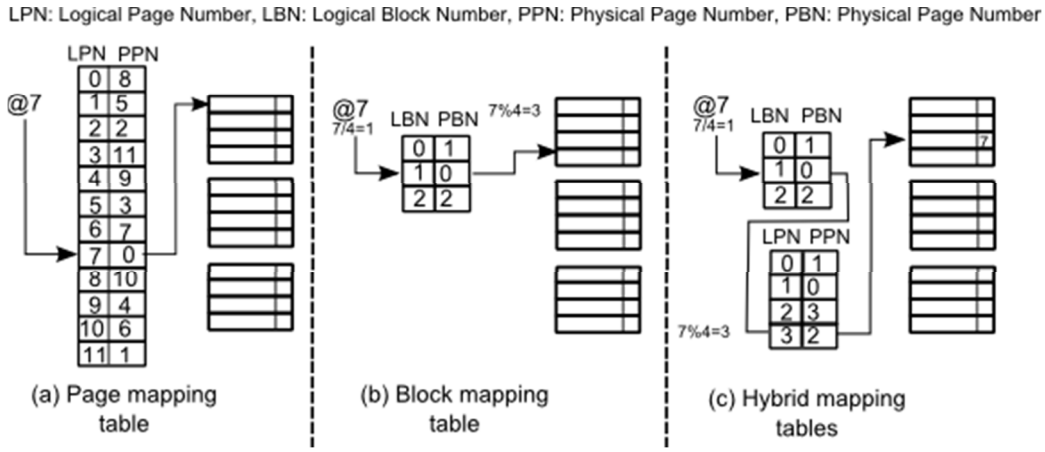


Figure 35. Basic mapping schemes

4.6 Related work

4.6.1 FTL schemes

Most state-of-the-art FTLs are hybrid, the page mapped region (PMR) being limited in order to moderate the PM table size. Many hybrid FTLs try to optimize the use of the PMR dynamically through some I/O patterns based heuristics [176][43][177][178][179][180][181]. These FTLs mainly rely on three metrics: (1) spatial locality, declined in terms of access pattern and I/O request size, (2) temporal locality, and (3) I/O type (read/write).

Most hybrid mapping schemes are based on a primary block-mapping scheme, for data blocks, completed by additional page-mapped log-blocks. Log-blocks are spare flash memory blocks used to avoid a block copy for each page update. In case of a page update, the modification is reported on a page of the log-block rather than performing a whole data block copy operation. A log-block can be either dedicated to one data block (such as M-Systems [188], AFTL [189], CNFTL [190], and BAST [191]) or shared between many (such as RNFTL [192], FAST [40], and KAST [193]). The associativity of pages in log-blocks is also crucial for mapping performance. While first FTLs use log-blocks that are directly mapped to data blocks as in ANAND [188], most recent FTLs are fully associative (FAST,

LAST [41], HFTL [176], BlogFTL [42], MNFTL [195]). In these hybrid FTLs, page-mapping is generally used to map pages of the log-blocks.

In order to cope with the dynamicity and heterogeneity of I/O workloads, many workload adaptive FTLs have emerged such as: CFTL [177], WAFTL [43], AFTL, S-FTL [178], ADAPT [179], HFTL, and Janus-FTL. Those adaptive hybrid FTLs try to reduce and/or optimize the use of the PMR thanks to some I/O patterns based heuristics. These FTLs mainly rely on three metrics: spatial locality (such as WAFTL and ADAPT), declined in terms of access pattern and I/O request size, temporal locality (such as CFTL, AFTL, SFTL, Janus, and HFTL), and I/O type: read/write operations (most adaptive FTLs use this metric to optimize write operations). They decide on the cost effectiveness of data placement on the PMR, based on (at least) one of the abovementioned metrics. Most of these workload adaptive FTLs use a dynamic PMR that grows and shrinks according to I/O workload characteristics.

4.6.2 Flash specific cache systems

In order to optimize the performance of write operations on flash memories, different data cache systems can be placed above the FTL. Most of them reflect the granularity of the erase operation by dealing with groups of pages (FAB [32], CLC [33], BPLRU [34], BPAC [182], LB-Clock [183], PUD-LRU [184], REF [185]). These cache mechanisms try to: (1) maximize the number of flushed pages from a given block, and (2) evict data that are unlikely to be accessed (temporal and spatial locality). To do so, most caches evict the largest set of pages belonging to the same block to reduce the merge operations cost (we will not consider caching mechanisms for meta data such as in [194] [196]).

The information gap between FTLs and flash memory buffer has been already discussed in [186] where the authors proposed to handle this issue by making the cache and the FTL cooperate. This is done through a modification of the cache in order to provide many candidates for flushing data, and on the other hand, the FTL is upgraded by making it achieve a decision on the data to flush according to the cleaning cost. In [201] and [202], the authors presented a very interesting work on an integrated write buffer management scheme for log-block based FTLs that manages log-blocks and the write buffer in coordination for a more efficient and stable buffer/FTL performance. In [203], authors present another relevant cache system integrated into the FTL and that is dedicated to partial updates absorption, thus reducing the number of sustained erase operations.

We propose in this chapter one cache system and two FTLs that impose no modification on the cache and only use information about evicted data from cache. Our design of FTLs tries to cope with the same information gap issue and can be seen as complementary to state-of-the-art proposals.

4.7 Contribution

4.7.1 C-lash, a cache mechanism for flash memory

In this section we describe C-lash (Cache for fLASH) system architecture and policies.

4.7.1.1 C-lash architecture

In C-lash, the cache space is partitioned into two spaces, a page space (p-space) and a block space (b-space). The p-space consists of a set of pages that may come from different blocks in the flash while the b-space is composed of blocks (see Figure 36).

C-lash is hierarchical, it has two levels of eviction policies: one which evicts pages from the p-space to the b-space (G in Figure 36), and another level in which blocks from the b-space are evicted into the flash media (I in Figure 36). With this scheme, we insure that the system always flushes blocks rather than pages to the flash memory. P-space and b-space regions always contain respectively either valid or free pages or blocks.

When a read request arrives, data are searched in both spaces. If we get a cache hit, data are read from the cache (B or E in Figure 36), otherwise, they are read from the flash memory (A in Figure 36). A read miss does not generate a copy into the cache.

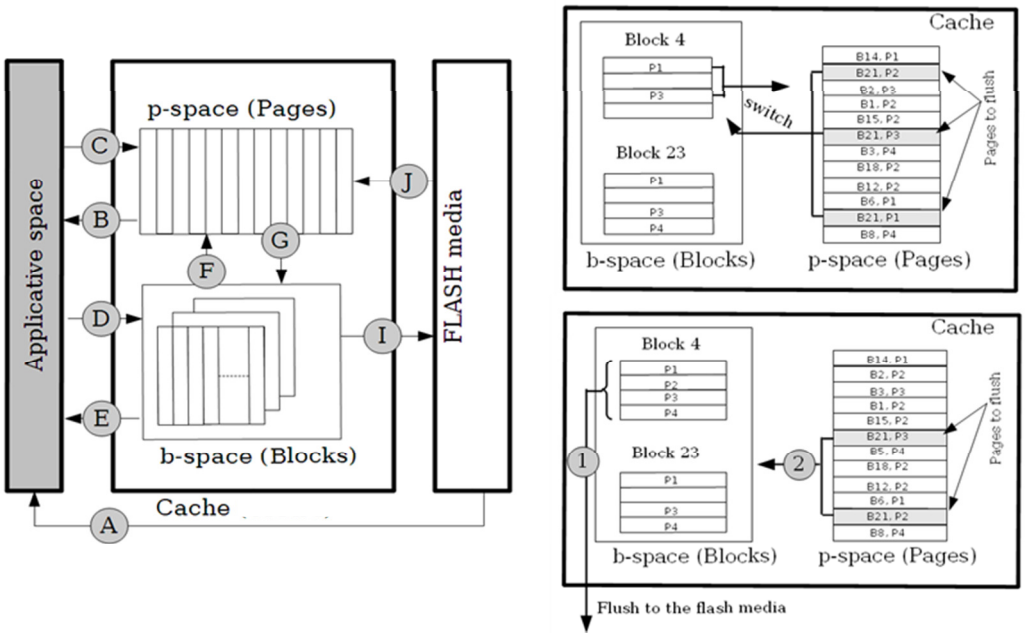


Figure 36. Structure of the C-lash system

When a write operation is issued, in case of a cache hit, data are overwritten (respectively C or D in Figure 36) with no impact on the flash media. If data are not in the cache, they can only be written into the p-space (C in Figure 36). If enough pages are available, we use them to write the data. If not, we choose some pages to flush from the p-space to the b-space (G in Figure 36) and write the new data.

4.7.1.2 Cache policies algorithms

Two eviction policies are implemented (respectively G and I in Figure 36).

P-space Eviction Policy: When a write request is issued and the considered page is neither present in the p-space nor in the b-space, a new page must be allocated in the cache p-space. If a free page is available, the new page is written and data in the corresponding location in the flash is invalidated. If no free space is available, the system chooses a set of pages to evict into the b-space (not into the flash media). Pages to evict are those forming the largest set from the same block. Once this set is found, we have two options:

1. A free block is available in the b-space and so victim pages are copied in it.
2. No free block is available and then, the set of victim pages is compared to the number of valid pages contained in each block of the b-space area:
    a. If there exists a block containing less valid pages than the victim ones, a switch operation is performed (F and G in Figure 36): pages in the victim block are moved in the p-space while the subset of victim pages is moved in the freed block. This induces no impact on the flash memory.
    b. If all blocks in the b-space contain more valid pages than the subset of victim pages, the b-space eviction policy is executed to flush a block into the flash media.

In the upper-right illustration of Figure 36, we describe an example of a p-space eviction. The chosen pages are those belonging to block B21 containing the largest number of pages. One block in the b-space contains two valid pages which is less than the 3 pages to evict. Thus, C-lash switches between the block B4 and the three pages subset. The system, therefore, frees one page without flushing data into the flash.

B-space Eviction Policy: An LRU algorithm is used for this eviction to take into consideration, in the b-space, the temporal locality exhibited by many I/O workloads. When a block eviction is performed, the whole corresponding block in the flash memory is erased before being replaced by the one in the cache.

In case the flash memory still contains some valid pages of the block to evict from the cache, a merge operation (J in Figure 36) is performed. This merge operation consists in reading the still valid pages in the flash memory before flushing the whole block from the cache. The merge operation can be done either during a p-space eviction, we call it *early merge*, or just before flushing the block on the flash media, we call it *late merge*. Early merge is more advantageous if the workload is read intensive and shows temporal and/or spatial locality. If the workload is write-intensive, we would prefer using the late merge. By delaying the merge operation we insure two main optimizations: (1) we read a minimum number of pages because between the moment the pages are evicted into the b-space and the moment they will be evicted into flash, many pages can be written and so invalidated from the flash. (2) Since it is possible for a block in b-space to be moved to p-space during a p-space pages eviction (a switch operation), it may not be worth doing the merge operation too early. This can cause extra read operations.

An example of block eviction is shown in the right part of Figure 36. In the lower illustration, we describe a p-space eviction leading to a b-space flush to the flash media. In the p-space eviction phase, two pages of the B21 block are chosen for eviction. The blocks in the b-space contain more valid pages than the page number to be evicted from the p-space. So, the system needs to evict a block into the flash beforehand. After that, the system copies both pages of the B21 into the freed block. In this specific example, no merge operation occurs because the flushed block is full of valid pages. More details about C-lash can be found in [35].

4.7.2 CACH-FTL, Cache-Aware Configurable Hybrid FTL

In this section, we describe the first FTL we designed: CACH-FTL.

4.7.2.1 CACH-FTL design

CACH-FTL system architecture is illustrated in Figure 37. CACH-FTL splits the flash memory into two regions: (1) an over-provisioning region managed with a page-mapping scheme (called PMR); and (2) a data region managed by the use of a block-mapping scheme (called BMR). The BMR size gives the addressable space for the applicative layer.

CACH-FTL scheme is generic, as it can be used together with any cache system, provided that it flushes groups of pages from the same block. In this section, C-lash was used as a cache system. Using other cache mechanisms proved to have similar results.

.

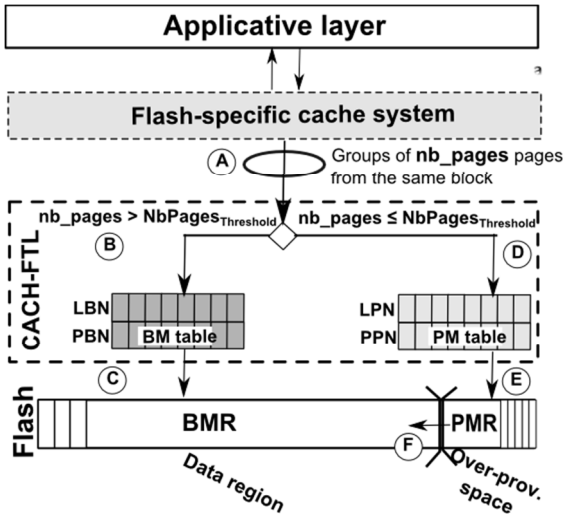


Figure 37. CACH-FTL architecture

Read Operation: In CACH-FTL, when a read request is issued from the application, data are first searched in the cache. If not found, the request is forwarded to CACH-FTL that checks into the PMR first. If data are not found in the PMR, the read request is forwarded to the BMR.

Write Operation: The write operation algorithm is described in Algorithm 1. Write operations always come from the above cache under the form of a group of pages (belonging to the same logical flash block). CACH-FTL defines a

redirection threshold $NbPages_{Threshold}$ (see Figure 37) for the evicted pages flushed from the cache (see Algorithm 1, Functions 1, 2, and 3). Above $NbPages_{Threshold,}$ pages, the group of victim pages is evicted to the BMR (A, B then C in Figure 37). Under $NbPages_{Threshold,}$ pages, the group of victim pages is evicted to the PMR (A, D, and then E Figure 37). In CACH-FTL, the I/O patterns from the host are abstracted to concentrate only on the output of the above cache system. So, when the group of evicted pages is small, CACH-FTL considers that it is more cost-effective to send it to the PMR (avoiding a costly merge operation in BMR that can delay response times). It assumes that pages from the same block would accumulate in the PMR before being moved more efficiently to the BMR by a specific garbage collector.

Algorithm 1. CACH-FTL write algorithms

1: **input**:
2: Redirection page group size threshold: $NbPages_{Threshold}$
3: Number of evicted pages: $NbPages_{Evicted}$
4: The group of pages to evict: $GroupPages_{evicted}$
5: The block in the flash corresponding to data to evict: $Block_{evicted}$
6: Free pages in the PMR: $FreePages_{PMR}$
7: PMR free pages synchronous GC threshold: $SyncFreeP_{Threshold}$ // see the following GC section for details on this variable
8: **FUNCTION 1: CACH-FTL_Write_To_Flash** ($NbPages_{Threshold,}$ $NbPages_{Evicted,}$ $GroupPages_{evicted}$)
9: **if** ($NbPages_{Evicted}$ > $NbPages_{Threshold}$)
10: *Flush_Pages_BMR*($GroupPages_{evicted}$)
11: **else**
12: *Flush_Pages_PMR*($GroupPages_{evicted}$)
13: **end if**
14: **FUNCTION 2: Flush_Pages_BMR**($GroupPages_{evicted}$)
15: **if** ($GroupPages_{evicted}$ <*NumberOfValidPagesInABlock*)
16: //If there are still valid pages → Read valid pages in flash from the BMR and/or PMR
17: Read ($Block_{evicted}$ - $GroupPages_{evicted}$)
18: Erase $Block_{evicted}$ // erase the block in the flash memory
19: Flush the block from the cache → $Block_{evicted}$
20: Update page-mapping tables (PMR if needed and BMR)
21: Delete $GroupPages_{evicted}$ from the cache
22: **else**
23: Erase $Block_{evicted}$
24: Write the $GroupPages_{evicted}$ → $Block_{evicted}$
25: Update the block-mapping table of BMR
26: Delete $GroupPages_{evicted}$ from the cache
27: **end if**
28: **FUNCTION 3: Flush_Pages_PMR**($GroupPages_{evicted}$)
29: **if** ($FreePages_{PMR}$ - $GroupPages_{evicted}$ < $SyncFreeP_{Threshold}$)
30: // space available in PMR is < $SyncFreeP_{Threshold}$
31: *Activate Noninterruptible GC*
32: *PMR_GC*($NbPages_{Eviceted}$) // see Algorithm 2.
33: **end if**
34: Flush $GroupPages_{evicted}$ → PMR
35: Update the page-mapping table PMR

Algorithm 1, Function 2 shows how large groups of pages flushed from the cache are dealt with. The main contribution of CACH-FTL is a hybrid mapping scheme, neither the wear leveling, nor overall flash memory GC were implemented. For the sake of our study, as one can see from lines 17, 18, 19, 23, and 24 of Algorithm 1, a simple direct (block) mapping in-place update algorithm was used: when updated, a given data block is written at the same location. This is considered as the worst case for CACH-FTL.

4.7.3 BMR & PMR specific garbage collectors (GC)

In Function 3, line 29, algorithm 1 tests whether there is enough free space in the PMR to flush the small group of pages. Actually, if the free space falls under a given limit, the GC is launched (see line 32). Otherwise, the set of pages are flushed to the PMR and the mapping table is updated.

In fact, CACH-FTL uses two distinct garbage collectors: (1) a garbage collector to recycle free space within the PMR named PMR-GC, and (2) a garbage collector that flushes pages from the PMR to the BMR, named BMR-GC. Both GCs are described in Algorithm 2[13].

PMR-GC of CACH-FTL: The PMR-GC is launched when the free space in the PMR falls under a predefined threshold. It uses a simple greedy reclamation algorithm that selects the physical block from the PMR containing the least number of valid pages. Valid pages are copied to a free block, and then an erase operation is triggered to recycle the victim block (Algorithm 2 Function 4 and 5). In brief, PMR-GC recycles invalid pages in the PMR by compacting valid pages.

BMR-GC of CACH-FTL: The BMR-GC is launched in case the PMR-GC could not find physical blocks containing invalid data. BMR-GC uses a greedy reclamation algorithm selecting the largest group of PMR pages belonging to the same data block (in BMR). After the pages are found, they are merged with the valid pages of the corresponding block in the BMR. Once done, the PMR pages are invalidated. Note that the BMR-GC does not recycle free space in PMR, but it invalidates some pages that can be spread on different PMR blocks. In order to recycle free space, the PMR-GC is launched after a BMR-GC pass.

Figure 38 shows an example of a PMR-GC and BMR-GC operations.

Algorithm 2. CACH-FTL Garbage Collection algorithms

input:
  Number of evicted pages: $NbPages_{Evicted}$
  The group of pages to evict: $GroupPages_{evicted}$
  The block in the flash of data to evict: $Block_{evicted}$
  Free pages in the PMR: $FreePages_{PMR}$
  PMR-GC free page asynchronous threshold: $AsyncFreeP_{Threshold}$
  PMR free page synchronous GC threshold: $SyncFreeP_{Threshold}$
  The PMR block with the largest number of invalid pages: $Block_{MaxInvalid}$
  Number of invalid pages in $Block_{MaxInvalid}$: $NbPages_{Invalid}$
  Maximum number of pages in PMR belonging to the same data block: $MaxGroupPages_{PMR}$
// FUNCTION 4 is launched at the end of each I/O request
FUNCTION 4: PMR_Garbage_Collector()
if ($FreePages_{PMR} \leq AsyncFreeP_{Threshold}$)
  Activate Interruptible GC
  PMR_GC($NbPages_{Evicted}$)
else
  Wait for next PMR write request
end if
FUNCTION 5: PMR_GC($NbPages_{Evicted}$)
while ($NbPages_{Evicted} > 0$)
  for i ←1 **to** *sizeOfPMRinBlocks*
    Find $Block_{MaxInvalid}$
  end for
  if (no $Block_{MaxInvalid}$ found) //no invalid pages in PMR
    BMR_GC()
  else
    Find $NbPages_{Invalid}$
    Read valid pages from $Block_{MaxInvalid}$
    Copy $Block_{MaxInvalid}$ → freePMRBlock
    Update page-mapping table
    Erase $Block_{MaxInvalid}$
  end if
  $NbPages_{Evicted} = NbPages_{Evicted} - NbPages_{Invalid}$
end while
FUNCTION 6: BMR_GC()
for i ←1 **to** *sizeOfPMRinBlocks*
  Find $MaxGroupPages_{PMR}$
Read all pages $MaxGroupPages_{PMR}$ → specific block in the cache
Read valid pages of the same block as $MaxGroupPages_{PMR}$ *from BMR*
Invalidate data from the PMR and BMR mapping tables

[13] The GC that recycles free space from BMR was not investigated.

41: Erase the data block in the BMR
42: Flush the whole block of $MaxGroupPages_{PMR}$ from the cache and update the mapping table

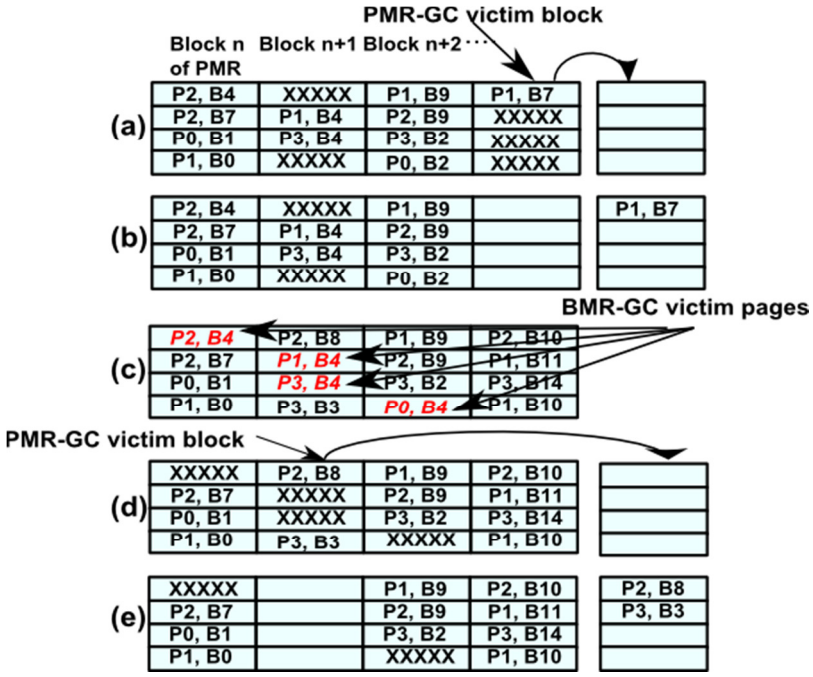


Figure 38. Examples of PMR and BMR GC: each page of PMR can belong to a given data block (for instance, in the top left, Block n of PMR contains Page 2 of data block 4). In (a), when a PMR-GC is launched, CACH-FTL chooses the block with the maximum number of invalid pages (marked "XXXXX"). It copies the valid pages to a new block and erases the original one, which finishes as State (b). In (c), the PMR is full. A BMR-GC is launched, and the system chooses the largest group of pages belonging to the same block; in this example, B4 (four pages). It then copies those pages to the BMR and invalidates them (State (d)). Once done, the PMR-GC is launched to free a block. It then chooses the second block (State (d)), copies the two valid pages and erases the first block (State (e)).

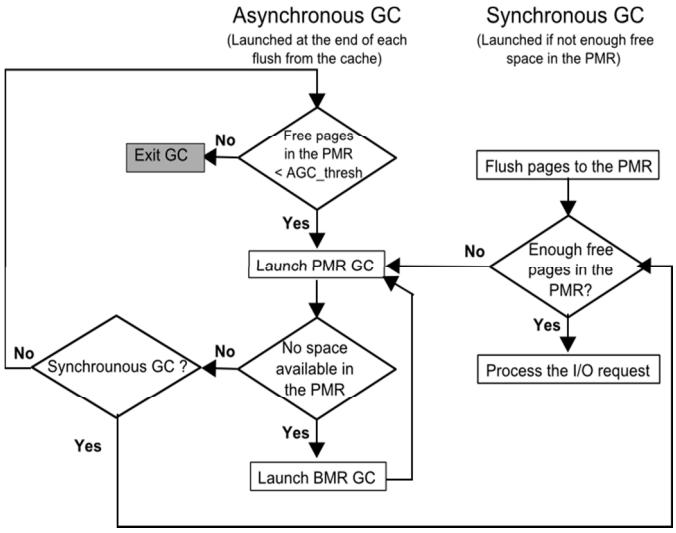


Figure 39. CACH-FTL synchronous and asynchronous GC processing

PMR-GC and BMR-GC Asynchronous Design: As it can be observed in Figure 39, PMR and BMR GCs have a synchronous and an asynchronous implementation. Synchronous GC is launched whenever no free space is found on the PMR to fulfill a write request from the cache. If the PMR cannot recycle free pages, it launches BMR-GC to copy the set of pages from one block from the PMR to the BMR then it triggers back the PMR-GC to recycle invalidated pages. During asynchronous GC, the GC can be interrupted to satisfy an upcoming I/O request. Asynchronous GC is

launched whenever the free space falls under a given threshold (named AGC_thresh in Figure 39) provided that I/O timeouts are sufficient for a PMR-GC sub-operation. Asynchronous GC operations are divided into atomic steps. Indeed, asynchronous GC can be interrupted between these atomic phases, without losing data consistency between the cache, PMR and BMR. More details about CACH-FTL can be found in [37][38] .

4.7.4 MaCACH: a workload adaptive CACH-FTL design

4.7.4.1 Motivation and assumptions

MaCACH design originates from the following observation: many adaptive hybrid FTLs do not explicitly use a fixed size PMR (implemented in over-provisioning space) even if they consider it as bounded. Indeed, the used PMR size evolves dynamically according to I/O workload characteristics. This contributes in wasting flash memory space as the unused flash over-provisioning space is invisible from the applicative point of view. In our opinion, in many cases, it would be necessary to have an FTL that adapts to the fixed over-provisioning space size that hosts the PMR. Indeed, over-provisioning space in SSDs can be configured but is bounded and fixed in operation [205].

In CACH-FTL, we think it is relevant to make the most of the PMR by maintaining a high utilization ratio. We make the assumption that the more the PMR is occupied the higher the probability for finding large groups of pages from the same block to flush to the BMR. This would enhance the efficiency of BMR-GC.

MaCACH uses feedback control theory coupled with an asynchronous GC to stabilize the PMR fill rate at high values. MaCACH is a workload adaptive upgraded version of CACH-FTL. It uses a Proportional-Integral-Derivative (PID) controller to tune redirection threshold according to the PMR fill rate. We remind that this threshold determines if pages are evicted to PMR or directly to BMR. If the PMR free space size is high, the redirection threshold value is increased so that the PMR receives larger groups of pages. Conversely, if the PMR free space size is low, the redirection threshold is lowered so that the PMR receives smaller groups of pages (thus destaging data to the BMR) while the asynchronous GC is activated to recycle free space in the PMR.

4.7.4.2 PID feedback control on the PMR

In MaCACH, the regulation of the PMR utilization ratio is achieved using a discrete Proportional-Integral-Derivative (PID) controller. With its simple structure, characterized by low memory and computational complexity requirements, the PID controller is the most common type of feedback control.

The discrete PID controller can be described by the following equation [204]:

$$u(k) = u(k-1) + K_p \left[\left(1 + \frac{K_i \Delta t}{K_p} + \frac{K_d}{K_p \Delta t} \right) e(k) + \left(-1 - \frac{2K_d}{K_p \Delta t} \right) e(k-1) + \frac{K_d}{K_p \Delta t} e(k-2) \right] \quad (25)$$

where $u(k)$ and $e(k)$ represent, respectively, the required redirection threshold value and the difference between the controller set point (ideal PMR free space rate) and the PMR free space rate level at instant k, Δt is the sampling time (see Figure 40). K_p, K_i and K_d represent the traditional gains of the PID controller (i.e. proportional, integral and derivative gains). Note that $u(k)$ should have a strictly positive integer value. For this sake only the integer part of the PID output is considered.

Stabilizing the PMR free space rate in steady state requires finding good values for the PID gains. The tuning of these interdependent parameters is generally a subjective and time-consuming procedure. In this way, we rely on the Ziegler-Nichols method [204], which has proven to be efficient for many problems [206]. With this method, the PID gains can be expressed as follows:

$$\begin{cases} K_p = 0.2\, K_u \\ K_i = \frac{2\, K_p}{T_u} \\ K_d = \frac{K_p\, T_u}{3} \end{cases} \quad (26)$$

Where K_u and T_u represent, respectively, the ultimate gain, making the output oscillating with a constant amplitude, and the oscillation period.

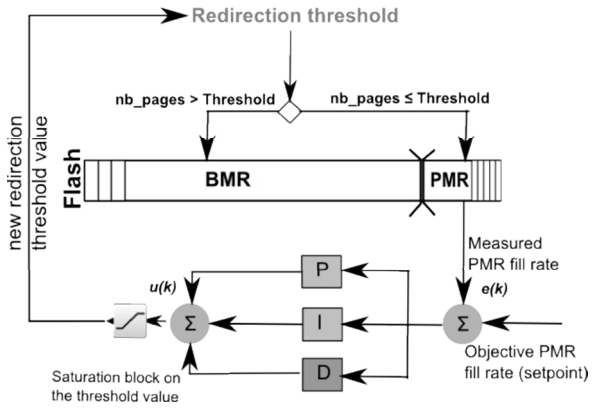


Figure 40. MaCACH PID

Some different values of the gain K_u were tested to see how the choice of this parameter impacts the performance. This is discussed in the results and discussion section.

A saturation block, a component imposing higher and lower bounds, was added to the output of the controller block described above in order to limit the value of the redirection threshold of MaCACH (see Figure 40). In fact the value of the threshold should vary between one and the maximum number of pages contained in a block (in the performance evaluation part, it is equal to 64). However, for performance sake, those extreme values were not used as they put too much pressure on one flash region. The choice of the maximum and minimum values of the saturation block is subject to discussion in the results section.

In MaCACH, the update frequency is event based and the Δt parameter is reported according to the I/O request arrival times at the FTL (MaCACH) level. We varied this parameter and studied its impact.

4.7.4.3 PID controller and MaCACH GC

The PMR is considered as an in-flash buffer that is filled when it receives evicted pages from the above cache system. The redirection threshold gives the maximum number of pages that can be flushed at once from the cache. It represents, for a given I/O trace, the maximum write throughput on the PMR (number of pages per eviction). The PMR is emptied through the BMR-GC process that moves pages from the PMR to the BMR.

The GC and the PID controller work together to reach two main objectives: (1) keep the PMR as full as possible; (2) avoid synchronous GC that has a dramatic impact on response times.

In MaCACH, some optimizations have been applied to the GC mechanism of CACH-FTL (see Figure 41). The asynchronous GC has been modified in order to reduce the total amount of erase operations performed. In fact, in the initial design, when the PMR free space falls under a given threshold, the PMR GC was launched. Once initiated, the PMR GC generated an erase operation even when there was only one page to recycle. In MaCACH, we inserted some efficiency related constraints. As one can see in Figure 41, the optimized asynchronous GC begins by launching a BMR-GC to invalidate some space in the PMR. Then, it checks if the PMR can recycle a minimum number of pages (*nb_pages_thresh*) in one pass. If not, it runs the BMR-GC again. In fact, in asynchronous mode, the PMR-GC will only be launched if it can recycle a minimal number of pages in one pass, thus satisfying an efficiency-related constraint. This lower bound value, *nb_pages_thresh*, can have a significant impact on the number of erase operations generated.

The preceding modification was not applied on the synchronous GC. In fact, the priority in this case is to decrease the time taken by the GC by getting a sufficient space for writing data rather than increasing its efficiency (maximum space in minimum time) as the GC response time in this case will be added to the I/O request response time. Thus, in the synchronous implementation of the GC, the system first launches a PMR-GC to recycle invalid pages from the PMR (see Figure 41). The BMR-GC is only triggered if no space is available.

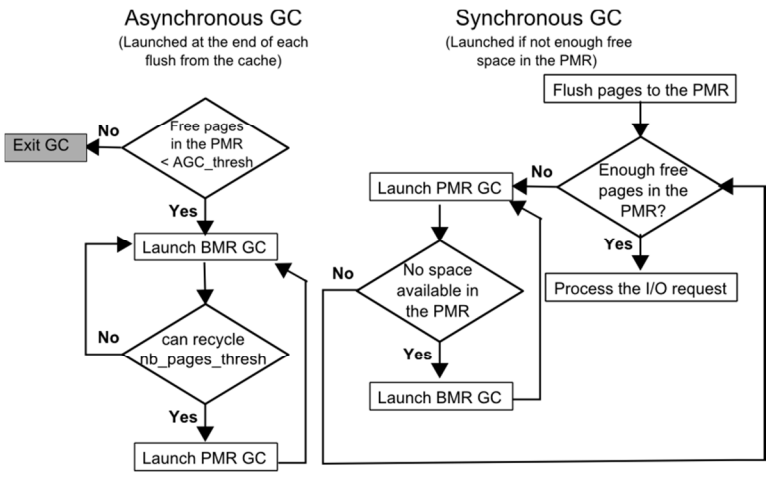


Figure 41. MaCACH GC synchronous and asynchronous processing

4.7.4.4 Parameters of MaCACH

MaCACH behavior depends on the calibration of a set of parameters. Those parameters are mainly related to the PID controller and the GC. The percentage of reserved flash over-provisioning space is also a parameter but it can be seen as a constraint. Its limit is SSD dependent but it can be generally configured by the user.

The main parameters related to the PID controller are:

- The PID set point: the percentage of PMR size free space objective;
- PID update frequency: the frequency at which the controller recalculates the threshold value in terms of number of I/O requests to MaCACH;
- PID K_u and T_u: representing, respectively, the ultimate gain and the oscillation period;
- PID saturation block upper and lower bounds (for the redirection threshold);

The main GC related parameters are:

- Asynchronous GC threshold: it determines the percentage of free space in the PMR under which the asynchronous GC must be launched;
- Asynchronous GC frequency: the frequency at which the system checks the asynchronous GC threshold;
- Asynchronous GC minimal benefit: the number of candidate pages to recycle from the PMR under which PMR GC is not launched (noted *nb_page_thresh* Figure 41).

The tuning of those parameters is studied in the results and discussion part.

4.7.5 Performance evaluation methodology of C-lash, CACH and MaCACH

In this performance evaluation part, we try to partially cover the three proposed mechanisms: C-lash, CACH-FTL and MaCACH. A modified version of the FlashSim [145] simulator was used, based on Disksim [208] discrete event simulator, the most popular disk-based storage simulator. FlashSim implements some FTL schemes: FAST, DFTL schemes and an idealized page based FTL.

We have modified and upgraded the functionality of FlashSim to simulate a dual cache subsystem placed on top of the flash media. In addition, we have developed other FTLs to compare with our schemes, this includes an optimized block mapped FTL, and a workload adaptive FTL named WAFTL.

We mainly relied on two performance metrics: the average response time, and the number of performed erase operations. Other metrics, more specific to the tested scheme, will be detailed afterwards. In our study, the response

time is captured from the I/O driver point of view, including all intermediate delays: caches, controllers, I/O queues, etc. We tried to minimize the intermediate elements' impact to focus on the flash memory subsystem behavior. The second metric we captured was the number of performed erase operations. It gives an indication of the wear out of the memory.

4.7.5.1 Performance evaluation of C-lash

We simulated a NAND flash memory with a page size of 2KB and a block size of 128KB (+4KB for metadata). The three operations latencies have the following values: a page read: 130.9µs, a page write: 405.9µs and a block erase: 2ms. This configuration is based on a Micron NAND flash memory [216]; its values are representative of the read/write/erase performance ratio.

As C-lash was meant to replace FTL schemes, in the performed set of tests, we compared one configuration with different FTLs: FAST, DFTL (a very efficient page mapped FTL) and the idealized page map. As pointed earlier, page map FTL consumes a huge amount of memory but gives ideal performances as compared to other FTLs, we used it as a baseline. The chosen C-lash configuration has 2 blocks (128KB each) and 128 pages which constitutes a total small size of 512KB. The hybrid (block and page) mapping table used with C-lash is very small as compared to the cache size; for a 512KB cache, its size is about 320 bytes. Simulations were performed with synthetic and real workloads (see Table 12). Spatial locality in this table defines requested data that are neighbors but not strictly contiguous. Two requests, R1 and R2, are supposed to be neighbors if the distance between the starting address of R1 and the one of R2 is under a given threshold. All the performed simulations begun with the same initial state, the flash media was supposed to be completely dirty (or at least the addressed area). Each new flash write in a given block generated an erase operation.

Table 12. Synthetic workloads tested with C-lash, FAST, DFTL and page map, the used size is 1GB.

Sequential Rate	**Request Number**	**Inter-Arrival Times**
0%, default value (0→100%) steps of 5%	60000	(0, 200 ms) default value
Spatial locality	**Write Rate**	**Mean Request Size**
0%, default value (0→100%) steps of 5%	80%, default value	4 pages default value (1, 2, 4, 8, 16 , 32, 64, 128)

4.7.5.2 Performance evaluation of CACH-FTL

As CACH-FTL was developed after C-lash, we have updated the NAND flash memory configuration used, the simulated NAND flash memory characteristics were as follows: a 4 KB page size and a 256 KB block size. The three basic operations had the following delays: 25 µs for a page read, 200 µs for a page write and 1.5 ms for a block erasure. Those numbers are related to a real flash storage system [210]. The chosen cache size in all the tests was 2 MB (the simulated flash memory address space was less than 8 GB for real traces). For this study, the cache system was configured and fixed to six blocks in the b-space and 128 pages in the p-space (256KB*6+4KB*128).

A fixed default configuration for CACH-FTL was used for the first set of tests. It had the following characteristics: a redirection threshold of four pages and an over-provisioning space (PMR) representing 10% of the flash space (the configuration can be optimized according to the workload). Similarly, FAST was set to use 10% for over-provisioning space for log-blocks (PM and BM do not use over-provisioning space).

CACH-FTL was compared with three other FTL schemes: (1) page-mapping (PM); (2) an optimized block-mapping scheme (BM); and (3) FAST hybrid FTL. The same cache system (C-lash in our case) with the same configuration was placed on top of each of the tested FTLs.

Both real and synthetically generated I/O workloads were considered. We focus on real traces in this document, for the complete study, one can refer to [38]. For real traces, some widely used I/O traces available from the Storage Performance Council (SPC) were chosen. These workloads describe traces of Online Transaction Processing (OLTP) applications obtained from two financial institutions [210]. Another tested trace subset was Cello99 [211], which is issued from the activity of a workgroup file server used in HP labs. For Cello99, eight-day traces were selected from five different disks. For real traces, see Table 13.

Table 13. Financial 1, Financial 2 and Cello99 I/O trace characteristics. Note that MSRC was only tested with MaCACH

Number format: Mean, (min, max)	Financial 1 (24 volumes)	Financial 2 (19 volumes)	Cello99 (5 volumes)	MSRC (2 volumes)
Write rate	77%, (4%, 100%)	18%, (0%, 98%)	34%, (19%, 53%)	(81%,60%)
Sequentiality per request/per page	23% / 46%, (1% , 99%)	9% / 38% (3%, 96%)	9% / 45% (4%, 27%)	<6% / >90% (for both)
Mean req. size (KB)	5.6	5.3	4.5	9, 22.7
Trace time (hours)	12	12	168	168

4.7.5.3 Performance evaluation of MaCACH FTL

We evaluated the performance of MaCACH through a two steps procedure: (1) a performance comparison of MaCACH with six FTLs: an optimized block mapping FTL (OBM), an ideal page mapped FTL (PM), DFTL [44], FAST, CACH FTL, and WAFTL. (2) An evaluation of the impact of the most relevant parameters on MaCACH performance.

The same flash memory configuration as the one of CACH-FTL was used. A fixed default configuration for MaCACH was used for the first set of tests. It had a PID set point and asynchronous GC of 5% (free space in the PMR), a PID update frequency and an asynchronous GC frequency of one I/O request, PID Ku of 0.04, PID saturation block upper and lower bounds for the redirection threshold of 24 and 8 respectively, and an asynchronous GC minimal benefit of 8 pages. The PMR percentage values for MaCACH were chosen to be equal to 5% and 10%. Similarly, FAST was set to use 5% and 10% for over-provisioning space for log-blocks (PM and BM do not use over-provisioning space).

In addition to real traces tested for CACH-FTL, we used MSRC traces from [212] by selecting two different disks, one from a research server project and another from user home directory, both have sizes of about 16GB (see Table 13). We also used synthetic traces for the second step consisting in evaluating the impact of different MaCACH parameters, see Table 14.

Table 14. Synthetic I/O trace parameters

Sequential rate	**Request number**	**Inter-arrival times**
30%	50000	exp (0, 200ms)
Spatial locality	**Write rate**	**Request size**
10%	100%	1 page (4KB)

4.7.6 Results and discussion

4.7.6.1 Evaluation of C-lash

In this section, we briefly describe the results obtained when comparing our cache for flash solution with the following FTLs: FAST, DFTL and an idealized page mapping FTL. As stated earlier, the pure page mapping FTL is very RAM consuming and is not really usable, but it gives the best FTL performance. In this section, in addition to the mean response time and the number of erase operations, we also evaluated weighted standard deviation of the erase operations distribution over the flash memory blocks to express the quality of wear leveling.

Sequentiality

As we can see in Figure 42, C-lash performs better than DFTL, in terms of response time, if the sequentiality rate is above 40 %. It always performs better than FAST. It achieves less erase operations if the sequentiality rate is above 50% and always outperforms FAST. Thanks to the dual cache eviction properties of C-lash which help absorbing the erase operations, C-lash always perform a better wear leveling than both FAST and DFTL. The higher the sequentiality rate, the more C-lash approaches the ideal page mapping FTL performance for the three performance metrics.

Spatial Locality

We can draw the same conclusions concerning spatial locality. In terms of response time, C-lash always achieves better performance than FAST, it performs better than DFTL if spatial locality is above 10%, and performs even better than page mapping FTL if spatial locality is greater than 85%. Concerning the number of erase operations, C-lash takes advantage of its buffering mechanism to reveal sequentiality and so performs better than DFTL for more than 20% spatial locality, and better than page map FTL for more than 85% spatial locality.

We can observe that the weighted standard deviation increases for 100% spatial locality. For the three FTLs, this is mainly due to the very high temporal locality making the same data blocks to be very frequently modified. For the case of C-lash, the number of erase operations is too small (write/updates absorbed by the cache) to consider the case of 100% spatial locality. This case is not representative of real workloads.

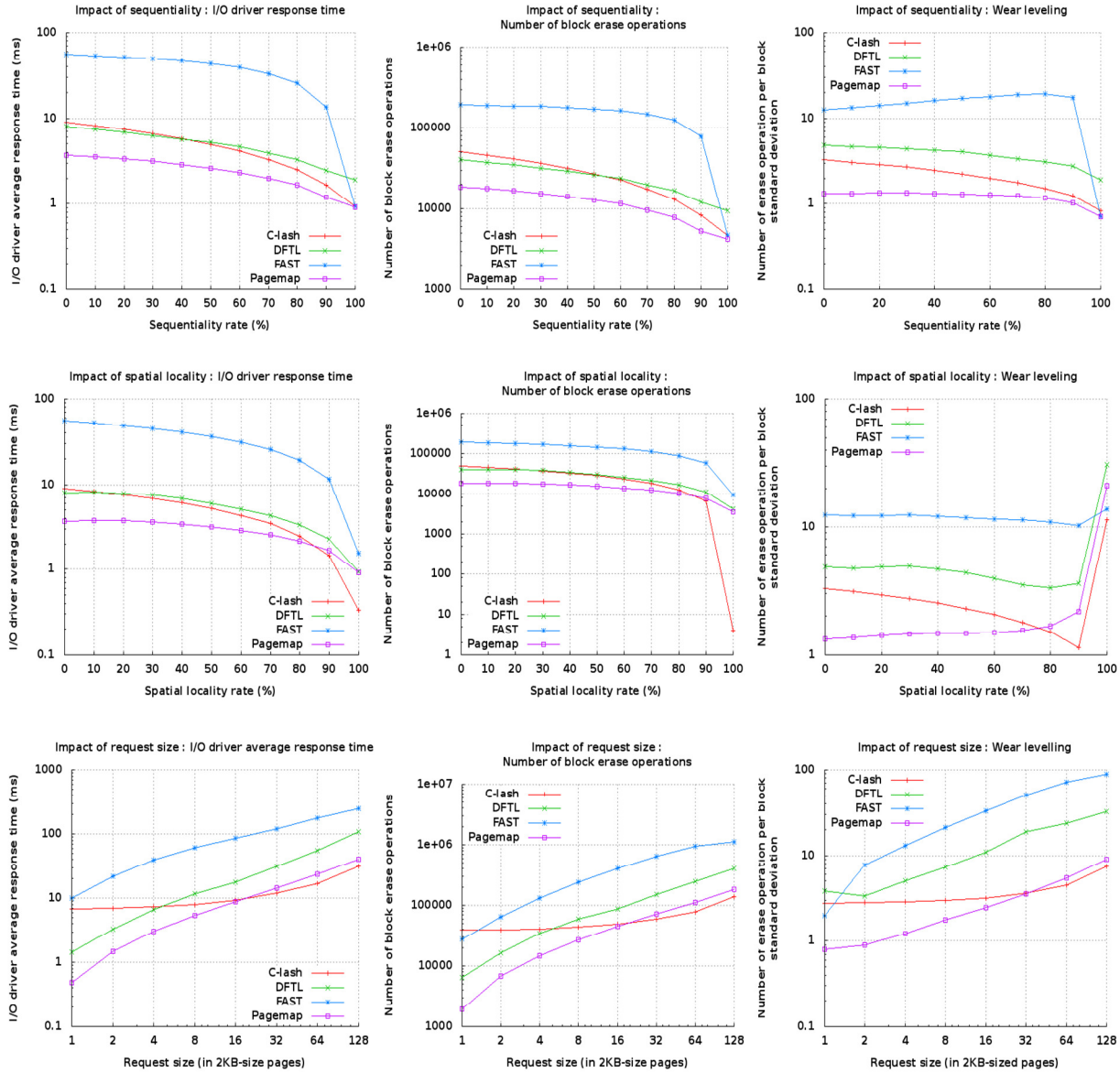


Figure 42. Performance evaluation for C-lash according to sequential rate, spatial locality and request size.

Request Size

These tests were performed with 0% sequentiality and 20% spatial locality. We see that for request sizes greater than 6 pages (of 2KB), C-lash mean response time is better than DFTL's. C-lash even performs better than the page mapping

FTL for request sizes greater than 16 pages. We can observe the same behavior for the number of erase operations. We can also notice that the slope of both response times and number of erase operations curves for the three FTLs is steeper (logarithmic scale in the figure) than the one of C-lash. In fact, C-lash takes more benefit from the increasing request sizes. The larger the request size, the less the system has to perform costly merge operations. In fact, the request size is just another facet of sequentiality. The larger the request size, the more contiguous pages are accessed, and so, better is the throughput.

We demonstrated throughout this section that C-lash could replace FTL services (mapping, wear levelling and garbage collection) for highly sequential workloads. C-lash can be, for instance used in many multimedia appliances where data are accessed sequentially.

4.7.6.2 Evaluation of CACH-FTL

This section describes the results of the tests conducted for CACH-FTL evaluation.

CACH-FTL versus PM, BM and FAST

Figure 44-a shows the mean response time per request for CACH-FTL as compared with BM, PM and FAST. One can observe that for the chosen configuration, CACH-FTL approaches the performance of the ideal PM in most cases and always performs better than BM and FAST. For Financial 1 traces, CACH-FTL improves BM by 47% and FAST by 71%. For Financial 2, CACH-FTL improves BM by 65% and FAST by 58%. Finally, for the Cello99 trace, CACH-FTL improves BM by 86% and FAST by 38%. The average differences between the ideal PM and CACH-FTL are: 39% for Financial 1, 21% for Financial 2 and 35% for Cello99. Thus, CACH-FTL drastically improves response times and is closer to the ideal PM case for the tested I/O workload.

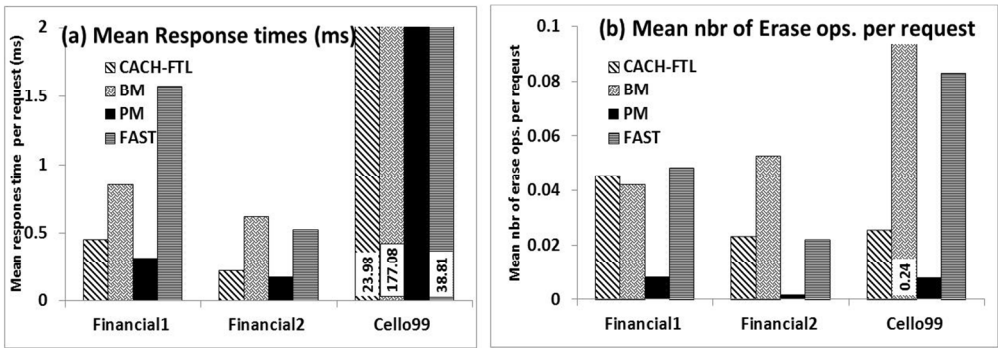


Figure 44. Performance of CACH-FTL. (a) The mean response times for the studied traces; (b) The number of generated erase operations.

One can see that the tested CACH-FTL configuration generates 8% more erasures than BM for Financial 1 and 6% less than FAST. For the Financial 2 trace, CACH-FTL performs 55% better than BM and 5% less then FAST. Finally, for the Cello workload, CACH-FTL performs 89% better than BM and 69% better than FAST. For all erase operation results, one can observe that PM performs, by far, better than the other FTLs.

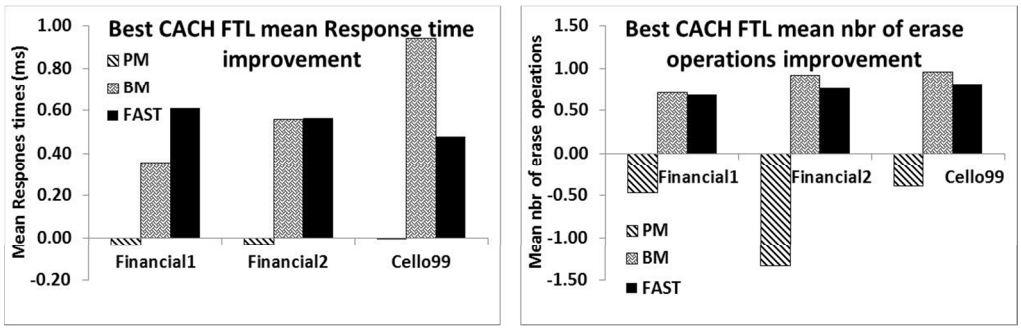


Figure 45. CACH-FTL's best configuration improvement over BM and FAST.

Figure 45 presents the best results that CACH-FTL can give when varying the redirection threshold from 1 to 32 and the over-provisioning space from 5% to 25% of the total flash memory space. Those two parameters were varied for each single disk of each trace, and the best configuration was chosen for each disk. For FAST FTL, the log-block space was varied accordingly. The main observation one can draw from the mean response times figure is that CACH-FTL performs approximately as good as PM, the ideal FTL, with a difference of 4%, 3% and zero for Financial 1, Financial 2 and Cello, respectively. CACH-FTL improves the performance of BM by 36%, 56% and 94% for Financial 1, Financial 2 and Cello, respectively. Finally, CACH-FTL enhances the performance of FAST by a factor of 61%, 56% and 48% for Financial 1, Financial 2 and Cello, respectively.

Concerning the mean number of erase operations, one can observe that the best CACH-FTL configuration always performs better than BM and FAST by more than 69%. One can infer that this would prevent the flash memory from wearing out quickly, as the total number of erase operations is drastically reduced. However, compared to PM, CACH-FTL performs very poorly as it generates much more erase operations. This is mainly due to the additional number of erase operations generated because of the garbage collection mechanism. As this mechanism is asynchronous in CACH-FTL, it does not always impact response times. One must also keep in mind that PM uses approximately 80% more memory for storing the mapping table than CACH-FTL (with a 25% over-provisioning space, and the difference is higher when the over-provisioning space is smaller).

CACH-FTL adaptability: redirection threshold and over-provisioning space configuration

Figure 46 depicts the I/O performance variation of CACH-FTL with different configurations for both redirection threshold and PMR size, compared to FAST for three sample disks (one from Cello99, one from Financial 1 and one from Financial 2). For nearly all the tested workloads (around 50 for the real I/O traces), simulations showed that there is always at least one CACH-FTL configuration that surpasses, or at least matches, FAST for a given over-provisioning space size. Figure 46 (a-1) and Figure 46 (c-1) shows that even though CACH-FTL gives very bad performance when poorly dimensioned (very small values of the BMR/PMR redirection threshold), it is capable of optimal performance. In Figure 46 (a-1), the optimal configuration of CACH-FTL is given for high values of the redirection threshold and it improves FAST performance by 27%. In Figure 46 (c-1) and (b-1), the best CACH-FTL performance is also reached by high values of the threshold (32) and improves it by 68%, and by 90% for Figure 46 (b-1). Optimal performance depends highly on the applied workload, as it can be observed in Figure 46. The same conclusions can be drawn based on the number of erase operation curves in Figure 46 (a-2),(b-2),(c-2). The optimal points are always given by CACH-FTL for all the tested disks. Note that the best performances are not always achieved at high threshold values and high over-provisioning space.

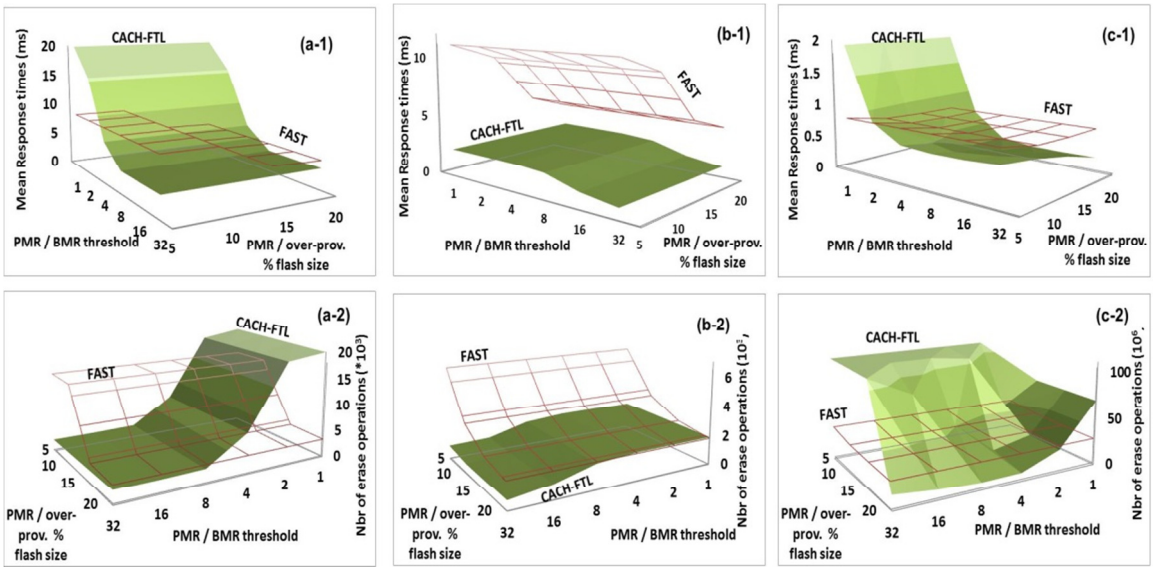


Figure 46. CACH-FTL and FAST performance comparison on three different volumes of Financial and Cello real traces. We varied the over-provisioning space for FAST and the PMR region and threshold for CACH-FTL. The curves show one volume from the Cello workload (a) and two volumes from the financial traces (b, c). Both response times and the number of erase operations are illustrated.

CACH-FTL offers the user a large design space, providing flexibility that allows one to compromise between the performance and lifetime of the flash memory (the number of erase operations performed). It is from the observation of Figure 46 that came the idea of designing an adaptive FTL (MaCACH) in order to avoid cumbersome and complex manual tuning according to I/O workloads.

4.7.6.3 Evaluation of MaCACH

Comparing with other FTLs

Figure 47 shows a comparison between MaCACH, PM, DFTL, OBM, FAST, CACH FTL and WAFTL in terms of mean response time reported to the size of a flash memory page (total response time divided by the total addressed size in terms of flash memory pages of 4KB) and the mean number of erase operations also reported to a page size. For both FAST and MaCACH, we used two configurations for the over-provisioning PMR space: 5% and 10%.

As one can observe from Figure 47-a and Figure 47-b, OBM proved to be the less good performing FTL from mean response time and number of erase operations points of view. Indeed, each cache flush operation systematically resulted in one erase operation and a merge operation between the flushed pages and the still valid pages in the written flash block.

MaCACH and PM: As one can observe from Figure 47-a, MaCACH performs very well as compared to ideal PM from the response time point of view with a maximum difference of 15% in favor of PM and a mean of 6%. For the mean erase operation metric, MaCACH performs better than most of the tested FTLs, except PM, which gives much better results thanks to its flexibility.

MaCACH and DFTL: MaCACH performs better than DFTL in term of response time for all the tested workloads regardless of the size of PMR (5% or 10%). The observed average improvement of MaCACH on response times with 10% PMR as compared to DFTL was 10% for Cello, 24% for both Financial 1 and Financial 2 workloads and an average of 40% for MSRC traces (63% and 19% for Rsrch and Usr respectively). Concerning the mean number of erase operations, MaCACH (10% of PMR) for Financial 1 performs the same as DFTL while for Financial 2, Cello and MSRC, DFTL outperforms MaCACH as it produces a smaller number of erase operations. In fact, the tested MaCACH configuration was identified to generate more erase operations for a small number of workloads having large inter-arrival times. During those timeouts, MaCACH performs many asynchronous GCs to empty the PMR which generates many erase operations as compared to DFTL. Another point is that, as discussed earlier, MaCACH writes blocks in the BMR *in-place* as no wear levelling was implemented. This generates an extra erase operation for each write in the BMR while in DFTL and PM, this never happens as pages are written out-of-place.

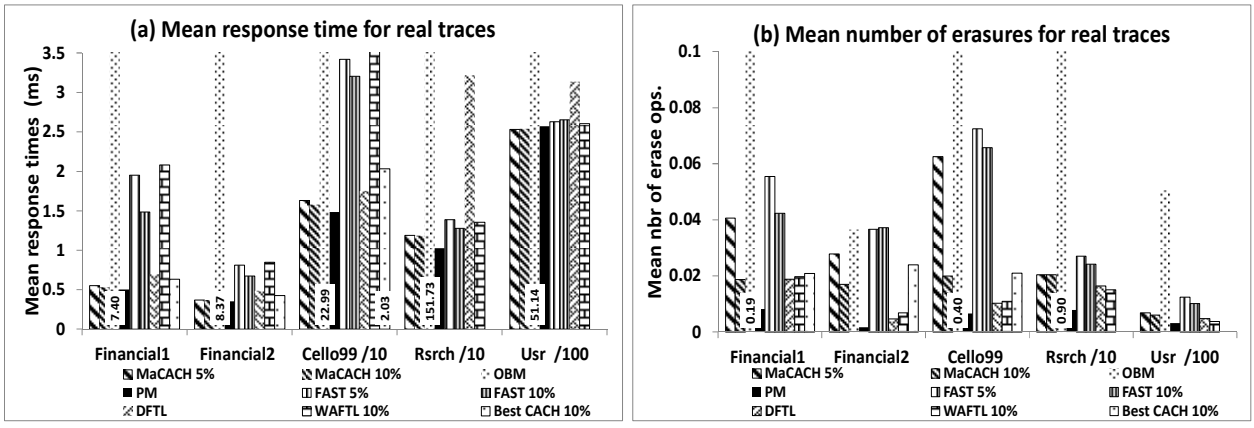


Figure 47. Performance of MaCACH. a) Mean response times for the studied real traces reported per flash memory page. b) Mean number of generated erase operations reported per flash memory page. For reasons of clarity, some results are outside the scale of the graph, values are reported on the figure.

MaCACH and FAST: MaCACH enhances the mean response time of FAST by more than 50% for all the tested cases whatever the size of the over-provisioning space (71%, 54%, 52% for Financial 1, 2, and Cello respectively for 5% over-provisioning space, and 64%, 45%, 51% for the same traces and 10% over-provisioning space) except for MSRC traces for which the enhancement was about 10%. MaCACH also reduces the generated number of erase

operations by 26%, 23%, 13% and 35% for 5% of over-provisioning space, and 55%, 54%, 69% and 28% for 10% of over-provisioning space for Financial 1, 2, Cello and MSRC respectively. In fact, MaCACH makes a better use of the over-provisioning space thanks to the PID feedback control.

As one can observe in Figure 47-a, response times of MaCACH do not decrease dramatically when increasing the over-provisioning space as MaCACH optimizes efficiently response times even with low over-provisioning space. This is achieved thanks to the asynchronous GC.

MaCACH and CACH: MaCACH shows better mean response times than the best static configuration of CACH for all workloads (16%, 13%, and 22% for Financial 1 and 2 respectively). On the other hand, for the number of erase operations MaCACH performs always better as it reduces the best CACH configuration mean number of erase operations per request to 9%, 29%, and 4% for Financial 1, 2, and Cello respectively. This enhancement is mainly due to the modification made on the asynchronous GC coupled with the adaptive threshold behavior. Indeed, for CACH, with a given volume, the threshold cannot be modified. Note that the Best CACH configuration is unrealistic as it supposes to have a different configuration for each volume, which is unfeasible in the static version.

MaCACH and WAFTL: WAFTL was configured with a buffer zone (page mapped) of 10% of the flash memory size. More exactly, it was configured according to the workload address space (the same as MaCACH). MaCACH shows better response times as compared to WAFTL. It presents a performance enhancement of 74%, 58%, 10% and 7% for Financial1, Financial 2, Cello and MSRC workloads respectively. However, for the number of erase operations MaCACH shows better performance only for Financial1 workload (5% less). While for the other workloads, WAFTL performs better. One of the reasons behind this is that we implemented the original WAFTL in which block updates are performed out of place, while they are performed in place in MaCACH (in which we did not integrate a wear leveler).

However, from the performance point of view, MaCACH is more efficient for a given overprovisioning space. In fact, for WAFTL to be more efficient, one needs to have a very large buffer zone as shown in the original paper. In addition, in WAFTL, once data are mapped (to block or page mapped region), if the access pattern changes, data cannot migrate from a zone to another. While in MaCACH, if a previously sequentially accessed set of pages are accessed randomly, they are put in the PMR. MaCACH thus shows more workload adaptability.

MaCACH configuration discussion

Two main sets of parameters allow tuning MaCACH: parameters related to the PID controller, and those related to the GC. In this performance evaluation part, we focused on the synthetic trace described in Table 14.

The main parameters related to the PID controller are: the PID set point, the PID update frequency, the PID K_u, and the PID saturation block upper and lower bounds.

The GC parameters are: the asynchronous GC threshold, the asynchronous GC frequency, and the asynchronous GC minimal benefit.

Varying the controller gain: Figure 48 shows the variation of the controller output of MaCACH for two different values of K_u. One can observe the initial filling of the PMR space marked A in Figure 48. Since the PMR is initially (after the warm-up) ~8% empty, the PID controller fixes the threshold to its maximal value, 24 for this experiment, in order to flush the maximum number of pages to the PMR. The PMR free space drops abruptly and we can observe that the PID lowers the threshold to its minimal value (fixed by the saturation block to 8) to destage written pages to the BMR while launching the asynchronous GC to free PMR space. Phase B is the steady state phase where we can observe that the controller and the asynchronous GC succeed in maintaining the PMR free space to 5%. Phases C and D are samples of Figure 48 that show the difference in behavior with different gains. In fact, the higher the gain K_u, the more aggressive is the PID in increasing and decreasing the threshold value. Phase C represents the case where the PMR free space amount is above the threshold while phase D represents a PMR free space value being under the threshold. One can observe the update of the threshold value, which is more aggressive in case of a higher gain K_u. In this special case, both gains achieve PMR free space stability thanks to the asynchronous GC.

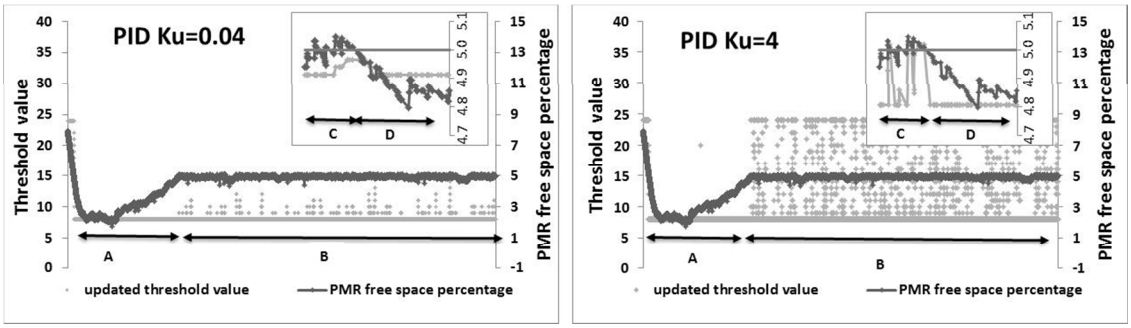


Figure 48. The threshold variation according to PMR free space percentage for different values of the gain Ku for a PID setpoint of 5%

Varying MaCACH update frequencies: MaCACH behavior depends on both the PID and the asynchronous GC update frequencies. In order to have a reactive system, one may choose to update the system at each I/O operation. This is not always necessary and depends on the applied I/O workload.

Table 15. Varying the update frequencies in MaCACH

Asynchronous GC update frequency	PID update frequency	Mean response time (ms)	Number of erase operations
1	1	0.66	48062
10	10	16.86	84454
100	100	18.44	84939

Table 15 shows an example of the impact of varying the update frequencies. Update frequencies are expressed in terms of number of I/O operations. In the case of I/O workloads described in Table 14, increasing the update frequency from 1 to 10 generates a very high performance drop. Indeed, during the 10 flush operations for which neither the asynchronous GC nor the PID monitor the system, the PMR could be filled thus generating a high number of synchronous GC operations. As seen earlier, more synchronous operations means higher response times and less efficient GC. It can be seen from Table 15 that maximizing the system performance requires a reduced frequency interval (i.e. up to 1 update per request).

Varying the controller set point: The PID set point, in terms of free space in the PMR, should be set carefully. If the value is too high, then the PMR will not be fully used, thus lowering the chances to get efficient GCs. If set too low, there is a risk to perform synchronous GC in case of write operations burst from the cache.

Another critical issue is the difference between the PID set point and the asynchronous GC thresholds. A PID set point different from the asynchronous GC threshold means that the PMR free space amount will permanently oscillate between those two values. We chose to set both parameters to the same value in order to have both mechanisms work to reach a steady state around the configured PMR free space percentage.

Table 16. Varying the controller setpoint

Asynchronous GC threshold (% of free space)	PID set point (% of free space)	Mean response time (ms)	Number of erase operations
2	2	0.8	47767
5	5	0.66	48062
10	10	0.66	48879

Table 16 shows an example of varying the above discussed parameters. One can observe that the lowest value gave the higher mean response time due to generated synchronous GCs. But still, the number is not very high as the PID and asynchronous GC have worked efficiently (the update frequencies were set to". On the other hand, one can observe that the number of erase operations is higher for large values. This is due to a larger number of write operations generated (GC).

Other parameters of MaCACH are important to consider. The saturation block configuration gives a lower and a higher bound for the threshold. For the lower bound, too small values mean that if the PMR free space is above the

threshold, the system accepts to flush a very low number of pages on the BMR. This might incur some dramatic performance drops. Conversely, for the higher bound, the value should not be too high as this could provoke eviction of full blocks in the PMR if the latter contains too much free space. This would increase the write amplification phenomenon, thus, in a certain measure, the number of erase operations.

Tuning the asynchronous GC minimal benefit parameter can also help in optimizing the global performance. A high value reduces the number of PMR GC operations, but increases the chances to trigger synchronous GC as invalid space in the PMR is less frequently recycled. On the other hand, with a too small value, the PMR GC is launched more frequently, thus generating more erase operations.

We have also performed a study on the configuration space exploration of MaCACH. To do so, we have used multi objective tools to investigate the best configuration that reduces the number of erase operations and maximizes the performance for a set of workloads. This is detailed in [39].

4.8 Conclusion

In this section we have presented three mechanisms for a better integration of flash memories in storage systems, one flash specific cache mechanism and two FTLs (CACH and MaCACH).

C-lash was proposed as an alternative to FTLs for I/O workloads presenting high sequential rates. C-lash takes into account the cost of different flash memory operations (read, write and erase) in terms of access time and lifetime by flushing onto the media only blocks containing the maximum number of valid pages. This tends to severely reduce the number of erase operations and so improves the performance. C-lash takes also into account the temporal and spatial locality of data with its b-space LRU eviction policy. We showed that simple caching mechanism can outperform complex FTLs in many simple cases. In addition, this caching mechanism could be implemented in software or in hardware (controller's memory).

CACH-FTL was a first tentative FTL toward reducing the information gap between the FTL and the cache above and the need to better use storage architecture by exploiting some information coming from the cache to manage the hybrid mapping in a simple and efficient way. This was first done by taking into consideration the output of the cache and not the one of the host (above the cache). CACH-FTL supposes a common feature of most flash specific caches that flush data by groups of pages and try to optimize page eviction according to this metric. Some characteristics of CACH-FTL are: (1) **genericity**, as it can be used with any flash-specific cache system, provided that it flushes groups of pages; (2) **flexibility**, as it offers a large configuration space, for efficient tuning of I/O performance (response time and lifetime) according to application needs; this is achieved by modifying PMR size and the redirection threshold; (3) **scalability**, as CACH-FTL can work with whatever cache memory/flash storage ratio, as one can adjust the size of the mapping table accordingly (reducing the PMR size); (4) **efficiency**, as CACH-FTL uses a small RAM amount to store the mapping table: with four to five times less memory usage for a PMR, we have a drop in performance of less than 5% when the configuration is well chosen.

MaCACH-FTL is a workload adaptive version of CACH FTL. It uses a PID controller. MaCACH relies on the double action of a PID feedback control and an asynchronous GC to keep the PMR space utilization stabilized on a high value. The objective is to maximize the efficiency of the PMR and BMR GC by recycling the largest possible set of pages from the same block in order to reduce the cleaning cost. MaCACH makes the decision on writing a group of pages in the PMR or in the BMR dynamically according to the space utilization of the PMR thanks to the PID controller. Using the PID controller in conjunction with an asynchronous GC to stabilize and optimize the utilization of the PMR could be transposed to some other hybrid FTLs if we consider that the PMR region should be limited.

4.9 Outcome

Advised PhD student	**Project**	**Collaborations**	**Publications**		
			Journals	**Int conf.**	**Others**
None	None	UBO (P. Olivier, S. Rubini, and L. Lemarchand)	4	4	1

Part 4: Current projects and future work, Research Area 3

1 INTRODUCTION

In the current data centric context, it is becoming crucial to optimize the performance of I/O storage systems. Indeed, the storage subsystem is the historical bottleneck because of continuing growth of performance gap with processing units.

As discussed earlier, flash memory-based storage systems came to reduce the gap between existing processing units' performance and storage systems. They have clear advantages in terms of performance and energy consumption. However, their cost and the world's high demand for storage let us think that hard drives are not expected to disappear in the near future. Therefore, it is relevant to consider hybrid storage systems containing both types of devices: hard disks and flash based devices, and see how to better exploit this heterogeneity at the applicative and system levels.

It is in this context that I initiated my work on research area 3. The major topic tackled in this area is how to take full advantage of storage system heterogeneity for different types of applications. I am mainly working on two application domains in this context:

1. **Cloud infrastructure**.
2. **Database systems**.

This section presents ongoing work on both topics in addition to a discussion on long term perspectives. This part is divided into three chapters:

- **Chapter 1** presents ongoing work and perspectives related to storage systems in a Cloud context.
- **Chapter 2** summarizes some of our contributions and perspectives on Database systems and flash memory.
- **Chapter 3** presents some long term perspectives about emerging Non-Volatile Memories (NVM), other than flash.

In the different chapters of this part, the used notations are not always homogeneous and some variables can have different names in different chapters. In fact, they are left as they appeared in the related published papers so that the reader can easily refer to the latter for more details.

Note that all the work presented in this part is an ongoing work and as a matter of fact, is not finalized. We plan to carry on this work in the near future as many new challenges have been identified. They are discussed in the last section of each chapter.

2 ONGOING WORK AND PERSPECTIVES ON CLOUD STORAGE SYSTEMS

2.1 Summary

This chapter describes our work regarding storage systems in the Cloud. We will focus on IaaS and DBaaS (Database as a Service) Cloud services. We relied on a MAPE-K autonomic approach (detailed later in this chapter). To this end, we developed monitoring frameworks, cost models for performance and energy consumption evaluation, and optimization techniques for data placement on hybrid storage systems. The achieved work on Cloud hybrid storage is mainly related to the PhD theses of Mr. Hamza Ouarnoughi and Mr. Djillali Boukhelef, while the ongoing work discussed in the conclusion is related to PhD theses of Mr. Jean-Emile Dartois and Mr. Islam Naas.

2.2 Context

Two main ideas have been briefly introduced in previous sections: (1) the explosion of data volumes for both private and professional use, and (2) the move toward data-centric applications and the need for processing power. Those issues could not be thought of without having energy efficiency in mind. In fact, data centers in the US consumed more than 1.5% of the energy generated and this ratio is expected to increase by 18% each year [217]. The storage system part in energy consumption is estimated to represent between 20% and 40% of the total energy consumption in a typical data center [218]. Other studies gave different figures, for instance, DRAM main memory subsystem consumes between 30% and 50% of an HPC (High Performance Computing) node, even though this ratio depends on DRAM capacity and configuration [218]. Indeed, one of the most challenging technological innovations for Exascale computing systems planned for 2020 is power consumption management and optimization [67]. In fact, green computing and energy efficiency became rightly very important issues that were confined to embedded systems and that are now generalized to all (or most) systems. To illustrate this trend, since 2010, most of the top 10 supercomputers of the TOP500 increased their computational rates while keeping their power consumption lower than earlier estimates [67][219]. To partly reduce energy costs, big companies such as Microsoft, Google and Yahoo have built new data centers near to large and cost efficient power sources. In fact, data centers are more and more driven by the management of their limited energy budget [218].

The cost of energy is a growing concern for data center operators. It can represent half of the total cost of ownership [221]. Actually, a data center consists mainly of electrical equipment, cooling equipment and IT equipment each of which represent approximately an energy overhead of 22%, 25% and 50%, respectively [222]. IT equipment consists of servers for data processing, data storage and networking devices for communication. Cooling devices (also called CRAC (for Computers Room Air Conditioning) are responsible of keeping the data center at a constant temperature, they do so by extracting the heat and controlling the temperature and humidity levels. An approximate distribution of peak power usage in IT equipment is shown in Figure 49.

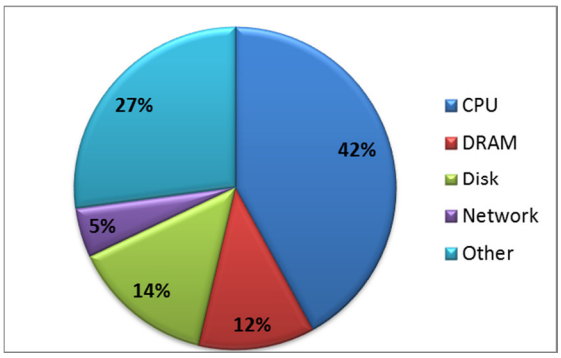


Figure 49. Approximate distribution of peak power usage by hardware subsystem in one of Google's data centers (late 2012) [222]

In has been noticed in some studies [223] that data storage devices can be responsible of about 25%-35% of data center power consumption (including their cooling).

The traditional approach for building high performance and capacity storage systems consists in using arrays of disk drives and distributing data across the disks to provide parallel I/O along with error detection/correction to provide fault tolerance. Because of performance and energy limitations of this approach, while considering the huge amounts of data still to be stored and managed, and in addition to technological limits of disks (heat dissipation limits the increasing of the disk RPM to enhance performance), adopting non-volatile memories, especially flash memories which are the most popular and mature candidate, is part of the answer. In fact, advances in the adoption of flash memories were considered as "*the most important technology changes pertinent to data centric computing*" [51].

Flash memory can be integrated into storage servers in three ways [141]: (1) as an extension of the memory system, (2) as a storage accelerator where flash is vertically integrated and is managed as a cache storing frequently accessed code and data and so reducing the number of accesses to HDD, or (3) as an alternative or complementary storage device where flash SSDs are used as a replacement to (some) HDDs.

2.3 Problem statement

The major issue when considering hybrid storage architectures is to define an *optimal* data placement policy according to some objectives and constraints. Data placement depends mainly on two sets of parameters: (1) characteristics of storage devices, for instance: throughput, latency, power consumption, and lifetime, and (2) properties of the workload, for example: type of operations (read/write rate), and I/O pattern (sequential/random percentage). Combining both sets of parameters is crucial in order to achieve optimal data placement according to administrator constraints and user/customer required QoS.

In a cloud context, the QoS is expressed in SLA terms. From a storage system point of view, mainly two metrics can be agreed upon in the SLA (more precisely the Service Level Objectives, SLO[14]) for storage systems: throughput (in IOPS, generally used to characterize random I/Os) or bandwidth (in MB/s for sequential I/Os) and access latency (in ms).

The main issue tackled in our research is to design a framework that would help the CSP to minimize the operation cost of IT equipment while satisfying the QoS terms by relying on hybrid storage systems. We are working on two Cloud services: IaaS and DBaaS. In the former, one needs to investigate VM placement in hybrid storage systems while in the latter database object granularity is used (an object can be any database entity such as a table, an index, etc.).

2.4 Related work

Many state-of-the-art studies have tried to integrate flash based storage systems with traditional ones, whether in a general context, or in a more specific Cloud context.

Vertical integration, flash on top of disks: flash memory in this case is used as a cache for the underlying storage system thereby reducing access to HDDs gaining performance and power. Flash storage management is achieved by user application, device driver stack and/or the firmware (for instance FlashPCI Express).

In [225], the authors proposed one of the first attempts to integrate flash memory technology as a complementary cache (in addition to DRAM) for a HDD in order to optimize I/O performance and energy consumption in laptops. Other studies such as [226] and [227] proposed integrating SSDs as a cache for disk subsystems with the objective to reduce energy consumption. For Cloud specific environment, [228] proposed implementing an SSD based cache at the hypervisor layer for performance optimization.

In a larger scale, flash memory is used to build tiered storage architecture consisting of flash disks at storage system front end for performance and traditional disks as a back end for capacity [229]. Using flash memory in a multi-tiered architecture consists in adding an array of flash disks tier on top of a HDD tier. When dealing with the integration of flash storage in tiered architectures, some important points are to consider among which: dimensioning the storage systems (e.g. how much SSDs, HDDs?), energy proportionality (variation in power consumption should be proportional to I/O workload), and data migration from one tier to another.

[14] We will use the more general term SLA in the rest of the document

In [230], tradeoffs and **dimensioning** of tiered storage systems with SSD tier as a cache for disk tier were described. This study has been performed on a set of real workloads and takes into account the point of view of an administrator that needs to upgrade or define the storage architecture according to workload requirements. Conclusions of the paper are quite pessimistic about the relevance of the use of SSDs because of price considerations, which contradicts the current adoption of SSDs (this paper was published in 2009).

Another important issue to figure out is **energy proportionality** [229] which also plays in favor of flash memories. This property has been used in [231] to design a power and reliability-aware hybrid storage systems. Indeed, disk based storage systems are not energy proportional as they consume almost the same energy amount in idle (spinning) state as in active state. The authors showed the benefit of relying on the synergy of both flash-caching and disk spin down in order to increase the global energy proportionality of the storage system. This is realized by considering many disk power states and by having an explicit management of disk reliability through a token bucket algorithm.

When dealing with tiered storage systems, **data migration** between SSD and HDD tier in time is a very important issue to investigate because of changing data access patterns over time. In [232], an adaptive data migration mechanism between storage tiers in the Cloud is proposed, especially to take full advantage of, and capitalize on, the expensive flash memory based (SSD) limited tier. The solution is based on the dynamic use of I/O profile to guide the migration of hot data toward SSD tier. Examples of considered parameters include: heat information of data, migration deadline, tradeoff between the gain in migrating the selected workload to higher level SSD tier and the degradation from the ongoing workload, etc.

Horizontal integration: a complementary storage device: In this model, hybrid storage systems include flash disks and traditional disks at the same level and depending on the application requirements and characteristics, data can be placed into SSD or HDD.

When conceiving such a system, designers must implement some data placement techniques and migration policies. This is done according to storage system performance and energy characteristics, in addition to application I/O access patterns.

Lightning [233] is an example of hybrid storage for Cloud systems based on a multi zone approach. The cloud storage servers are logically dynamically partitioned into hot and cold zones. Each zone having its own characteristics: performance, cost, data layout and power behavior. Flash storage is used in this architecture in the first zone, the one that contains a subset of the hottest data, more exactly for read intensive data. The other zones contain HDDs with different characteristics, and the last zone (4th one) is used for backup allowing long sleep (idle state) periods though consuming less energy. In order to perform high energy-conservation, lightning uses data-classification policies, application hints, and file system derived insights to place files in specific zones that are managed differently from the energy point of view. Other state-of-the-art work proposed some similar approaches about data placement according to I/O access patterns [234][235]. Another interesting study is I-CASH (***I**ntelligently **C**oupled **A**rray of **S**SD and **H**DD*) [237]. Here in addition to storing data into SSDs according their patterns, the authors took into account the lifetime of SSDs and avoid updating data directly to the SSD. They save data logs of data stored in SSDs into HDDs.

In [238] was described another data migration mechanism for hybrid flash/disk storage systems. The presented work transparently and automatically manages the migration of data through the storage system according to their access patterns. In this approach, the system dynamically keeps track of the access behavior (throughout counters) to measure read and write accesses. In order to have a transparent and generic approach, the system manages devices' characteristics with the help of continuous and dynamic I/O performance measurements for both read and write operations. This study focused on I/O performance and did not take into account energy consumption.

A replacement for traditional storage devices: in this model, the flash disk is supposed to replace HDDs. This would improve latency, throughput and the power consumption cost. Those advantages are compensated by the main brake to flash memory expansion that is the higher cost per Giga Byte. This is the main reason why flash memory is currently more seen as a mean to enhance traditional HDDs rather than a way to replace them in a short term [45]. Flash disks can be used in this case to architect the same storage platforms such as NAS or SAN.

One example of exclusively based storage system architecture is Gordon [123], a flash memory based cluster architecture for massively parallel data centric applications. Gordon framework combines three technologies: peta-scale data-centric parallel programming, low power processors, and flash memory.

From the industrial world point of view, many flash-based only storage systems have emerged in the last years, one may cite: Fusio-io, Intel, Kaminario, IBM, Violin, Virident, etc. For hybrid storage, one can site Tegile and Nutanix for horizontal integration and for vertical integration: NetApp, and EMC².

2.5 Contribution

In this work, we focused on studying data placement mechanisms for hybrid storage systems in Cloud environment for optimizing operation costs. As noted in the previous section, most state-of-the-art studies focused whether on optimizing performance or energy consumption. To the best of our knowledge, there were no studies considering: performance, energy and Cloud related parameters such as QoS metrics agreed upon in SLA, and penalties that should be paid by CSP in case of SLA violation.

For the sake of our study, the storage management middleware is autonomic and obeys the MAPE-K (Monitor, Analyze, Plan, Execute - Knowledge) reference model [239] as shown in Figure 50.

- In the **Monitor** step, one collects information from the managed resources. It consists in monitoring the I/O behavior of issued requests by customers' VMs on storage systems' disks (HDDs and SSDs).
- In the **Analyze** step, one performs some data analysis on collected data. It consists in analyzing the I/O workload of each VM in order to characterize it in terms of random/sequential read/write ratios, and performance requirement.
- The **Plan** step consists in structuring the actions needed to achieve some given goals. It consists in finding a data placement strategy for VMs or DB objects in order to optimize: performance, energy consumption, and customer's QoS according to a modeled global cost.
- The **Execute** step consists in realizing the actions planned by the previous step. It consists in performing data placement, this means dispatching VMs on storage systems with minimal cost.
- The cumulated **Knowledge** about VM I/O behavior and performance according to data placement is stored and shared among monitor, analyze, plan and execute functions.

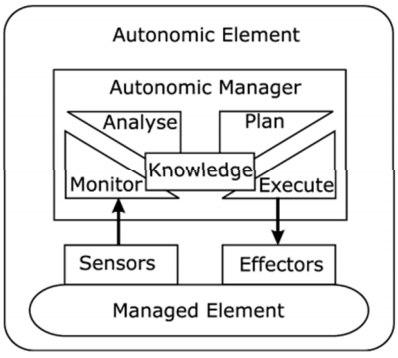


Figure 50. IBM's MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge) reference model for autonomic control [239]

This same approach has been used for two main projects concerning two Cloud services: IaaS and DBaaS. In the first case, data placement concerns VM images while in the second one, it concerns DB objects. A third project started in late 2015 with Orange Labs concerns IoT and Cloud. Its aims to perform data placement while having some knowledge on the overall network (a Fog approach) from the object and up to the Cloud data center. This last project will not be detailed in this document as it started on late 2015.

Figure 51 shows our global approach. In our work, we assumed that VM I/O workload is continuously monitored with an I/O tracer. Obviously this would have an impact on the overall performance, but it was measured to be small enough (<5%). Periodically (let us name this period "T"), the planner (Data decision maker in Figure 51) analyzes the overall system behavior to see if it is cost effective to migrate some VMs from one storage system to another (SSD and

HDD). The system behavior is mainly described in terms of **VM behavior** (I/O patterns coming from the workload analyzer Figure 51), **storage resource characteristics** in terms of performance energy behavior and lifetime, **storage system state** (VM placement, storage systems/performance space available per storage device, etc.). Once the planer decides to move some VMs in the next period, the data dispatcher proceeds to migrating VMs between devices. The data dispatcher needs to find the more efficient way to migrate VMs while minimizing the impact on VMs QoS. The same approach was adopted in case of DB objects with the difference in granularity that may lead to different optimization techniques.

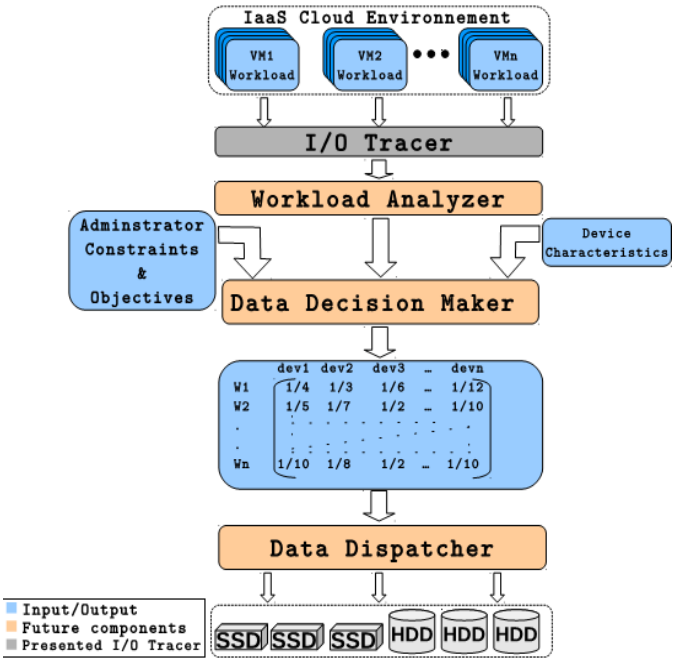


Figure 51. Instantiating the MAPE-K loop [w6]

Current status of the work

Currently, we have achieved three steps towards designing a MAPE-K solution. The first one is to develop a **monitoring infrastructure** for the Cloud context which also makes it possible to perform some **analysis** on the extracted I/O workload. Secondly, we have developed **cost models** to evaluate data placement of VM and DB objects on a hybrid storage system. In addition, we have upgraded an existing Cloud simulation tool [288] to account for I/O workload processing. Finally some heuristics about data placement for the **planner** were developed in order to find an optimized data placement for hybrid storage systems. This is described in the following three subsections.

2.5.1 The monitoring infrastructure

In the context of IaaS Cloud, monitoring is very important for both providers and customers. It is mainly used for billing and cloud orchestration [243]. The Cloud monitoring tools such as Ceilometer [244] for OpenStack and Nagios [245]give general quantitative information about the state of IaaS Cloud (network activities, system load, etc.). To optimize VM placement, one needs more details about the I/O operations performed. There are also different monitoring tools at different levels of the I/O software stack system. blktrace [240] is a well-known block layer I/O tracing tool which acts at the driver I/O request queue level. It provides detailed information about the state of I/O requests during its execution on the block storage device. iostat [246] is a Linux command that gives statistics about I/O device by observing the activity time of the monitored devices according to their average transfer rate. iotop [247] is an I/O monitoring program written in Python. It gives a user interface similar to the top Linux command output, by showing per-process I/O rates. Many other generic tracing frameworks exist, such as strace [241] and SystemTap [242]. They provide tracing facilities of user and kernel level functions. The tools presented above are not dedicated to monitor virtualized environments. Each tool works separately on a given system layer. The I/O tracer we developed

combines and increments different tools at different system levels. The goal is to have a global view and a better understanding of the overall I/O request flow for each VM.

I/O tracer architecture

An I/O request executed by a virtual machine goes through several software layers before reaching the physical storage device. In order to have a precise idea about the I/O access pattern of a given VM, we have to collect information from every relevant layer crossed by those requests.

At the hypervisor level: we look for the type and the size of I/O requests issued to the virtual storage devices. We also need to identify files accessed by different VMs in order to capture access patterns according to users' behavior. This is extracted in our tracer thanks to the libvirt API.

At the host level: we need to know which I/O operations are generated on the host VFS level once the I/O workload of the VM is issued. For such a purpose, we used strace in a first stage to monitor all system calls executed by the hypervisor. The results are then parsed and filtered to select only I/O related system calls.

At the file system level: I/O system calls are performed on files that are stored on physical storage devices, and organized using a specific file system. We do not trace at this level, but we need to perform a mapping between files captured in the preceding levels, and block numbers on the storage device. This is done in order to bridge the gap between the high level file view and low level block device view. We used libext2fs library [248] to extract the structure of each file in the physical storage device

At the I/O block layer: before reaching the storage device, I/Os can be merged and/or scheduled in the block I/O layer, then forwarded to the storage device. At this level, we used blktrace.

The final output is obtained by combining different tracer outputs according to the timestamp. Figure 52 presents the architecture of our I/O tracer. This figure shows the different levels in the tracing environment. The right hand side shows a layered vision of the virtualized system while the left hand side part shows the different tracing levels.

In a first version, we have used state-of-the-art tools and combined them to work together. **Then, we developed in the IRT b<>com, a standalone kernel module available from https://github.com/b-com/iotracer.**

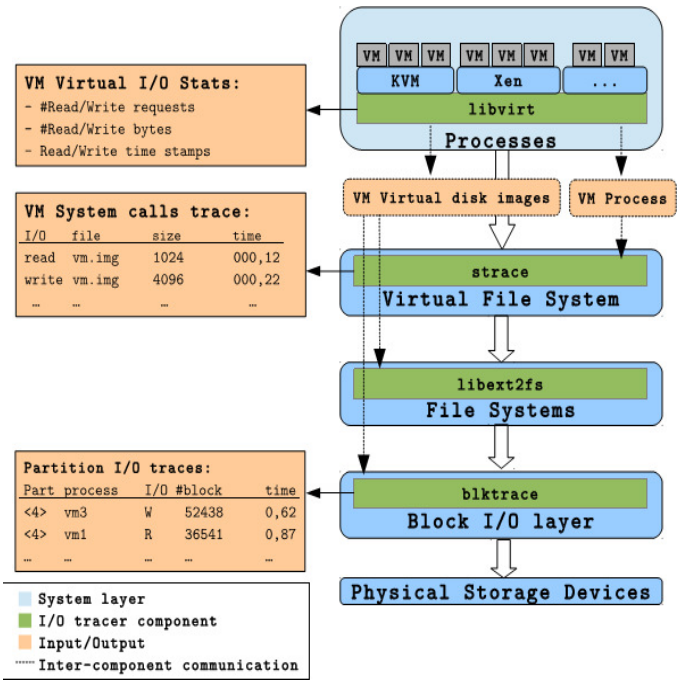


Figure 52. I/O monitoring architecture

Analysis of the I/O tracer

The developed I/O tracer is necessary to the MAPE-K loop as it helps to characterize the I/O behavior of each VM. This characterization is mainly based on the I/O pattern. In fact, we need to classify VMs (or DB objects) according to the issued workload in terms of: proportion of reads/writes, random/sequential workload, I/O burstiness, request sizes, etc. According to these metrics, it could be more relevant to use some given storage devices rather than other for instance SDD or HDD, or different categories of HDDs, or even different type of flash devices (user or enterprise SSD, PCI express flash memory, etc.). As it will be described in the cost model section, different storage classes can be defined according to several metrics, such as I/O performance, with regards to the sustained I/O workload.

The I/O tracer can also be used for OS, hypervisor, and VM resource tuning. In fact, it provides a precise idea of I/O request flow in the overall system: impact of hypervisor cache, VM RAM provisioning, OS page cache at different levels, etc. All these analysis data have not been yet fully exploited and will be in future work.

In addition to the released tool, a paper has been published with regards to I/O monitoring in a Cloud context: [w6].

2.5.2 The VM storage cost model

One necessary step toward the implementation of an optimization method is the design of an accurate cost model. In order to do so, 4 sets of parameters need to be considered.

First, a precise model of **storage** system characteristics must be established. Then, the cost model needs to take into account the **virtual environment** as it was proved to have a high impact on the storage system performance [250]. The cost model needs to take into account the **cloud** end user specific requirements, such as those related to the QoS in SLA terms. Finally, **energy** consumption should be considered for data placement.

Table 17. state-of-the-art cost models

State-of-the-art work	VM	Energy	Storage	Cloud
[249] [257] [258]	✓	✓	✓	✗
[252] [253] [254] [255] [256]	✓	✓	✗	✗
[259] [260]	✗	✗	✓	✓
[261] [235]	✗	✓	✓	✓
[251]	✓	✓	✗	✓
[263] [262]	✗	✓	✓	✗

Several cost models have been proposed in state-of-the-art work, see Table 17. Many focus on VM's energy consumption [249][251][252][253][254][255][256][257][258], from which some consider storage issues [257][249][258]. There are also several work that dealt specifically with storage systems [259][260][261][262][263][235], while some of them consider the Cloud context [259][260][261][235], others do not [263][262]. As one can observe in Table 17, none of state-of-the-art models cover the whole parameters.

VM storage cost, as seen in equation (27), is composed of three parts for a given monitoring period: (1) $Cost_{nrc}$, some non-recurring costs including expenses not directly related to the VM execution, (2) $Cost_{exe}$, the costs related to the VM execution on a given storage device D_i, (3) and $Cost_{mig}$, the cost of VM migration from a storage device to another one if any. Indeed, we consider that our optimization system will be ran periodically (each period T) to find out the optimal placement and move VMs in order to minimize the cost.

$$Cost_{stg} = Cost_{nrc} + \sum^{nb\ migrations} Cost_{mig}(VM, D_{source} \rightarrow D_{dest}) + \sum^{nb\ VMs} Cost_{exe}(VM, D\ ) \quad (27)$$

In this equation, the non-recurring cost (NRC) includes fixed expenses that do not depend on VM data storage or Cloud environment. This cost may be composed of resource setup, data center space rental, human resources cost, etc. In our model, the NRC is assumed to be constant for all VMs.

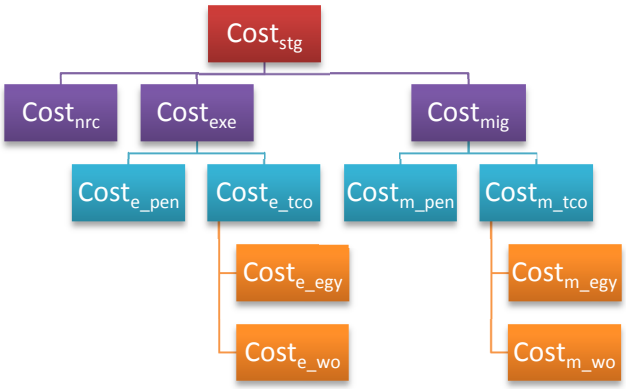


Figure 53. storage cost decomposition

As depicted in Figure 53, each of the execution and migration costs are decomposed into two different costs. The first one is related to the total cost of ownership namely **$Cost_{e\_tco}$** and **$Cost_{m\_tco}$**, respectively for execution and migration. The other cost is due to penalties that could be applied to the CSP in case of SLA violation, **$Cost_{e\_pen}$** and **$Cost_{m\_pen}$**, respectively for execution and migration. In effect, we consider that a penalty may happen whether during a VM migration for instance if the customer I/O requests completion is delayed, or during I/O workload execution, for example because of a burst of I/O request due to too many customers sharing the same SSD. In a given time window T, we suppose that the total penalty equals the sum of the execution and migration penalties.

The total cost of ownership is declined into two main costs, the energy cost noted **$Cost_{e\_egy}$** and **$Cost_{m\_egy}$**, respectively for execution and migration, and the wear out cost denoted **$Cost_{e\_wo}$** and **$Cost_{m\_wo}$** for execution and migration. Both are described in the following subsections.

Energy cost

The energy consumption cost is calculated from the amount of energy consumed during the I/O workload execution by a VM (E_{tot}), and the energy unitary price (UPe). It may be formulated as follows:

$$Cost_{e\_egy}(VM, D) = E_{tot}.UP_e \quad (28)$$

The energy unitary price (UPe) used in previous equation may be obtained from the local electricity authority. We assume that the amount of energy consumed during I/O workload execution (E_{tot}) is the sum of energy consumed by the main hardware components (CPU, memory, disks) contributing to workload execution [269][263][235].

For the rest of this subsection, we will focus on the storage system energy consumption. The CPU and memory part will be discussed later.

Consequently, the energy consumption of the storage device Di is calculated with:

$$E_{D_i} = P_{D_i}.T \quad (29)$$

We recall that T is the monitoring period. For simplicity reason, we assume that a given VM is stored on a single storage device (no replication issues considered, we expect to investigate this in the near future).

The power use of a given device depends on its state during the workload execution. An HDD may be operating in three modes: (1) operating mode, (2) idle mode, and (3) standby/sleep mode. The HDD is in operating mode when it processes I/O requests. The power is then high because of disk rotational movement and read/write head activities. The HDD switches to idle mode after a predefined timeout. The power use in this mode is related to the spinning disks. The lowest power use is the standby mode, because there is no mechanical activity.

For instance, for a given HDD:

$$E_{D_i} = P_{op}.T_{op} + P_{idl}.T_{idl} + P_{stdby}.T_{stdby} + (n.E_{D_i spn}) \quad (30)$$

For an HDD, the amount of consumed energy (E_{D_i}) during the monitoring period (T) is obtained from power values of different states (P_{op}, P_{idl}, P_{stdby}), and time passed in every state (T_{op}, T_{idl}, T_{stdby}). There is another amount of energy consumed during the HDD spin-up (E_{spn}), which is the transition from the standby mode to the operating mode, and *n* the number of spin-ups.

For SSDs, data sheets [25] mention three power modes, one operating mode and two idle modes. We used a similar equation.

The power in operating mode (P_{op}) varies depending on I/O access type (read/write) and/or I/O access pattern (random/sequential). For an HDD, performance and power are asymmetric between sequential and random accesses, then the access pattern impacts the energy consumption. In the case of SSD, an asymmetry exists between the I/O patterns (seq, rnd) and I/O types (read, write) [54]. The amount of energy consumed during the operating mode is then calculated as follows:

$$E_{op} = \sum_i \sum_j (P_{i,j}.T_{i,j}) \; where \begin{cases} i \in \{seq, rnd\} \; and \\ j \in \{read, write\} \end{cases} \quad (31)$$

The values of $T_{i,j}$ and $P_{i,j}$ are obtained from device specification data sheets or from measurements. In our case, time values ($T_{i,j}$) are obtained from I/O trace analysis, and power values ($P_{i,j}$) are measured using a Power Distribution Unit (PDU). The time duration of sequential/random read/write I/O operations are calculated from the rate of sequential/random read/write operations ($R_{i,j}$), during the operational time (T_{op}). This information is gathered thanks to the I/O tracer developed.

Subsequently, the total energy for executing a given VM's workload can be calculated as follows for a given storage device, see [58]:

$$Cost_{e\_egy} = \left[\sum_i \sum_j (P_{i,j}.(\frac{R_{i,j}}{100}.T_{op})) + P_{idl}.T_{idl} + P_{stdby}.T_{stdby} + (n.E_{spn}) \right] .UP_e \quad (32)$$

Wear out cost

The hardware wear out cost is more and more integrated into cost models [261][262]. This cost is usually calculated from the device lifetime estimation. For a storage device, the lifetime expectancy depends on the device type and its characteristics. In the case of HDD, the lifetime is estimated using different parameters usually provided in the data sheet, such as MTTF (Mean Time To Failure), AFR (Annualized Failure Rate), or number of start-stop cycles. The HDD lifetime is service time dependent. With warranty period, purchase and service costs, one can estimate the wear out cost of a given HDD.

Wear out is considered as one of the most important issues for flash based storage devices. For SSDs, it depends on the number of write/erase cycles. In our study, the wear out cost of storage device depends on VM's I/O workload:

$$Cost_{e\_wo} \propto \begin{cases} f(NB_{write}) & for\ SSD \\ f(MTTForAFR\ or \#start-stop) & for\ HDD \end{cases} \quad (33)$$

Equation (33) shows that SSD wear out cost depends on write operations generated by VMs, while for HDD, it depends on service time. We assume that an SSD is worn out and must be changed when it reaches the total data size written over the warranty period, which is specified in data sheets (e.g. 600TB written over 5 years warranty period for a 128GB SSD). In our case, we have to predict the amount of written data by each VM to quantify its SSD wear out cost. The amount of data written ($write_{tot}$) is obtained from the number of write requests (NB_{write}) and the average write request size (IO_{size}). In our approach, the number of write requests executed by a VM is obtained by analyzing VM's I/Os from the trace.

The next step is to quantify the wear out cost per MB written on the storage device ($Cost_{wo\_per\_mb}$), using the obtained data written by the VM and the device characteristics. This cost is calculated from the SSD unitary price (S_{up}), its capacity (C_{stg}), and the maximum amount of data written (M_{wrt}) during warranty period (W_{prd}):

$$\text{Cost}_{\text{wo\_per\_mb}}(SSD) = \frac{S_{up}.C_{stg}}{M_{wrt}(W_{prd})} \tag{34}$$

We can estimate the execution wear out cost as follows:

$$Cost_{e\_wo}(VM, SSD) = \text{Cost}_{\text{wo\_per\_mb}}(SSD) \,.\, write_{tot}\,(VM) \tag{35}$$

The total wear out cost for SSD is given as follows:

$$Cost_{e\_wo}(VM, SSD) = .\left[\frac{S_{up}.C_{stg}}{M_{wrt}\,(W_{prd})}\right].\left[\,\left(T_{rnd,write}.IOPS_{vm}.IO_{size}\,\right) + \left(T_{seq,write}.TR_{vm}\,\right)\right] \tag{36}$$

The wear out cost for a hard drive is not as straight forward, because of the statistical nature of parameters given in data sheets (e.g. MTTF, AFR, MTBF, etc) [264]. The number of start-stop cycles is more suitable to predict the wear out cost [262]. This data can be inferred with the help of S.M.A.R.T (Self-Monitoring, Analysis and Reporting Technology) in modern HDDs. In this case, the wear out cost of an HDD can be obtained using the cost per one start-stop cycle ($Cost_{start\text{-}stop}$), and the number of start-stop cycles ($N_{start\text{-}stop}$) during the monitoring period (T):

$$Cost_{e\_wo}(VM, HDD) = \text{Cost}_{\text{start\_stop}}(HDD) \,.\, N_{start-stop} \tag{37}$$

In the same way as equation (34), the cost per start-stop cycle ($Cost_{start\text{-}stop}$) for an HDD is obtained using its unitary price (S_{up}), its capacity (C_{stg}), and the number of start-stop cycles ($N_{start\text{-}stop}$) during the warranty period (W_{prd}):

$$\text{Cost}_{\text{start-stop}}(HDD) = \frac{S_{up}.C_{stg}}{N_{start-stop}\,(W_{prd})} \tag{38}$$

The total wear out cost for HDD is given as follows:

$$Cost_{e\_wo}(VM, HDD) = \frac{S_{up}.C_{stg}}{N_{start-stop}\,(W_{prd})}\,.\;N_{start-stop} \tag{39}$$

Penality cost

In the context of Cloud computing, the SLA is critical [265] as it specifies a set of terms to guarantee the QoS requested by the customers. If the QoS offered by the Cloud provider ($QoS_{offered}$) is less than the one requested by the customer ($QoS_{requested}$), then a penalty is applied to the provider. A specified amount (x) must be subtracted from the customer bill (depending on Cloud service unitary price UP_{cs}), and a new bill amount is calculated ($Bill_{tot}$) [265][266][267][268].

$$if\ QoS_{offered} < \text{QoS}_{\text{requested}} => Bill_{tot} = Bill - x\% \tag{40}$$

We relied on three main SLA models to calculate the penalty (x in equation (40)) [265] in case it applies:

- **Fixed penalty**: in this case, x is fixed and the price will be deduced every time the SLA terms are not met.
- **Delay dependent penalty**: x varies according to the period during which the terms of the SLA were violated.
- **Proportional penalty**: in this type of penalty x depends on the ratio between the offered and the requested QoS.

Independently from the storage systems, Cloud service providers calculate penalties using some mixed approach between proportional and fixed penalty [266][267][268]. They define a stepwise increase of the penalty if the offered QoS is not met.

We used the proportional penalty to calculate penalty cost. We used the following equation to estimate the penalty cost (note that any penalty model could be used):

$$Cost_{e\_pen}(VM, HDD) = x.Bill \quad \textbf{where} \begin{cases} x \geq 0 \ and \\ x = \dfrac{QoS_{offered}}{QoS_{requested}} \end{cases} \tag{41}$$

The allowed QoS parameters in our study are the throughput (in IOPS) or the data transfer rate (bandwidth in MB/sec). One can also consider the I/O latency.

There are some slight differences between the execution cost and the migration cost as for the latter (1) the size of the copied data is known; it is equal to the VM image size, (2) the pattern of the workload is also known; it consists of write operations from the source device and read operations to the destination device.

Here is an example of the total storage cost of VM on SSD, migrated to an HDD (from equations (32)(36)(37)(41)):

$$\begin{aligned} Cost_{stg} = & \left[\left[\sum_i \sum_j (P_{i,j}.(\frac{R_{i,j}}{100}.T_{op})) + P_{idl1}.T_{idl1} + P_{idl2}.T_{idl2})\right].UP_e\right] \\ & + \left[\left[\frac{S_{up}.C_{stg}}{M_{wrt}\ (W_{prd})}\right].\left[\left(T_{rnd,write}.IOPS_{vm}.IO_{size}\right) + \left(T_{seq,write}.TR_{vm}\right)\right]\right] \\ & + \left[\left(1 - \frac{TR_{dev}}{TR_{req}}\right).Bill\right] + \left[\sum_i \left(P_{i,read}.T_{i,read}\right) + \sum_i \left(P_{i,write}.T_{i,write}\right)\right] \\ & + \left[\frac{S_{up}.C_{stg}}{N_{start-stop}\ (W_{prd})}.\ N_{start-stop}(T_{mig})\right] + Cost_{nrc} \end{aligned} \tag{42}$$

Where the first term is related to I/O workload of the VM execution, the second one is related to the wear out cost of the SSD, the third one is penalty cost, the fourth one is VM migration cost (reading then writing), the fifth one is the wear out cost of the HDD and the last one is the NRC cost.

Some performance measurement issues

In order to get storage device performance and power parameters, one can use parameter values from the device data sheet specification. This method has some drawbacks. First, those values are obtained in a specific test environment that is hard to reproduce. Second, it is difficult to isolate the power consumption of the storage device from the other components (CPU, RAM, etc.). We then need specific tools [270]. In our study, we considered that the storage device power is the difference between the overall power during I/O execution and the one in idle mode. This includes CPU and memory I/O processing. We used a Power Distribution and Metering Unit (PDU) to obtain power parameters ($P_{seq,read}$, $P_{rnd,read}$,$P_{seq,write}$, $P_{rnd,write}$) for a given storage device.

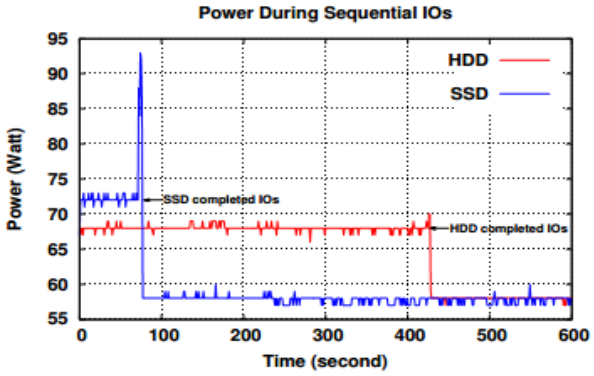


Figure 54. Sequential I/O power calibration

Figure 54 shows the overall power consumption (CPU, RAM, and storage device), during sequential I/Os execution on HDD and SSD. We used a server containing only one SSD and one HDD. We can observe that while performing I/O operations on SSD, the whole system consumes ~20% more than the idle state, and around 17% in case of HDD.

One can also notice that during SSD I/Os, the system consumes more power (~72 W) than for HDD's (~68 W). This does not mean that SSD consumes more than HDDs, but the overall power values when performing I/Os using SSD is higher than that for HDDs. This includes CPU and memory power consumptions. As SSD gives a much higher throughput, it pushes CPU and RAM to process more I/Os in a unit of time making the overall power consumption higher.

From the energy perspective, HDD I/Os consumes 4 times more energy than SSD's in the considered platform. For random I/Os, the power is ~60 W for HDD and ~71 W for SSD.

We observed, during calibration, that there was no significant difference between read and write power consumption for SSD and HDD tested. The main difference was noticed between read and write performance times, and sequential and random power values. Consequently, we considered that $P_{seq;read} = P_{seq;write}$ and $P_{rnd;read} = P_{rnd;write}$.

We performed some experiments to validate the execution and energy models developed by issuing different I/O workloads and we observed that our cost model performed well as the error ratio was around 2% for HDD with a peak value around 4%, and 8% for SSD with a peak value of 10% [58].

We published two papers related to cost modeling in a Cloud context, one related to IaaS [ci6] (for which this section presents a summary) and the other to DBaaS services [ci4].

2.5.3 Data placement strategies

In order to give an example about our contributions with regards to data placement strategies, we will rely on the study published in [ci2] which is related to data placement in a DBaaS Cloud. We consider a CSP that uses multitenant DBMS to provide database service (DBaaS) to a large number of customers. Customers have different I/O workloads, different I/O performance requirements, and different SLAs. We assume that customers can share the same storage device. The question we are trying to solve is **how to find the right object placement that meets customers' I/O requirements and minimizes the overall I/O cost?**

We proposed two strategies for database object placement, one based on genetic algorithms and a heuristic one. We will briefly detail the heuristic approach in this document.

2.5.3.1 Problem formulation

We call customer u_k any entity that hosts its database in the Cloud. Each customer u_k has a database that holds a set of objects $O_{u_k} = \{o_{1,u_k}, \dots, o_{i\,u_k,}\}$. In this work, an object is any logical entity of a database. We denote $s_{o_{i,u_k,}}$ the size of object $o_{i,u_k,}$. We consider the data placement at a database object granularity.

We consider that each customer u_k has an agreed upon I/O throughput represented by $iops_{soft,u_k}$ (soft SLA) and he tolerates a degradation down to $iops_{hard,u_k}$ (hard SLA). Let $iops_{offered,u_k}$ denote the IOPS delivered by the CSP to customer u_k. If the $iops_{offered,u_k}$ is between $iops_{soft,u_k}$ and $iops_{hard,u_k}$, a penalty of amount p is applied to the CSP. p is calculated by using a penalty function pn_{u_k} that depends on $iops_{offered,u_k}$. In our optimization problem, the $iops_{soft,u_k}$ is considered as a soft constraint and we assume that $iops_{hard,u_k}$ is a hard constraint that requires to be satisfied.

We assume that the storage system consists of two storage device classes, SSDs, and HDDs. For simplicity reasons, we consider that storage devices in a given class are homogeneous in terms of performance. Each storage class d_j with j = {HDD, SSD} has a capacity c_{d_j} and an I/O throughput $iops_{op,d_j}$ for I/O operations of type *op*. Customer' objects can be distributed among different storage classes (different from the case of IaaS). We denote O_{d_j} the set of objects placed in the storage class d_j. In this study, we handled the object placement at the granularity of a storage device class. We assume that the cloud administrator employs a virtualization technique that abstracts physical disk resources considering only classes.

In practice, customers' I/O workloads may change over time. In our model, we suppose that cloud administrator takes periodical decisions about object placement. T denotes the time period over which the overall cost for a placement ($Cost_{pl,T}$) is evaluated and decision about object placement is made. We define a placement as a mapping from a set

of objects to a set of storage device classes. The mapping function $pl(o_{i,u_k})$ indicates the storage class d_j which stores the object o_{i,u_k}. To achieve a relevant object placement for the next time period, the cloud administrator needs to predict the I/O profile. We simply assumed that the I/O profile of the next period is the same as the previous one (this assumption needs to be reevaluated for future work).

We distinguish four types of I/O operations (op): random read (rr), sequential read (sr), random write (rw), and sequential write (sw). In our study, we achieve continuous monitoring in order to extract objects' I/O patterns.

Our objective is to minimize the overall cost ($Cost_{pl,T}$):

$$\begin{cases} Minimize(Cost_{pl,T}) \\ \forall\, d_j, (\sum_{o_{i,u_k} \in o_{d_j}} s_{o_{i,u_k}}) \leq c_{d_j} \\ \forall\, d_j, \sum_{op \in OP} (\frac{\sum_{o_{i,u_k} \in o_{d_j}} req_{op,o_{i,u_k}}}{iops_{op,d_j}}) \leq 1 \\ \forall\, u_k \in U, iops_{offered,u_k} \geq iops_{hard,u_k} \\ \forall\, o_{i,u_k} \in O_{u_k}, \exists!\ d_j, pl(o_{i,u_k}) = d_j \end{cases}$$

This cost is calculated using our cost model for DB placement defined in [59]. Our optimization problem is subject to several constraints. The first one is related to the limited storage capacity of each class. The second one is related to the finite performance (IOPS) of each storage class. The third one is related to SLA constraint for which the performance degradation should be bounded. Finally, the last constraint specifies that each object must be stored only once (no replication for the time being).

2.5.3.2 H-COPS, a heuristic cost based object placement strategy

In H-COPS we build incrementally the best object placement through three steps: initialization, feasibility and optimization. We start by placing each object in the storage device class that has the lowest storage cost. Then we modify incrementally the initial placement to make feasible the solution generated in the previous step. Finally, we update the obtained placement in order to avoid customer penalties. We will discuss each step in more details below.

Initialization:

The initial placement is built by placing each object $o_{i,u_k} \in O$, in the storage device class that has the lowest cost for holding it, denoted $d_{cheap}(o_{i,u_k})$. We compute $d_{cheap}(o_{i,u_k})$ as follows:

$$d_{cheap}(o_{i,u_k}) = \min\{Cost(o_{i,u_k}, d_{hdd}), Cost(o_{i,u_k}, d_{ssd})\} \quad (43)$$

Where $Cost(o_{i,u_k}, d_{hdd})$ (respectively $Cost(o_{i,u_k}, d_{ssd})$) represents the placement cost of o_{i,u_k} in the HDD class (respectively SSD). We compute the placement cost of object o_{i,u_k} ($Cost(o_{i,u_k}, d_{hdd})$ /or $Cost(o_{i,u_k}, d_{ssd})$) by using our cost model presented in [59]. It includes both the storage cost and the migration cost to take into account current placement. Similarly, we denote $d_{costly}(o_{i,u_k})$ the storage class that has the highest placement cost for object o_{i,u_k}.

Feasibility:

This step improves the initial placement plan to make it feasible. A feasible placement should satisfy all the constraints defined earlier. We achieve the feasibility study in two steps: (1) storage constraint validation, and (2) SLO and performance constraints validation.

Storage constraint: For each storage device class we should guarantee that the size of database objects it contains is lower than its effective capacity. This step is skipped if this constraint is already satisfied. If the size of objects is greater than the sum of the capacities of both storage classes, the optimization process exits (no feasible placement). If one storage class is overfilled after the initialization phase, we update the object placement plan by moving some selected objects to the other class. This object selection is based on a greedy technique that requires to compute a score value (δ) for each object in the overfilled storage class. The idea behind this score is to move objects so that to

minimize the cost while maximizing the freed space. The score gives the priority to moving objects having large size and showing a low cost difference between cheap and expensive storage classes, as given below:

$$\delta(o_{i,u_k}) = \frac{s_{o_{i,u_k}}}{d_{costly}(o_{i,u_k}) - d_{cheap}(o_{i,u_k})} \quad (44)$$

Next, objects are sorted according to their score in descending order. Objects are moved in this order one by one until the storage constraint is satisfied.

SLO and performance constraints: For each customer u_k, the system should guarantee that the IOPS offered ($iops_{offered,u_k}$) exceeds the ($iops_{hard,u_k}$) specified in its SLA. When the IOPS really requested by the customer are lower than $iops_{hard,u_k}$, we should guarantee that $iops_{offered,u_k}$ equals to IOPS really requested.

Therefore, for each customer u_k, if the hard SLO constraint is satisfied, we do not move any object. Otherwise, we attempt to rearrange its objects in order to find the best placement that satisfies the hard SLA constraint and optimizes the cost. The idea is to move the objects, from HDD to SSD, that would improve performance the most while their moving do not cost a lot (e.g. small objects sustaining high random reads) until hard SLA constraint satisfaction.

We compute a score (λ) for each object for all customers with violated hard SLA, then, we sort them according to (λ) in descending order. We update the placement plan by placing the objects on their expensive class according to their scores until customer's SLO constraint is satisfied. (λ) is a ratio between performance gain and cost difference of moving objects as shown in eq. (45). The performance gain represents the execution time saved from moving from HDD to SSD. We calculate execution times related to objects ($t_{exec}(o_{i,u_k}, d_j)$) from the number of I/O requests issued and the storage device class response time (t_{op,d_j}) as shown in eq. (46). The value of $req_{op,o_{i,u_k}}$ is obtained from a monitoring phase, while the value t_{op,d_j} can be obtained from device data sheet or measured through a calibration phase [59].

$$\lambda(o_{i,u_k}) = \frac{t_{exec}(o_{i,u_k}, d_{costly}) - t_{exec}(o_{i,u_k}, d_{cheap})}{d_{costly}(o_{i,u_k}) - d_{cheap}(o_{i,u_k})} \quad (45)$$

$$t_{exec}(o_{i,u_k}, d_j) = \sum_{op=\{rr,sr,rw,sw\}} req_{op,o_{i,u_k}} . t_{op,d_j} \quad (46)$$

Optimization:

The goal of this step is to investigate trade-offs between storage cost and penalties to find the right object placement. Let us denote O_{soft,u_k} the set of objects of customer u_k needed to be moved to the expensive class in order to avoid penalty. This set is inferred from score calculation during the feasibility phase. We calculate the extra cost needed to avoid the penalty of customer u_k as shown in eq. (52).

$$Cost_{SLA}(u_k) = \sum_{o_{i,u_k} \in O_{soft,u_k}} d_{costly}(o_{i,u_k}) - d_{cheap}(o_{i,u_k}) \quad (47)$$

The idea is to start by avoiding penalties of customers having high penalty and low $Cost_{SLA}(u_k)$ (see Eq. (48)). Then, for each customer we compute a score (μ) which represents the difference between its penalty cost ($Cost_{pnl}(u_k)$) and its $Cost_{SLA}(u_k)$ as shown in eq. (48). Then, we sort customers in descending order. We handle customers one by one in the sequence presented previously until all soft SLAs are satisfied, or SSD resources are estimated to be critically low (we can fix a threshold below which SSD performance degradation, related for instance to garbage collection, is estimated to be too high).

$$\mu(u_k) = Cost_{pnl}(u_k) - Cost_{SLA}(u_k) \quad (48)$$

This way, soft SLOs are satisfied while cost is optimized without necessarily overloading the SSD.

Performance evaluation has been conducted comparing H-COPS with state-of-the-art mechanisms such as OPA [323] and DOT [322] using TPC-C and TPC-H workloads. It turned out that H-COPS was closer to the optimal solution

given by an exact algorithm. From the cost metric perspective, VM placement given by H-COPS enhanced the ones of state-of-the-art solution by 40% on average while it executed in few seconds for thousands of objects which makes it an ideal mechanism for runtime execution, one may refer to [60] for more details.

2.6 Conclusion & perspectives

We have presented in this chapter some of our contributions with regards to storage management in a Cloud context. More particularly, we discussed the developed multilevel I/O monitoring framework that was designed to track I/O requests from VMs at different level for a better understanding of the I/O behavior. Then, we summarized one of the cost models used as a base for data placement strategy. Actually, we developed two different cost models for two different case studies: IaaS [58] and DBaaS [59]. Finally, we have described part of our work about data placement strategies in hybrid storage systems [60].

There is still a lot to do in this topic and we are undergoing many studies related to storage and Cloud. We will enumerate some of them briefly in this section.

Execution step investigation. In order to close the MAPE-K loop, we still need to design the executor part. Once the planner issues the best data placement strategy, one needs to apply it on a running system. Many questions may arise: when and how to migrate the VMs (in case of IaaS service) or data objects (in case of DBaaS service) without disturbing the system, i.e. without degrading the QoS. In fact, one should compromise between three parameters: (1) the execution speed of the migration plan, (2) the disturbance generated by the migration on system execution, and (3) the efficiency of the executed plan. A large space of solutions is available, a part of which has been investigated in state-of-the-art work. On the one hand, one may use the remaining IOPS to perform migrations. In this case no guarantee can be given about when the actual placement strategy will be effectively performed, and once performed if it is still relevant. On the other hand a given amount of IOPS can be booked periodically to perform object or VM migrations. This is prone to SLA violation. Another alternative could be to compromise between data placement plan efficiency and migration cost, for instance through some multi objective mechanisms. This issue is being currenlty investigated.

From the evaluation point of view, we incremented Cloudsim [288] in order to take into consideration storage system energy consumption [199]. Cloudsim is an Open Source framework for modeling and simulation of Cloud computing infrastructures and services. This simulator does not include natively a support for hybrid storage systems. We developed such a support in order to integrate the storage system cost in the overall cost and investigate to what extent the storage system may impact the system performance. So, not only the CPU system time for processing the I/O was integrated [236], but also its energy consumption, and the whole storage system time and energy. From the dispatcher point of view, one should study in details the migration scheduling amongst the storage devices and their impact on performance and customer's QoS.

The case of federated Clouds and distributed data centers. We are also working toward considering geographically distributed data centers and data placement issues in distributed architectures. Indeed, both work of H. Ouarnoughi about VM placement and D. Boukhelef about DB object placement were performed without taking distributed architectures into considerations. We are investigating this issue and considering replication/migration policies in a federated Cloud architecture with hybrid storage systems. In such a case, for a given CSP, many new parameters need to be considered: the FLA (Federation Level Agreement), the dynamic nature of free resources, fluctuating cost models (different in each CSP), etc. These issues are studied in collaboration with USTHB University in Algeria and are a subject of a new PhD thesis.

SSD performance variation. In the achieved studies about VM and DB object placement, we assumed that the throughput of a given SSD is stable. In fact one could overprovision the performance values so that to satisfy the QoS in the worst case. In reality, the performance of an SSD depends on many parameters related to application, system and architectural intricacies. This performance variation can be huge, for instance if the garbage collector is being started because of some bursty write operations. So, one needs to have an accurate SSD performance model to avoid, on one hand, performance overprovisioning, and on the other hand, SLA violation. Due to the complexity of both SSD internal architecture and application behavior in a Cloud context, we are investigating the use of machine learning to model SSD-based storage system performance in a container Cloud environment. More generally, we are seeking to

develop some solutions to ensure resource isolation and optimization in a container context. We will investigate both main memory (such as [305]) and storage system resource optimizations. This work is achieved in a Cloud related project at the IRT b<>com, it gave place to new PhD thesis.

Data placement in a Fog infrastructure. We are also studying data placement in a Fog infrastructure with heterogeneous Fog nodes. This a very interesting challenge because of the rise of IoT applications in which many application instances can be deployed on different nodes of the Fog infrastructure. In addition, data can be shared between applications and stored in different (hybrid) storage systems. In this context, we are investigating data placement strategies to reduce data access latency through some dynamic and intelligent data placement algorithms. This study is integrated in the PhD thesis funded by Orange labs.

2.7 Outcome

Advised PhD student	**Projects**	**Collaborations**	**Publications**		
			Journals	**Int. conf.**	**Others**
Mr. Hamza Ouarnoughi	IRT b<>com Indeed	UBO (F. Singhoff, S. Rubini)	1	4	3
Mr. Djillali Boukhelef	PHC Tassili	USTHB, Algérie (K. Boukhalfa)			
Mr. Islam Naas	Orange	Orange Labs (P.Raipin)			
Mr. Jean-Emile Dartois	IRT b<>com Falcon	IRISA Rennes (O. Barais)			

3 ONGOING WORK AND PERSPECTIVES ON DATABASE STORAGE SYSTEMS

3.1 Summary

This chapter describes our contribution regarding Database systems. We will focus on three studies: (1) a work realized on characterizing and modeling of simple SQL queries on embedded systems based on flash specific file systems, (2) a novel algorithm for sorting data on SSDs that emphasizes on the scalability in a way to have good performance even in case of very large datasets under memory pressure, and (3) an initiated work on designing system prefetching mechanisms based on SSD performance model in order to optimize database access. We are carrying on our work about data management and storage systems with an industrial partner through the PhD thesis of Mr. Arezki Laga, and we are also starting to work on Big data framework, such as SPARK, for storage optimization in collaboration with The Hong Kong Polytechnic University and USTHB Algeria.

3.2 Context

Big data phenomenon makes critical the need for storing, managing and querying/extracting information in an efficient way. Huge amount of data issued from various heterogeneous sources are expected to be processed in order to increase the decisional power. Relational data warehousing is one of the most valuable technology to deal with this challenge. The analytical process is usually performed by the means of complex queries.

Recently, some research efforts have been made to study the contribution of query interactions to manage the buffer of DBMS. These efforts assume a traditional architecture of the DBMS in terms of storage: HDD for storing the huge amount of data and RAM for buffering data to speed up queries.

Since the advent of flash memories, many of the database related optimizations have been revisited and some new ones designed to take advantage of this new technology. In fact, as previously mentioned, flash memories have a different performance model as compared to traditional storage devices.

3.3 Problem statement

Within this context, many challenges need to be answered with regards to database optimization for flash technology. These optimizations can concern different levels of the system architecture going from high level applications (complex queries) to kernel optimizations (perfetching mechanisms).

In the rest of this chapter we will summarize three of our contributions in this domain. Each contribution can be read independently from the others:

- **Analyzing and modeling the impact of flash memory on embedded database systems**: Databases are more and more used in embedded system applications and especially in consumer electronics. This comes from the need to structure user and/or system data to be more efficiently managed and accessed. The transactional DBMSs widely used in embedded systems has been designed considering HDD as a storage device. In embedded systems, NAND flash memory is the main storage media and its internals make it fundamentally different from HDD. The performance behavior of embedded on-flash database applications was largely unknown, and we believe a better adequacy between applications and flash management systems would lead to strong optimizations. A first step toward this end was to explore and understand the performance behavior of database applications on flash memory. For this sake, we characterized and modeled SQLite database simple queries performance on embedded flash specific file systems. This work has been published in [62] [200].
- **Revisiting the data sorting problem on flash memory devices:** External sorting algorithms are commonly used by data-centric applications to sort large amounts of data that are larger in size than main-memory. Many external sorting algorithms were proposed in state-of-the-art studies to take advantage of SSD performance properties to accelerate the sorting process. We have observed that unfortunately, many of those algorithms fail to scale when it comes to increasing the dataset size under memory pressure. In order to tackle this issue, we proposed a new sorting algorithm named MONTRES (Merge ON-The-Run. External Sorting). MONTRES relies on SSD performance model while decreasing the overall number of I/O

operations. This contribution is part of the PhD thesis of Mr. Arezki Laga. This work has been published in [64].

- **Optimizing file prefetching mechanisms at the kernel level for database I/Os:** In order to achieve a good overall application performance, optimizations need to be achieved at the application level but also in the operating system's kernel as it interacts with the application layer to issue the final I/O request flow to the storage device. Current data-intensive applications I/O workloads are turning towards more random patterns while from the storage device perspective SSDs present good random read performance. In this work, we describe a new prefetching mechanism named Lynx. Lynx aims to adapt and/or complement the Linux sequential read-ahead prefetching system for SSD performance model and new applications needs. This work has been published in [65].

3.4 Related work

We chose not review all state-of-the-art work on database optimization for flash memories but rather try for each contribution that will be detailed, to identify and discuss relevant work.

3.4.1 Flash memory on embedded database systems: characterization and modeling

In the domain of on-flash DBMS performance evaluation in embedded systems, authors of [143] propose Androbench, a tool for benchmarking storage systems of Android based embedded devices. Androbench offers a suite of benchmarks dedicated to SQLite. There exists multiple other Android based SQLite benchmarks. SQLite is used by Android to store system and user databases, so authors claim that their benchmark results are somehow representative of system storage performance in general. In [297], the authors present a set of benchmarks to evaluate and compare performance of file based and DMBS persistent storage system in the context of embedded systems. Benchmarks are launched on a Windows Mobile 6 Professional device using a Microsoft SQL CE embedded database.

Concerning cost models for databases on flash memory, in [296] the author study the overhead in terms of number of flash operations (read and write) added by the FTL during the execution of join queries. This is done by comparing theoretical disk cost of queries to the effective flash cost. Several types of join queries were studied, with different FTL algorithms. A flash-aware cost model is proposed for various types of join queries.

Several studies propose optimizations for on-flash embedded databases. For example, in [291] it is proposed to disable the SQLite journal and to replace it with algorithms for atomic transaction management implemented at the FTL level, to enhance the performance. Authors of [292] present a review on research concerning the impact of flash memory integration in database management. In [294], a DBMS FlashDB is proposed. It is designed for sensor networks using flash storage, and its indexing structures auto-adapts according to the workload.

To the best of our knowledge, the specific problem of modeling the performance for on-flash embedded database managed with FFS (see Part 3, Chapter 2 section 3.5.3) remains unaddressed. We adapted and extended the modeling methodology presented in [296] by targeting specifically FFS based systems and by going further in results analysis. The modeling process is performed using benchmark results.

3.4.2 Revisiting the data sorting problem on flash memory devices

Sorting is one of the most fundamental data processing problems in computer science [300]. While well-studied algorithms like quick-sort and heap-sort are efficient when processing data in main-memory, we need to fall back to external sorting algorithms when the volume of data is too large to fit in the main-memory. Nowadays, as data volumes are several times larger than the main memory capacity, external sorting algorithms relying on secondary storage are increasingly used [301]. They are widely used by DBMS for query resolution and index creation [302].

External sorting algorithms are composed of two phases: a **run generation phase** and **a run merge phase** [301]. The run generation phase splits data into chunks to fit in the main memory, sorts them in memory and writes the sorted chunks into intermediate files named ***runs***. The run merge phase merges the generated runs and writes the overall sorted data into an output file. External sorting algorithms are I/O intensive [304] as they perform several read/write operations to the storage system. Then, their performance highly depends on the way they manage I/O operations [301].

Traditional external sorting algorithms were designed based on HDD performance model. The performance of read and write operations are symmetric, while sequential I/Os are more efficient than random ones. As already discussed, SSDs have been adopted in a wide range of areas thanks to their high I/O bandwidth and short access latency (at the expense of a higher cost). In fact, SSDs offer a new performance model, with nearly symmetric random/sequential read performance and asymmetric read/write performance. This new performance model requires external sorting algorithms to be adapted to take full advantage of the high random read bandwidth while reducing the amount of write operations for performance and endurance sake.

State-of-the-art external sorting optimizations for SSDs commonly follow two optimization paths. The first one consists in modifying the run generation phase in order to reduce the number of write operations. To this end, the authors of [306][307][308] try to directly generate sorted data in the first phase by either relying on efficient random reads [307], [308] to search for minimal values or by scanning the overall file many times [306] with the objective to avoid costly write operations. The second path consists in optimizing the run merge phase by relying on data distribution and using more random reads to reduce the sorting effort (number of comparisons) [309]. All these optimizations take benefit of SSD capabilities mainly by efficient random read operations. Unfortunately this is achieved at the expense of a too high memory utilization or too many additional read operations which represents a real brake for scalability.

3.4.3 Optimizing file prefetching mechanisms at the kernel level for database I/Os

Prefetching for HDDs: Data prefetching on HDDs has been largely studied in the literature. The most popular prefetching mechanism is the sequential one. It is based on spatial locality of I/O workloads. There have been several studies aiming to improve this type of prefetching. Authors of [314] propose to perform aggressive prefetching which is made possible thanks to the growing main memory capacities. In the same way, researchers in [315] mainly proposed a dynamic prefetching mechanism that adjusts the amount of prefetched data during runtime based on the incoming I/O operations. These two studies propose efficient prefetching policies that aggressively load data into memory for HDDs.

Prefetching for SSD: There have been some attempts to improve prefetching mechanisms relying on SSD performance model. FAST [316] focused on shortening application launch time and uses prefetching on SSDs for a quick start up for various applications. It takes advantage of the nearly identical block-level I/O pattern from run to run. This prefetching mechanism is specific to application starting phase and is implemented at the application level. Flashy prefetching [317] is another interesting work. It relies on flash based storage devices (SSDs). The proposed mechanism consists of four stages: (1) the trace collection stage which monitors the I/Os on stored data, (2) the pattern recognition stage that aims to understand the access patterns for a series of I/O requests, (3) the block prefetching stage which periodically moves data from the storage device to the cache, and finally, (4) the feedback monitoring stage that controls the efficiency of prefetching by comparing them with the real application request.

3.5 Contribution

This section describes three of our contributions in the domain of storage optimization for database systems.

3.5.1 Analyzing and modeling the impact of flash memory on embedded database systems

In this section, we will summarize our work on embedded database I/O characterization and modeling for simple SQL queries.

3.5.1.1 Benchmarking and modeling methodology

The objective of this study is twofold: (1) to measure the impact of unitary database requests on flash memory storage system from a performance and lifetime points of views, and to achieve comparative studies on different FFS and (2) model the behavior of each FFS for the tested SQL requests. We focused on two specific FFS: JFFS2 and UBIFS.

Methodology:

Figure 55 illustrates the overall methodology followed in this work. One can observe on the left hand side the tested platform. As discussed earlier, we relied on SQLite database engine for creating our databases and issuing SQL queries.

SQLite has been installed on an embedded Linux platform executing a FFS on top of an embedded flash memory chip. We developed a simple micro-benchmarking tool relying on the SQLite C API to create a database, generate unitary SQL requests and measure the completion time of each request. In parallel, Flashmon was executed in order to profile and trace SQL queries.

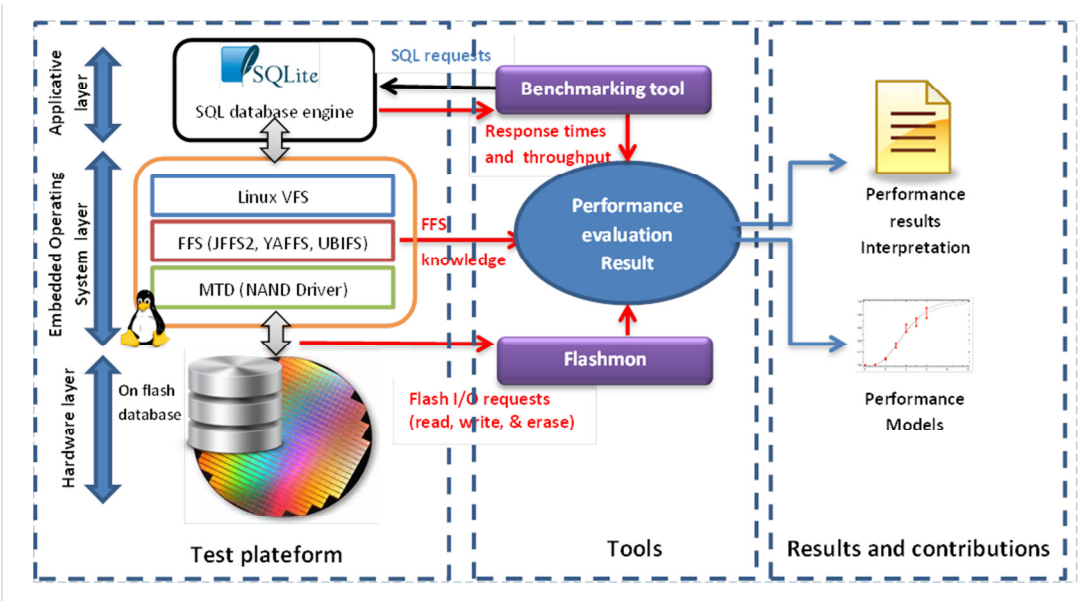


Figure 55. Performance evaluation methodology

We used both quantitative and qualitative information in order to understand system performance. By quantitative measures, we mean SQL request response times, while by qualitative measures we mean the type and number of flash memory operations generated (number of read, write, and erase operations). The use of both information gives more hints to understand the interactions between SQLite engine and the FFS (and more generally the embedded operating system, and the flash memory).

The results of our experimentations can help to: (1) provide the developer with guidelines on how to better use the SQL database engine, which file system to use and how to tune it in order to get better performance. (2) Secondly to model performance of SQL requests with the objective to extract cost models in order to predict performance for a given workload. (3) Finally, to design optimization techniques based on the understanding of the performance behavior and the modeling step (cost models). Optimizations can be applied on different levels: applicative layer (database engine itself) or OS layer (the flash management layer, FFS).

Benchmarking:

Query optimization is necessary in DBMS. The cost of the basic algebraic operations (Selection, Join, etc.) without optimization can be very high, especially when the database size is large, which is the case of data warehouses and scientific databases. In this work we focus on two types of queries: retrieval queries constructed from three basic algebraic operations known as Select-Project-Join queries (SPJ) and update queries involving insert, delete and update operations. In our case, we used tables with (key, value) schema to minimize the meta data overhead, where the key is a short integer and, the value is a fixed size randomly generated character string. In such a schema, the Project queries (retrieve a set of fields from records) are a subset of Select queries (retrieve a set of records) in terms of I/O operations. To be as close as possible to a normal use of a database, we used insert and update queries without delete, where the insert is necessary to build a database, and the update is more frequently used to modify databases. Several research efforts and industrial solutions have been proposed to optimize the basic operations of SPJ queries by offering advanced algorithms (hashing and sorting for join) [290], data structures such as indexes (B-tree, bitmap indexes) [295] and hardware solutions [289]. On the other hand, studying these queries is relevant as they are considered on different available benchmarks (TPC-C, TPC-H, etc.).

We chose to use four types of SQL queries to follow a classical use of database: filling the database after the creation of its schema (INSERT), read data from one (SELECT) or more (JOIN) tables, and modifying data (UPDATE). To simplify our modeling approach, we preferred to use unitary queries without any ordering or grouping constraints in the SQL WHERE clause.

The performed tests are simple, they consist in issuing SQL queries on a given table and measuring the unitary response time of each request. As stated earlier. We first began by generating a simple table that has two fields: a short integer that is the primary key and a fixed size character string. This string contained randomly generated data to avoid data compression at the FFS layer, and was fixed for each test set. Two parameters were varied for each SQL request: the number of records sustaining SQL queries and their size (the table size). Each set of experimentations for a given SQL request was preceded by a BEGIN TRANSACTION instruction and ended by an END TRANSACTION instruction. It is recommended [298] to use those two instructions especially in embedded systems, for instance, to prevent data corruption caused by power failures in the middle of a database transaction. This is implemented throughout the creation of a journal log file that is specific to a given transaction and that is removed once the transaction succeeds.

Algorithm 1 Micro-benchmarking algorithm

```
1:  Input:
2:  Number of requests: Nb_Req
3:  Record size: Rec_size
4:  Query type: Q_type = insert | select | join | update
5:  File system: FS = JFFS2 | UBIFS
6:  Output
7:  Response time of all requests: RT[Nb_Req]
8:  Flashmon output (I/O trace) for the experiment: IOtrace
9:  Initialization()
10:     Format and mount test partition with FS
11:     Reset I/O tracer Flashmon
12:     Create database schema
13:     if Q_type == select | join | update then
14:         Fill database with random data
15:     end if
16:     Empty all system caches
17:     Initialize I/O tracer Flashmon
18: Main Function()
19:     Initialization()
20:     RT[0] = responseTime(BEGIN TRANSACTION)
21:     for i ← 1to Nb_Req
22:         RT[i] = responseTime(request i of Qtype, with Recsize)
23:     End for
24:     RT[Nb_Req + i] = responseTime(END TRANSACTION)
25:     return IOtrace & RT
```

The database benchmark described in the preceding algorithm was executed on both UBIFS and JFFS2, for SQL queries with record sizes of 150, 300, 450, and 600 bytes, and a number of requests going from 100 to 5000 with an increment of 100. Record sizes were chosen according to SQLite page size. Indeed, the default size of an SQLite page in Unix-like systems is 1024 bytes, where 124 bytes are reserved to headers and an unallocated area, and 900 bytes to store records' data. So the record sizes were chosen to highlight the effect of SQLite page fragmentation

3.5.1.2 Theoretical cost model and modeling methodology

The cost model elaborated in this work models the performance by taking as input various parameters related to the workload, embedded database and DBMS characteristics, NAND flash memory chip performance and the FFS executed by the OS. The outputs of the cost model are the number of flash basic operations (flash page read / write / block erase) generated and the execution time. As our micro-benchmarks results will reveal, the FFS has a strong impact on the database operation performance, so our study particularly focuses on this software layer.

Our cost model building methodology is based on investigating the ratio between the **theoretical cost** of a query, and their **effective cost**. This ratio is measured through micro-benchmarking. This methodology was first used in [296], where the author presented the cost model as the ratio of the additional cost that arises from the use of an FTL. We extended this methodology to embedded flash memory based systems with SQLite DBMS and FFS. Note that the theoretical cost model presented in this section can apply for both FFS and FTL based systems.

The *theoretical cost* of a SQL operation is the number of flash accesses one would intuitively expect to be generated. The theoretical cost corresponds to the size of data accessed by the SQL operation at a logical (operating system) level, translated into flash storage unit (pages or blocks). For example, if a select operation targets 30 records, each of 100 bytes, the total logical space read is expected to be 30.100 = 3000 bytes. Using a flash chip with 2048 bytes sized pages, one would expect to observe 2 page read operations on the flash chip. The theoretical read cost for this SQL operation is then 2.

We make the following assumptions:

- The *select* and *join* operations only generate flash page read operations. The theoretical cost of these types of SQL queries is denoted *np* in the rest of this section: we expect *np* flash pages to be read in case of a select operation;
- The *insert* operations generate flash pages write operations and may generate erase operations under some conditions (for instance if the flash memory contains a small number of free space and the GC must be launched). We expect *np* flash pages to be written, and *nb* blocks to be erased ;
- The *update* operation generates flash read, write and erase operations. We expect *np* flash pages to be read, *np* flash pages to be written (reading the same amount of data before updating them), and *nb* blocks to be erased.

As opposed to the theoretical cost, the effective cost of a SQL operation is the number of I/O accesses actually generated on the flash chip when the SQL operation is executed on the system. At a high abstraction level:

$$SQL(W, Sys) \; is \; defined \; by \; \{ n_r, n_w , n_e \} \tag{49}$$

W and *Sys* represent all the parameters impacting the number of flash accesses generated. *W* denotes the characteristics of the SQL query workload: a set of operations defined by mainly their type (insert, select, etc.), the size and number of records accessed. *Sys* is related to the environment/system in which this operation/workload was launched: database accessed, DBMS parameters, but also operating system related parameters. In particular, the FFS type / parameters, and system cache parameters. Moreover, *Sys* also includes parameters about flash memory state in terms of free / invalid / valid space. n_r, n_w, n_e represent respectively the number of flash page read, flash page write and flash block erase operations generated (effective cost of the SQL operation). These are the output values of our model, expected to be computed from the various input parameters represented by *W* and *Sys*.

SQL operations (*OPi*) we are interested in can be unary (applied to one table), or binary (applied to two tables). A workload (*W*) is a set of unary and/or binary operations:

$$W \; is \; defined \; by \; \; \{ OP_1, OP_2 , \dots , OP_n \} \tag{50}$$

The global equation used to compute the execution time is the following:

$$T_{tot} = T_{flash}.SQL(W, Sys) + \; T_{Other}.SQL(W, Sys) \tag{51}$$

Where $T_{flash}.SQL(W, Sys)$ is flash memory operation execution time and $T_{Other}.SQL(W, Sys)$ represents other elements impacting the SQL operation execution time: algorithm execution by the CPU, operations in RAM, etc. The investigation of this term is out of the scope of this study, we only focus on the $T_{flash}.SQL(W, Sys)$ term.

The main idea of the modeling methodology we used is that for each type of flash access, the effective cost can be represented as the multiplication between the theoretical cost (*np* and *nb*) and a system related coefficient for flash read (α), flash write (β) and flash erase (γ) operations:

$$n_r = \; \alpha . np \; ; \;\; n_w = \; \beta \; . np \; ; n_e = \; \gamma \; . nb \tag{52}$$

As stated earlier these amplification coefficients mainly represent the FFS overhead, and indicate its performance in the context of our study. For instance if the application needs to read one database page (for example in the case of a select operation), and we observe that it ends up in the FFS generating two flash memory page read operations, then coefficient α equals 2. α, β, and γ are FFS related metrics. These coefficients are not constant and their values are impacted by various parameters related to *Sys*.

We defined α, β, and γ as functions because their values may change according to some parameters such as the number of performed queries. These functions represent **the way the effective cost evolves according to the theoretical cost values and the parameters related to Sys.** These functions are investigated through micro-benchmarking results on which we applied some linear regression. Finding those functions represent the main contribution of this work. Equation (53) can then be written as follows:

$$SQL(W,Sys) \quad is\ defined\ by \quad \{np\,.f_{\alpha}(W,Sys), np\,.f_{\beta}(W,Sys)\,, nb\,.f_{\gamma}(W,Sys)\} \tag{53}$$

Once α, β and γ functions obtained, we are able to compute the effective cost of SQL operations. It is then possible to calculate the flash memory response time of these operations as follows:

$$T_{flash}.SQL(W,Sys) = t_r.np\,.f_{\alpha}(W,Sys) + t_w.np\,.f_{\beta}(W,Sys) + t_e.nb\,.f_{\gamma}(W,Sys) \tag{54}$$

t_r, t_w and t_e are respectively the time to read / write a flash page, and to erase a flash block. They can be measured from simple benchmarks, or extracted from flash chip data sheets. This equation gives the total flash memory response time of a test in terms of: (1) basic response times measured from the tested platforms which are hardware and operating system related, (2) the number of pages calculated from the characteristics of the used DBMS and the workload which is application and DBMS related, and finally 3) the operation coefficients which are mostly file system related

3.5.1.3 Result example, the case of the `SELECT` query

In this section, we give some elements on the performance of the select query operation. Other queries were investigated and results can be found in [ri12].

The benchmarking phase

Before performing the tests, the database was filled through a set of insert queries. For each record size, we defined a fixed size database according to the maximum requests it handles.

A first observation one can make when looking at Figure 56 representing the throughput of the select query for different record sizes and request numbers, is that UBIFS slightly outperforms JFFS2 for the select operation for small selected volumes. The throughput follows a logarithmic shape for both file systems (and for both tested platforms) due to the constant overhead related to the log journal manipulation and other SQLite specific processing and memory usage.

Response time analysis (see Figure 57) shows more stability for UBIFS response times as compared to JFFS2. In fact, for UBIFS, we can observe periodic small peaks (around 2.2ms) which period decreases according to the request size. The peaks have values that are approximately double as compared to the other ones. For JFFS2, the amplitude of the peaks is much higher (around 30ms). The period of those peaks is also much larger and depends on the request size. In fact, this phenomenon (in JFFS2) is related to Linux page cache read-ahead algorithm that prefetches chunks of 128KB (around 32ms, giving 4MB/s throughput which approximately corresponds to the measured throughput on this platform) of data, allowing future SQLite read accesses to be satisfied from RAM rather than flash. This behavior is only observed for JFFS2 as in UBIFS the read-ahead mechanism is disabled by default [299]. The NAND driver (MTD) is fully synchronous, so in the best case read-ahead does not impact flash memory I/O performance. In the general case, read-ahead leads to a performance drop on flash storage, because all the prefetched data is not necessarily accessed in the future. This is the reason why it is disabled in UBIFS, which explains the performance difference between the two tested FFS.

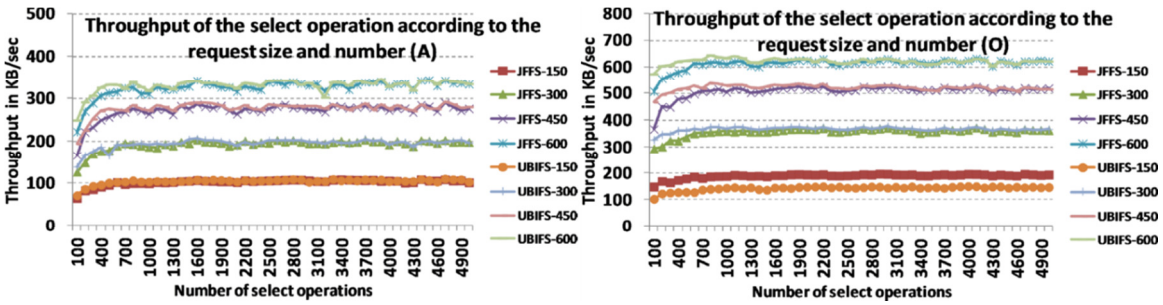


Figure 56. I/O throughput (KB/s) of select queries according to record size and number of queries for two platforms (O: OMAP 3530 and Armadeus, see [62] for more details).

Even though UBIFS disables the read-ahead mechanism of the VFS, one can observe that the period of the peak response times is 4. In fact, this is related to the I/O system page granularity on Linux which is 4KB.

Query nb.	JFFS2 (req. Of 600 B) Time (ms)	UBIFS (req, Of 600 B) Time (ms)
0	*26.99*	*28.14*
43	1.75	**1.93**
44	1.35	1.36
45	1.37	1.61
46	1.35	1.36
47	**29.78**	**2.20**
48	1.36	1.37
49	1.62	1.38
50	1.45	1.47
51	1.37	**2.14**
52	1.36	1.38
109	1.41	1.40
110	1.39	1.37
111	**30.00**	**2.21**
112	1.40	1.41

Figure 57. Snapshot of response times (ms) of select operations for 600 Bytes objects on JFFS2 and UBIFS

When observing the number of accesses (here I/O reads) in Figure 58, one can see the impact of the read-ahead prefetching algorithm. In fact, the peak values in Figure 58 (for JFFS2, especially for 150 B records) represent the case where a data prefetching (of 128KB) has been made but was not profitable as prefetched data were not used, thus generating more flash reads as compared to the needed data. In addition to the read-ahead related behavior, under JFFS2, the system performs more read operations especially for a number of requests smaller than 1300 select operations. This is coherent with the I/O throughput observed in Figure 56 but one could expect a larger difference in the throughput when looking at the number of I/O reads generated for both file systems. In fact, the flash memory I/O access times account for a considerable percentage of the total execution times but SQLite processing and memory activity should not be ignored. They represent a large part of the execution time especially for small number of requests. For the example shown in Figure 57, in case of JFFS2, experiments show ˜32ms response time each 64 requests. So if we consider a period of 64 requests, around 32ms are related to flash memory management while ˜(1.3*63)ms are due to SQLite processing (memory and CPU). This illustrates the significant overhead related to non-I/O operations.

The modeling phase

The select query generates only read operations, thus we do not have to model the write and erase operation coefficients. As observed from the result analysis part, the behavior of the α coefficient is independent from the platform. In order to evaluate the system performance executing a select query, the performance equation (55) is changed as follows:

$$T_{flash}.SQL(W,Sys) = t_r.np.f_\alpha(W,Sys) \quad (55)$$

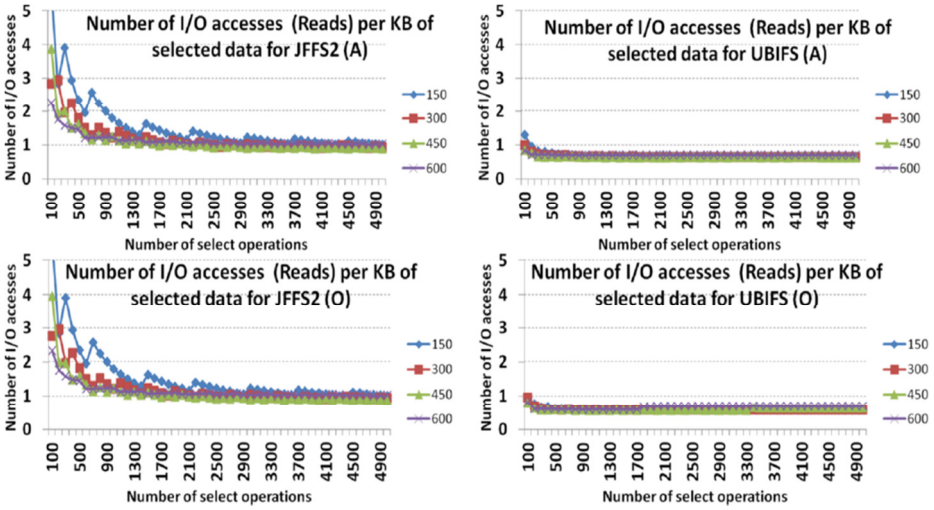


Figure 58. Number of I/O operations to the flash memory for select queries according to record size and number of queries, for two platforms.

In Figure 59 we observe that the read coefficient *a* has a different pattern for the two file systems. We performed model fitting on *a* and we observed that a power model is the most accurate one in this case. So the generic model for the coefficient is $f(x) = a\,.\,x^b$ or in our case:

$$f_\alpha(W, Sys) = a\,.\,Nb_{rec}^b \tag{56}$$

Where Nb_{rec} is the number of accessed records (x axis).

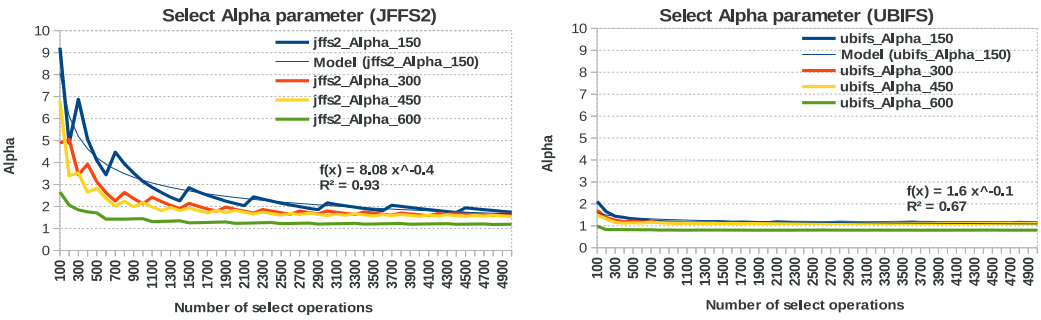


Figure 59. α parameter model for the select queries

For JFFS2, one can observe from Figure 60 that *a* decreases and *b* increases when increasing record sizes meaning more stability and better performance. The same thing is observed for UBIFS. However, UBIFS shows smaller *a* and higher (closer to zero) *b* values (more performance and stability) as compared to JFFS2.

In the last column of Figure 60, we show the R^2 values of the proposed models for the select query. This value is around 0.9 for JFFS2 and around 0.6 for UBIFS. The small value of R^2 for UBIFS mainly corresponds to coefficients that appear to be constant for large numbers of operations.

Summary on the modeling phase

From the modeling step performed for the four studied SQL queries, see [62], one can draw the following conclusions:

- The behaviors of different FFS (and operating systems) reflected by the α, *β*, and *γ* coefficient are independent from the platform and are FFS specific ;
- The α and *β* coefficient have proved to follow a power model for most cases and are sometimes constant ;
- The developed models depend on the query sizes. This is due to many reasons such as file system overhead, internal fragmentation of the SQLite pages, meta data size, etc.

Flash File System	Records size (byte)	Model of α	R^2
JFFS2	150	$8.084\,x^{-0.403}$	0.930
	300	$4.795\,x^{-0.298}$	0.913
	450	$4.245\,x^{-0.279}$	0.861
	600	$2.143\,x^{-0.164}$	0.872
UBIFS	150	$1.596\,x^{-0.098}$	0.672
	300	$1.377\,x^{-0.060}$	0.642
	450	$1.261\,x^{-0.039}$	0.528
	600	$0.864\,x^{-0.022}$	0.456

Figure 60. α Model for select operations. x is equal to the number of records divided by 100.

3.5.1.4 Conclusion

This study presents two contributions: (1) a micro-benchmarking methodology and the associated results on basic SQL queries performance according to different parameters related to the DBMS, workload, operating system (and more specifically the FFS) and embedded platform. (2) A cost model for I/O operations on the flash memory.

The micro-benchmarking part revealed very high disparities according to the different parameters that we varied. Each of the tested SQL query (insert, select, join, and update) showed some specific performance patterns, except the select and join that proved some similarities. We tested different workloads in terms of number, types and sizes of queries and observed that the performance highly depends on both. For the given test configuration, the insert operation showed mainly write operations, the select and join, only read operations, while the update showed read, write, and erase operations. The FFS type is also a determinant factor for both performance patterns and values. Flash memory raw performance also influenced tests results.

In the modeling phase, we have elaborated a generic model for flash I/O operations that relies on FFS specific coefficient. Our modeling part focuses on flash specific coefficients relative to JFFS2 and UBIFS named α, β, and γ. The α and β coefficients were modeled whether thanks to a power model or as constants. Their multiplicative and power values were different according to the query size and FFS. The γ coefficient applied to the erase operations was modeled with a logarithmic model.

The models we have elaborated can help designers to: (1) optimize their workloads by selecting the most suitable DBMS configuration, or (2) tune the way to query the DB (query/data size), (3) optimize the performance by selecting the most suitable FFS, according to application constraints. The knowledge extracted from these models can be integrated in database optimizers in order to better select the optimization technique according to the effective impact on storage systems, this can be, for instance, a better query scheduling.

Models elaborated in this work could be extended to FTL systems with the help of some tools allowing to quantify the number of operations (read, write, and erase) generated upon the micro-benchmark execution. This is a very interesting perspective as FTL based flash memories are now the *de facto* form factor in embedded systems (eMMC interface).

For future work, it would be interesting to investigate more SQLite complex queries such as joins and validate the obtained results. The Select-Project-Join queries with aggregation functions (SUM, AVERAGE, MAXIMUM, etc.) can also be a good target for optimization, as those queries would have more processing related to I/O operations, which highly impacts the system performance.

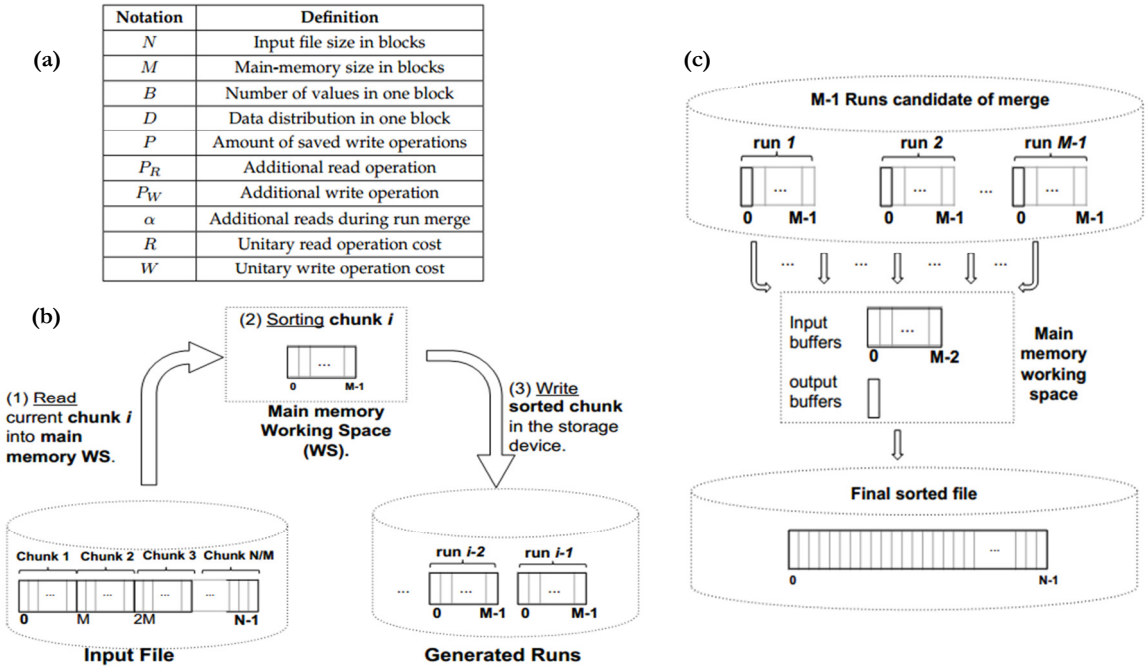
3.5.2 Revisiting the data sorting problem on flash memory devices

In this section we will summarize our work on data sorting optimization for flash memory based devices.

3.5.2.1 Background

In order to describe external sorting algorithms, we use the notation summarized in Figure 61-a. We consider an input file containing n values. These values are grouped in blocks of B elements. We denote by N the number of blocks in the file. We consider the main memory working space as the amount of memory allocated by the DBMS to the sorting operator. The main memory working space can contain M blocks, thus storing m values. D denotes the distribution of

data in the blocks of the input file. As in [308], the distribution of data within a block is given by the number of different values in that block. Finally, R and W denote the unitary read and write costs respectively.



Notation	Definition
N	Input file size in blocks
M	Main-memory size in blocks
B	Number of values in one block
D	Data distribution in one block
P	Amount of saved write operations
P_R	Additional read operation
P_W	Additional write operation
α	Additional reads during run merge
R	Unitary read operation cost
W	Unitary write operation cost

Figure 61. (a) notation table, (b) illustration of the run creation phase, (c) illustration of the run merge phase.

Mergesort [300] is one of the most used and studied external sorting algorithms in the literature. Chunks of data from the input file handled during the **run generation phase** have to fit in main memory working space. Thus, each chunk contains M contiguous blocks. Chunks are processed successively (see Figure 61-b). They are first loaded into main-memory working space (Figure 61-b (1)), sorted using an in-memory sorting algorithm (Figure 61-b (2)) and finally written to the run, an intermediate file in the external memory (see Figure 61-b (3)). The total number of generated runs is equal to $\frac{N}{M}$. The run generation phase results in N read operations on the input file in order to load data and N write operations for writing the generated runs.

The **run merge phase** consists in merging the $\frac{N}{M}$ runs created during the run generation phase (see Figure 61-c). A mergesort algorithm is commonly used for this task. The number of runs to be merged at once is limited by the size of the main memory working space *M*. The mergesort algorithm needs $M-1$ buffers as input buffers and uses the remaining buffer as an output buffer. Each input buffer stores a block from a different run and the output buffer stores the resulting merged data to be written to the output file.

When the number of generated runs $\frac{N}{M}$ exceeds $M-1$, the merge process needs $\lceil log_{M-1}{}^{N}/_{M} \rceil$ passes to generate the final sorted file from all the generated runs. Thus, the run merge phase will result in $N.\lceil log_{M-1}{}^{N}/_{M} \rceil$ read and write operations, and the total I/O cost of the traditional mergesort algorithm is given by:

$$Cost_{IO} = N.(R+W) + N.\lceil log_{M-1}{}^{N}/_{M} \rceil.(R+W) \quad (57)$$

Where the first expression is related to the **run generation phase** while the second one is due to the **run merge phase**.

3.5.2.2 MONTRES: Merge ON The Run External Sorting

In this section, we describe the external sorting algorithm for SSD performance model we have proposed, namely MONTRES. Similarly to traditional mergesort algorithm, MONTRES consists in a run generation and a run merge phase. The run generation phase of MONTRES includes three main optimizations: (1) an ascending block selection algorithm for an optimized run creation, (2) a continuous run expansion policy, and (3) a merge on-the-fly mechanism during the run generation phase:

1) **Ascending block selection**. The objective of this optimization technique is to create sorted runs and evict as much data as possible to the final sorted file during the run generation phase. To do so, we rely on random reads in order to select blocks in ascending order according to their minimal value for run creation step. Doing so allows first to evict minimal values directly to the sorted file, and makes the following two optimizations possible.
2) **Continuous run expansion policy**. The objective of this optimization is to create runs as large as possible. Doing so helps to reduce the run merge phase as seen earlier. So, rather than separately creating each run from its respective chunk in the run generation phase, we continuously expand already created runs when possible with large values from ongoing chunks.
3) **Merge on-the-fly mechanism**. The objective of this optimization is again to reduce the run merge phase effort by reducing the number of values to merge. This is done by detecting and then evicting smallest values from previous runs directly to the sorted file during each new run creation.

Thanks to the abovementioned techniques, MONTRES first reduces the amount of intermediate sorted data written to the runs and second, expands the size of runs, thus decreasing the run merge phase I/O cost.

During the run merge phase, MONTRES was designed to proceed in a single pass rather than $\lceil log_{M-1}{}^{N}/_{M} \rceil$ passes. This is achieved by continuously retrieving minimum values from the generated runs and outputting them to the final sorted file at the expanse of some additional, yet reasonable, number of read operations.

3.5.2.3 Run generation algorithm

Ascending block selection

The ascending block selection mechanism starts by building an index (called **min-index**) on the minimum values contained in each block of the input file. To do so, MONTRES performs a first scan of the whole input file in order to detect the smallest values of each block, giving *N* values. The created index is then sorted in ascending order of minimum values of the contained blocks.

Once the index created, chunks are successively loaded and sorted in main-memory. Rather than reading blocks sequentially from the input file, the block selection is operated according to min-index beginning with blocks containing minimum values. By doing so, we seek two main optimizations: (1) first, as minimum values are already detected, they can be directly evicted to the final sorted file, (2) then, we rely on data values locality and assume that if a block contains a small value, there is a high probability that it contains neighboring (also small) values, that can therefore be also evicted into the final sorted file directly.

The loaded data are sorted using an in-memory sorting algorithm creating a sorted chunk. Then, sorted values in main-memory that are smaller than the next minimal value from the next block of the min-index (called ***next_min_value***) can be directly evicted into the sorted file while higher values are written to the current run. Note that for optimizing I/Os, the eviction is achieved at the end of the phase together with merge on-the-fly that will be discussed later. The next example gives an overview of this algorithm.

Example 1. Figure 62-a shows an example of a file to sort. The input file contains $N = 8$ blocks where each block stores $B = 4$ values. The main memory working space is limited to $M = 2$ blocks. First, file blocks are scanned sequentially in order to retrieve the smallest value for each block. These minimum values associated with corresponding block ID are used to create the **min-index** presented in the right side of Figure 62-b. Block5 and Block4 containing the lowest values in the min-index are loaded to the main memory working space (operation (**a**) in Figure 62-b) to form a chunk of data in main-memory. This chunk is sorted using an in-memory algorithm. Values in the sorted chunk lower than or equal to the next lowest value in the min-index (here *next_min_value* = 8) are evicted to the final sorted file (operation (**b**) in Figure 62-b). The remaining values are written to the storage device to form a sorted run (operation (**c**) in Figure 62-b).

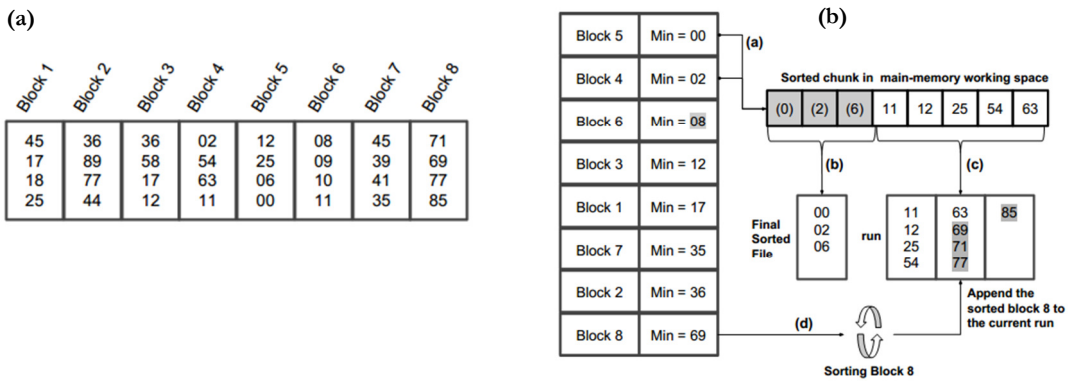


Figure 62. (a) example of a file content to sort, (b) building the min-index

Continuous run expansion policy

As previously discussed, the complexity of merge performance largely depends on the number of generated runs. The proposed run creation policy aims to reduce the number of generated runs by expanding their size.

Once the current run generated, MONTRES tries to retrieve from min-index blocks having all values higher than the maximum value in the current generated run. Retrieved blocks are loaded into main memo ry, sorted using an in-memory algorithm, and finally appended to the generated run.

Example 2. In the previous example, all values of Block 8 are higher than the maximum value of the generated run (which is 63, see Figure 62-b). So, Block 8 is loaded into the main memory, sorted, and the resulting block is appended to the run (operation (**d**) in Figure 62-b).

Merge on-the-fly mechanism

As previously described, for each iteration of the run generation algorithm, thanks to the ascending block selection mechanism, values in the main memory that are lower than or equal to *next_min_value* are expected to be evicted to the final sorted file. Before this eviction takes place, those values are merged with values of all previous runs that are also lower than *next_min_value*. After this, the effective eviction takes place. So, values to merge from the ascending block selection and merge on-the-fly mechanisms are grouped and evicted altogether to optimize I/O cost. By doing so, MONTRES reduces the amount of data candidate to the merge phase which decreases the merge complexity.

Example 3. Figure 63 shows the processing of the second chunk from the min-index and the input file. The next M = 2 blocks from the min-index, Block 6 and Block 3, are loaded to the main-memory working space, then sorted. Values in the sorted chunk that are lower than or equal to next_min_value = 17 are merged on-the-fly (operation (f)) with values 11 and 12 from previous run, and directly evicted to the final sorted file (Operation (g)). Note that if there were more runs already sorted, MONTRES would have looked for values smaller than *next_min_value* = 17 in all of them.

Preparing the Run Merge Phase

In order to prepare for the run merge phase, blocks of the intermediate runs are indexed according to the minimum value they contain (in an ascending order) and their position in the run (a run may have several blocks). This index is created when outputting runs into the storage device. We name this data structure **run-index**.

To conclude this first phase, the run generation phase of MONTRES algorithm results in the following three outputs:

- **Sorted data**: Part of the data will be stored into the final sorted file.
- **Sorted runs**: Another part will be stored into runs still to be merged.
- **run-index**: Index of blocks belonging to the runs sorted in the ascending order according to minimal values.

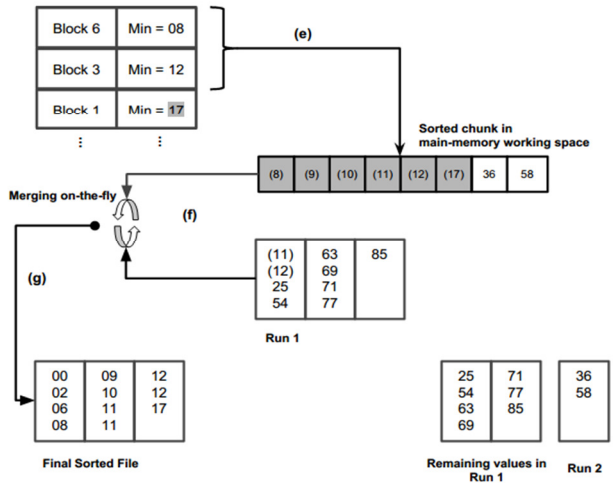


Figure 63. Example of merge on-the-fly

3.5.2.4 Run merge phase

The run merge phase aims to produce the final sorted file going from the intermediate runs. With the traditional run merge algorithm, several merge passes $\lceil log_{M-1}{}^{N}/_{M} \rceil$ are required to generate the final sorted file when the number of runs generated $\frac{N}{M}$ exceeds the number of input buffers in main memory ($M-1$).

Instead of performing several merge passes, MONTRES relies on the created **run-index** to load minimal values in the main memory working space by performing random read operations on the run files. Then, those values are evicted to the sorted file and this process carries on until there is no more values to merge. The run merge is realized in one pass at the expense of some additional read operations.

Min-max heap

The merge process sorts runs' data using a tree based data structure called a heap [310]. Nodes of the data structure obey the min-max heap property: each node at an even level in the tree has a value smaller than all of its descendants, while each node at an odd level in the tree has a value larger than all of its descendants. We rely on this data structure because it allows retrieving minimum and maximum values with a constant complexity.

Data merge processing

The merge process starts by selecting $M-1$ blocks (1 is left for the output) which contain the smallest values from the run-index. These blocks are then read from intermediate runs and minimum values are inserted into the tree data structure.

Once the data inserted in the tree, MONTRES retrieves from the tree all values smaller than the next minimum value in the run-index, and appends them on the final sorted file. When all those values are written into the final sorted file, MONTRES needs to load the next block(s) (according to run-index) to carry on the merge process.

If one or several main-memory buffers are fully emptied thanks to the merge process, next blocks in the run-index are loaded into the free buffers and inserted into the tree. If there is no buffer available, two options are possible:

- When two input buffers have at least less than $B/2$ values, these values are merged in main-memory using mergesort algorithm to free one input buffer. This allows to compact the content of main memory working space and frees some input buffers.
- When the first option is not possible, the buffer containing the maximum value in the tree is dropped from main-memory. A block containing next minimum value is then loaded from the storage device into the main-memory.

Example 4. Figure 64 illustrates an example of run merge phase of MONTRES with 4 runs generated using 3 input buffers. Each run is composed of many blocks (2 are shown in the figure), and each block contains 4 values. The run index describes the distribution of values in the runs. It is sorted in ascending order according to minimum values. The

run merge process starts by reading from runs three blocks having the lowest value in run-index (Block 1−1, Block 3−1 and Block 2−1). These blocks are inserted into the tree, then all minimum values lower than or equal to the next minimum value in the run-index (value 14 from Block 4−1) are outputted from the data structure into the final sorted file. Once done, MONTRES tries to load the next block of the run-index. As no input buffer is emptied, the algorithm frees two input buffers by successively merging half free input buffers

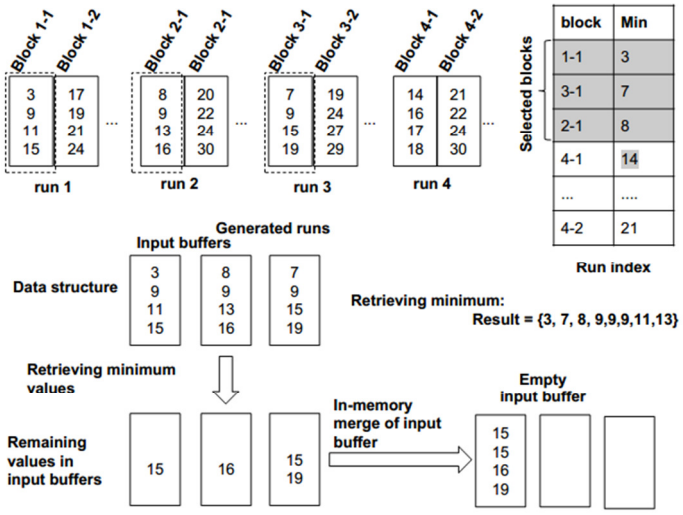


Figure 64. Example of run merge phase with 4 runs and 3 input

3.5.2.5 Some experimental results

This section describes some of the experimentations achieved to validate the efficiency of MONTRES as compared to state-of-the-art algorithms. The rest of the results can be found in [64].

Experimental Methodology

We measured the performance of MONTRES by varying three main parameters: (1) the distribution of data, (2) the allocated main memory working space for sorting to have different values of $\frac{N}{M}$ and (3) the SSD model, we experimented with three different classes of SSDs (PC SSC, a Data Center SSD, and a Server SSD). The performance of MONTRES was compared with many algorithms, two (the best performing ones) were selected for this evaluation: the traditional External mergesort and FSort [303].

We evaluated the total execution time of the sorting process for each algorithm.

We used the TPC-H benchmark to generate experimental datasets. We experimented with two different datasets. The first dataset (A) was obtained from the column Quantity of the Lineitem Table. It contains a large value interval and few unique values. The second dataset (B) is obtained with the year information from the Date column of the Orders Table. This dataset contains smaller value interval, thus more equal values.

Scalability is one of the key issues that motivated the design of MONTRES. Indeed, external sorting algorithms should be efficient even with a growing $\frac{N}{M}$ ratio. Sorting operators in DBMS usually compete for main memory with other operators and other requests. Therefore, main memory working space allocated for sorting is limited while input data to sort are growing larger. In order to evaluate MONTRES with different values of Ri = $\frac{N}{M}$. We chose to experiment with 8GB data size (N) on different amount of main memory working space (M). We used in the experiments 7 different values of Ri=$\frac{N}{M}$: *R*1=8, *R*2=32, *R*3=64, *R*4=128, *R*5=512, *R*6 = 2048 and *R*7=8192. While most state-of-the-art studies, like [309], experimented with $\frac{N}{M}$ lower than 512, we chose to experiment with higher values (2048 and 8192) to prove the scalability of MONTRES as compared to state-of-the-art algorithms.

Experimentation Results

Results with Dataset (A): Figure 65 shows MONTRES and FSort execution time speedups for dataset (A) with different ratios of $\frac{N}{M}$ using the experimented SSDs. The figure shows that in the best case, MONTRES algorithm accelerates the external mergesort by 25% while FSort enhances it by a ratio of 8% for $\frac{N}{M}$ = 8192 (*R*7). The worst case of MONTRES occurs when the ratio $\frac{N}{M}$ is very low, in case of $\frac{N}{M}$ = 8 (*R*1), MONTRES algorithm improves the external mergesort only by 5%.

One can observe that MONTRES increases the run generation phase I/O cost by 4% on average, considering all SSDs, when $\frac{N}{M}$ > 64 (*R*4 and more) and speeds up the run merge phase by 20%, while FSort overloads the run generation phase by 71% on average and enhances the run merge phase by 19%. When $\frac{N}{M}$ = 8, 32 and 64, MONTRES speeds up the run generation by 2% on average in addition to the speedup of run merge which is about 7% on average.

MONTRES run generation overhead is due to additional read/write operations performed by the ascending block selection and merge on-the-fly mechanisms. FSort run generation overhead is mainly due to additional CPU operations performed to create larger sorted runs. We can also notice that the narrower is the main-memory working space (higher $\frac{N}{M}$), the lower is the run generation execution time and the higher is the run merge execution time. When the main memory working space is large, MONTRES merge on-the-fly mechanism processes then produces more data to the final sorted file. This increases the run generation overhead and enhances the run merge phase. On the other hand, with larger working space, FSort sorts more data during the run generation which results in larger runs. This increases the run generation phase I/O cost but accelerates the run merge phase.

MONTRES outperforms FSort during the run merge phase because it reduces the amount of intermediate data to merge and performs the run merge phase in a single pass while FSort does in $\lceil log_{M-1}{}^{N}/_{2M} \rceil$ passes.

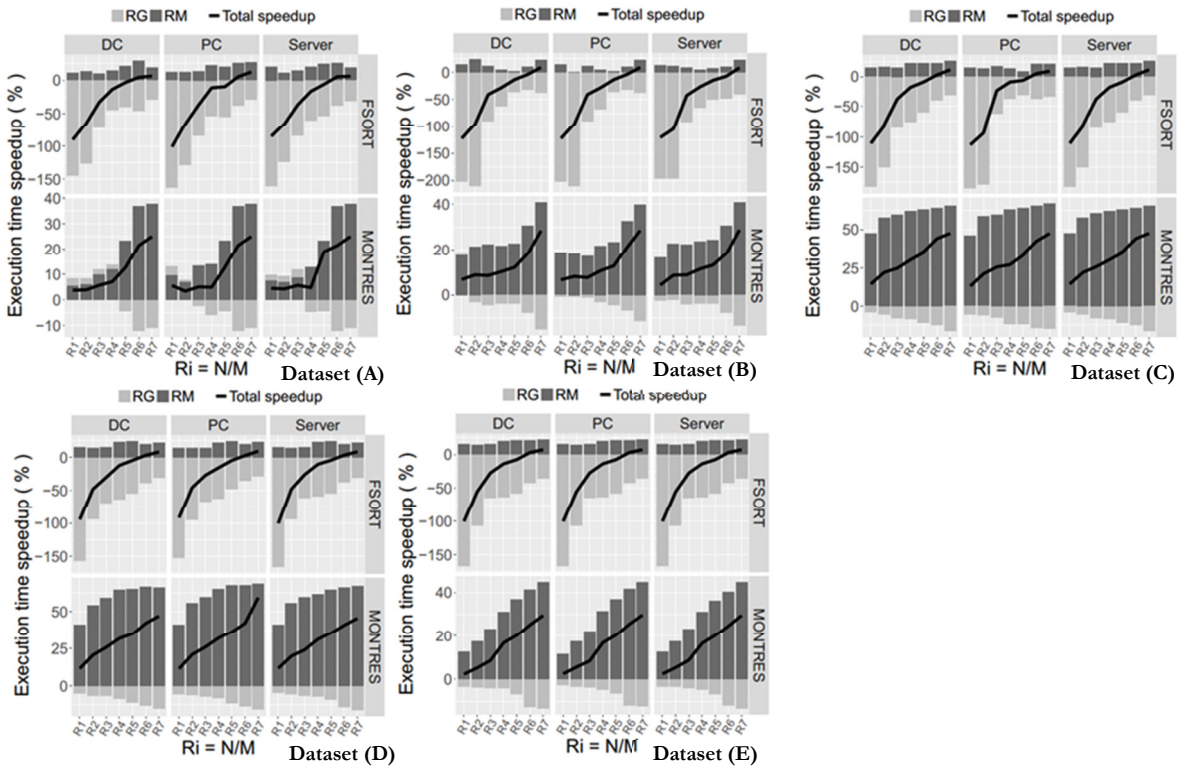


Figure 65. Execution time speedup for MONTRES and FSort as compared to mergesort for Datasets (D), (E) for DC, PC.

Results with Dataset (B): Figure 65 also presents the execution time speedup of experimented algorithms when sorting dataset (B) with different ratios of $\frac{N}{M}$. One can observe that MONTRES speeds up the external mergesort by 28% while FSort decreases the execution time by 8% for $\frac{N}{M}$ = 8192 (*R*7). Similar to the results observed with dataset (A), both MONTRES and FSort increase the run generation phase I/O cost and speed up the run merge phase.
We have also noticed that MONTRES execution times are lower when sorting dataset (B) as compared to dataset (A). In fact, as dataset (B) contains more redundant values, the merge on-the-fly mechanism outputs more data into the final sorted file and reduces the amount of intermediate data candidate to the run merge phase.

The amount of data written into the final sorted file at the end of the run generation phase is higher with dataset (B) than with dataset (A). Both datasets have a random distribution, but dataset (B) has a smaller interval of values resulting in higher density of neighboring values within a block. This allows the merge on-the-fly mechanism to output more data values into the final sorted file. Similar to the dataset (A), the higher the ratio $\frac{N}{M}$, the better the performance of MONTRES as compared to mergesort and FSort.

More results about MONTRES are available on the published paper [ri1].

3.5.2.6 Conclusion

Due to the exponentially growing volume of data generated, sorting algorithms need to adapt to be more scalable when it comes to dealing with large datasets under memory pressure. The idea behind MONTRES is to reduce the amount of intermediate data written to the storage device by continuously retrieving minimum values from the input file and evicting them directly into the final sorted file. We also tried to smooth the growth of the number of I/O operations generated with regards to the increasing $\frac{N}{M}$ ratio.

We evaluated the I/O cost of MONTRES with different ratios, data distributions, and SSDs. MONTRES outperformed traditional algorithms by reducing execution times by more than 40% when $\frac{N}{M}$ is high. MONTRES will be integrated in a commercial DBMS.

As a perspective, it would be interesting to study the impact of page cache mechanisms on different sorting algorithms to see how to take benefit of prefetching and write back algorithms tailored for SSDs to reduce more the I/O cost (see next section). We also would like to investigate such sorting algorithms considering one level of emerging Non-Volatile Memory (NVM) that have yet another performance model (see the next chapter). We have already begun exploring this topic with promising results.

3.5.3 Optimizing file prefetching mechanisms at the kernel level for database I/Os

3.5.3.1 Motivation

Data-intensive applications have increasing needs in I/O performance. They are continuously optimized to take advantage of new evolutions of the memory hierarchy. Traditional Linux read-ahead prefetching mechanism design is based on two main assumptions: the sequential/random performance asymmetry of HDDs due to mechanical latencies, and the spatial locality of I/O workloads [318]. There are two main reasons that motivate adapting the Linux read-ahead. First, as described earlier, random reads are used more and more in current data-intensive applications [312][313] such as data analytics, modern DBMS [319]. Second, SSDs have similar latencies for both sequential and random read operations.

We designed Lynx, a new prefetching mechanism for Linux system. Lynx design aims to take advantage of the similar random and sequential read performance on SSDs in order to satisfy all types of I/O patterns for data-centric applications. Of course, predicting fully random workloads is not feasible. We assume that applications expose algorithmic locality [311], meaning a repeated I/O pattern/sequence that is not necessarily sequential on disk.

3.5.3.2 Lynx prefetching mechanism: design and implementation

The use of Markov chains

Machine learning is about making computers modify or adapt their actions so that these actions get more accurate [320]. The accuracy is evaluated by how well the chosen action reflects the correct one (occurred event). Our solution needs to be able to predict the next action (in our case a read operation) based on previous events and to weight actions according to their occurrence number. We assume that the more an action occurs, the more it has chances to occur in the near future.

For the sake of Lynx design, we chose to use a simple Markov chain as a probabilistic machine to learn and predict I/O access patterns with minimum CPU and memory overhead. Markov chains have been already adopted for prefetching in a different context such as in [321].

A Markov chain is a sequence S of random variables $S = X_1, X_2,\ldots, X_n$ with the Markov property, namely that the probability of moving to next state depends only on the current state and not on the previous ones[15].

Markov chains are often described as state machines where each state models one random variable of the set S. The directed edges between states are labeled with the probabilities of going from one state to the other. A Markov chain can also be represented by a transition matrix T where lines are labeled with states at time t and columns are labeled with states at time t + 1. The matrix T holds the occurrence number of a given transition between two states.

We used Markov chains in our context to solve the problem of predicting I/O patterns:

- Each file is represented by an independent state-machine.
- Each node/state of a given state-machine represents uniquely a file page.
- An edge/transition between two states represents successively requested (read) file pages.

By using Markov learning machine, transition probabilities between pages are computed continuously in runtime. This enables Lynx to start the prediction phase as soon as the learning starts.

Learning and predicting I/O patterns

We describe our approach on a dataset DS containing N file pages $Fp_1,\ldots, Fp_N$, where all file pages have similar sizes (more precisely one Linux page).

The current implementation of Lynx answers the two questions concerning the prefetching as follows:

- When to prefetch? as soon as there is an outgoing edge from the actual node representing the accessed file page.
- How much to prefetch? this parameter is configurable in Lynx; in the performance evaluation part, it was set to be one file page considering the latency of accessing SSD is small enough.

Figure 66 shows an example of a Markov chain with its transition table, this example is described further in the next sections.

–	Fp_2	Fp_3	Fp_4	...	Fp_8	Fp_9
Fp_2	0	2	1	...	3	0
Fp_3	0	0	2	...	0	0
Fp_4	3	0	0	...	0	0
...				...		
Fp_8	2	0	0	...	0	1
Fp_9	1	0	0	...	0	0

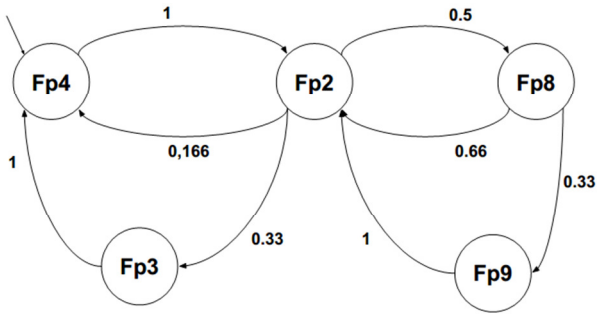


Figure 66. Markov state machine and transition table example

[15] Taking into account more than current and previous I/Os to a file is currently under study

1) Learning phase: The learning phase captures incoming read operations and updates transitions between file pages to build up the state machine. For each transition, from Fp_i to Fp_j, Lynx continuously computes and updates the occurrence number and builds the transition table. The transition table T holds the number of occurrences $O_{i,j}$ for each transition from a file page Fp_i to the next file page Fp_j. The number of occurrences allows to calculate the transition probability by dividing the number of occurrences by the total number of transitions.

2) Prediction phase: During the prediction phase, the transition table T is used to predict the next file page(s) accessed based on the current read operation to the file page Fp_i. The predicted access is given by retrieving, from T, all the possible transitions from the file page Fp_i and choosing the transition which has the highest probability. This probability is computed by dividing the number of occurrences of a given transition by the total number of transitions on Fp_i.

Example: For a given file page Fp_i, we can have one or many possible transitions in case the file page has been accessed in different page file sequences. For instance, if we have the following sequence of read file page numbers (for a given file): {Fp_4, Fp_2, Fp_8, Fp_9, Fp_2, Fp_3, Fp_4, Fp_2 Fp_8, Fp_2, Fp_3, Fp_4, Fp_2, Fp_8, Fp_2, **Fp_4**, (Fp_2), (Fp_8)}. The Fp_4 in bold is the last accessed page and parenthesized file pages are assumed to be the future read operations. Based on the sequence of reads given above, we build the transition table and the corresponding state machine, see Figure 66. For example, file page Fp_2 has three possible transitions with a number of occurrences equal to 2 for $O_{2,3}$, 1 for $O_{2,4}$ and 3 for $O_{2,8}$. As a consequence, probabilities are equal to 0.333, 0.166 and 0.5, respectively. Fp_4 in bold is the last accessed page which represents the current state of the Markov chain. Fp_4 state has only one outgoing edge to Fp_2, then the predicted next element is Fp_2. The Markov chain next state is then Fp_2. From Fp_2, we have three out-going edges; the most accurate is the one going to Fp_8. Thus, Fp_8 is chosen as the predicted next access.

Implementation

We implemented Lynx on the Linux kernel version 3.2. We integrated the following parts to the traditional read-ahead system:

- The **learning mechanism** intercepting all read operations and building the data structures. We focused in the first version of Lynx on files accessed through mapping operations.
- The **prediction mechanism**, we adapted synchronous and asynchronous read-ahead mechanism to take into account all I/O access patterns by modifying the associated kernel routines.
- A **prediction accuracy control** which continuously checks the efficiency of predictions by counting the number of misses on each accessed file and reinitializing the state machine in case an (in)efficiency threshold is reached.
- A **user-space control mechanism** through a system call, that allows to switch on Lynx for some specific files through `madvise()` system call. Lynx can be activated by choosing the MADV_RANDOM hint as argument. This flag is originally chosen to tell the system the I/O pattern is random so that to deactivate read-ahead. We chose to modify the behavior to call Lynx.

In the future, Lynx can be either seen as a replacement for read-ahead or an alternative prefetching mechanism for a given number of files (to reduce the overhead).

3.5.3.3 Performance evaluation of Lynx

We evaluated the performance of Lynx according to 4 metrics: the total execution time, the number of major page faults, the system time and the memory overhead. We evaluated these metrics for both prefetching policies: the traditional policy and Lynx. All experiments were performed on memory mapped files.

We experimented using the TPC-H benchmark which offers a decisional database and 22 ad hoc queries. We created a database using an industrial high performance DBMS and populated it with 10GB of data. To avoid measuring transient regime, we first ran a warm up phase for Lynx. This warm up consists in running the TPC-H benchmark once. After the warm up phase, we flushed the page cache before measuring the performance of the 22 tested queries. In order to isolate the effect of Lynx for each query, we also flushed the page cache before each query experimentation.

Lynx was configured to prefetch one page for future access on each miss/hit. We set the limit of transitions per state to 20. All these values are configurable with Lynx.

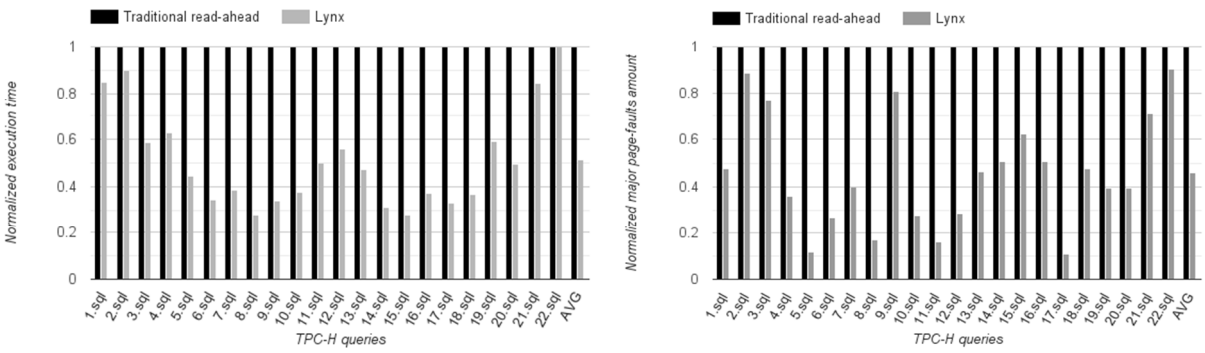


Figure 67. Execution times and major page fault normalized to read-ahead results

Figure 67 shows the execution time for each TPC-H query normalized to the traditional read-ahead mechanism and compared to Lynx. This figure shows that Lynx read-ahead reduced the execution time for all TPC-H queries except the 22.sql query. Indeed, the 22.sql request uses the same table as previous requests with a different pattern, so Lynx applies its prediction based on previous request occurrences which interferes with the actual pattern. This issue is subject to optimization in future work. Lynx reduces the execution time of queries by 50% on average (from 103.7 to 55.3 seconds) and up to 70%. The resulting TPC-H queries execution time enhancement is correlated with the number of major page faults and system time reduction.

Figure 67 presents, on the right hand side, the amount of major page faults for each TPC-H query normalized to the traditional read-ahead mechanism and compared to Lynx. The figure shows that Lynx reduces considerably the number of major page-faults (by 54% on average).

We measured the time spent by the CPU for executing kernel instructions during the execution of the queries. This time reflects the kernel overhead of Lynx. This measured system time proved to be lower than the one with traditional read-ahead (around 50% reduction). This is due to the dynamic nature of data size prefetched in the traditional read-ahead while it is static (but configurable) in Lynx. In addition, by reducing the number of major page-faults, we also reduced the system time.

The total main memory allocated to build the state machine for the TPC-H database files was measured to be equal to 94MB. This represents less than 1% of the TPC-H file data size (10GB), and less than 0.5% of the overall workload (18GB). This memory overhead scales with the number of file pages and the number of transitions between file pages in the state machine.

3.5.3.4 Conclusions and future work

This section described a prefetching mechanism named Lynx, to replace or complement the Linux read-ahead system. The design of Lynx was motivated by two main observations, first I/O workloads for current data intensive applications are less sequential and second, flash memory based devices have symmetric sequential and random performance with low latencies.

Rather than focusing on spatial locality, Lynx focuses on algorithmic locality and prefetchs data regardless of the I/O pattern (sequential or random). Lynx relies on learning I/O patterns and representing them under the form of a Markov chain (state machine). Lynx was implemented and tested on a Linux kernel, and gave very good performance. We experimented with TPC-H database benchmark on an industrial DBMS and our solution has shown 50% enhancement on the prefetching efficiency and on the overall execution time as compared to traditional read-ahead. Lynx is also very light with about 200 lines of code.

While the achieved performance of Lynx is good, there is still optimization room as some limitations that need to be addressed in future work have already been identified. One can cite the following:

- As each file is represented by a unique state machine, when two applications access the same file in an interleaved way, there will be only one pattern prefetched at the expense of the other. A way to address this issue would be to tag the state machine per process identifier or to consider previous file pages to detect the sequence. In the latter case many data mining techniques can be used such as: apriori classification, FP growth, etc.
- Another limitation is the size of metadata. In fact, one needs to store the state machine in which each state represents a file page. This can be very cumbersome for a large data volume. One way to solve this issue is to consider blocks of pages per state rather than pages, or a more adaptive structure for prefetching (according to the I/O pattern).

3.6 Conclusion & perspectives

We have summarized in this chapter three studies we have achieved that are related to database management optimization through storage related techniques. One of those studies was realized on the application level (sorting), one at a system level (Lynx) and one at both application and system level (embedded database I/O modeling).

We have first presented a study on cost modeling for DBMS simple queries on embedded systems. This study was performed on SQLite on embedded systems using flash specific file systems. The objective was to understand and model the performance behavior of SQL queries according to the storage system and more precisely to the file system used. Our experimentations exposed many differences in query management between JFFS2 and UBIFS which gave different models for simple queries. Our study may allow modeling more complex queries and integrating optimization into the DBMS based on the developed models. As embedded systems turned to FTL based storage systems since eMMC standardization, we think it would be relevant to study the case of those systems. Also, we would like to go a bit further and propose some optimization techniques based on the cost models.

The second contribution we have discussed is on data sorting in DBMS for flash based devices. We have designed a novel data sorting algorithms that relied on SSD particular performance model. Besides performing better than state of the art work, our algorithm scales well with large datasets and small memory space. As a perspective, we would like to study how to make the sorting performance robust. Indeed, SSD performance may fluctuate in case of heavy burst of write operations. This is due to GC algorithms. One direction would be to propose some adaptive sorting algorithms that adapt the sorting process according to each SSD performance model and state, to get better performance. We also intend to work on join queries for SSDs.

The third contribution is more a system related one for database application optimization. We have designed a simple, yet effective prefetching mechanism integrated in Linux. It is based on algorithmic locality and relies on a simple Markov chain for prediction. This prefetching mechanism was possible since on flash memories random reads are about as fast as sequential ones. Many directions are open concerning this contribution as already noted in section 3.5.3.4. One needs to explore some other way to represent the Markov chain as storing one node per page proved to have a large memory footprint.

We will carry on our work toward a better storage system for database management. We would also like to develop new research topics concerning Big Data analytics systems. For instance, the SPARK engine for data processing, or large Graph tools such as X-stream, Graphlab or Venus for processing and Neo4j for graph databases. All those Big Data tools have an extensive usage of memory and storage and there is room for optimizations that can be explored.

3.7 Outcome

Advised PhD student	**Projects**	**Collaborations**	**Publications**		
			Journals	**Int conf.**	**Others**
Mr. Chaikh Salmi		ENSMA Poitiers (L. Bellatreche)			
Mr. Arezki Laga	FlashBaD	KoDe Sotware (M. Koskas)	2	5	1
Ms. Assia Ydroudj	PHC Tassili	USTHB, Algérie (K. Boukhalfa)			

4 MID TO LONG TERM RESEARCH PROJECT

4.1 Summary

This chapter introduces to emerging Non-Volatile Memories (NVM) and their integration in computer systems. I will discuss the challenges and issues that seem interesting to tackle in my future work.

4.2 Context

According to IDC, the overall amount of data generated in 2013 is about 4.4 Zettabytes. It will grow by one order of magnitude to reach 44 Zettabytes in 2020, and it is forecasted to attain 180 Zettabytes by 2025. This is mainly due to the growth of the number of devices (and sensors within devices) generating data: around 11 billion devices are connected to internet by now.

Processing data efficiently is a major economic and social issue. According to European commission, "*Big data technology and services are expected to grow worldwide to USD 16.9 billion in 2015 at a compound annual growth rate of 40% – about seven times that of the information and communications technology (ICT) market overall*". In general, better analyzing data means better results, processes and decisions. It can help to generate new ideas or solutions or to predict future events more accurately (disease propagation, weather forecast, etc.). Therefore, data is at the center of the future knowledge economy and society, and new storage techniques need to be developed in order to keep pace with the data explosion.

More data processing leads to more pressure on memory and storage systems. In fact, current applications such as social networking, business intelligence, video and photo sharing, transaction processing, and search engines are becoming more and more data intensive. Those data are processed intensively and in different ways which puts more and more pressure on the memory and storage system as those data need to be extracted and accessed in an efficient way.

4.3 Issues with current memory systems

Memory hierarchy efficiency can be characterized by many metrics such as: (1) performance metrics that can be evaluated through: data *throughput* and *latency*. Data throughput optimization helps to reduce the processor/memory/storage gaps. The *latency* should also be decreased in order to enhance applications' response times. (2) E*nergy efficiency* is also critical as the memory system consumes a large proportion of the overall computer system energy.

Drastically improving current memory systems needs a paradigm shift as it can hardly be carried out on traditional systems, this is because of the reasons stated hereunder.

DRAM technology does hardly scale, is expensive, and not energy efficient: DRAM technology has undergone impressive improvements in terms of size, capacity and performance in the past decade. However, many studies predict that its scaling trends are near to be hit and we are to attain a plateau over the next 5 to 10 years. In fact, DRAM cells need to be large enough for reliable sensing, which is contradictory with the feature size shrinking trend. As a consequence, high density DRAMs impose an exponential cost penalty (e.g. in 2008, using 1 DIMM 8GB costed 212$/GB while 2*4GB DIMMS costed 50$/GB, and 15$/GB for 4*2GB DIMMS [273]). On the other hand, a DRAM cell needs to be periodically refreshed, thus consuming power even when no operation is performed. It was proved to consume between 19% and 31% of peak power [273]. Other problems related to disturbance issues and reliability for very reduced size features seem very hard to handle for future DRAM designs [67][274]. So it is more and more difficult to increase scaling of charge based memory technologies such as DRAM but also flash memories [272].

Storage systems performance is (still) lagging: even if the storage system took a breath of fresh air thanks to flash memory technology, the gap with DRAM speed is still too high. This makes accessing the storage system for data intensive application a real bottleneck. Some applications use in-memory computing to avoid such latencies (milliseconds for hard disk drives and tens to hundreds of microseconds for flash memories as compared to tens of nanoseconds for DRAM). Nevertheless, as stated earlier, data are getting larger while DRAM can hardly scale to larger sizes, thus processing large datasets should rely on efficient storage systems. As a matter of fact, alternative storage system solutions need to be investigated.

Operating system overheads are too high: current implementation of operating system storage stack is based on a millisecond latency storage system, and thus cannot take full profit of high speed storage system such as flash memory. In addition, many of the current storage system structures do not scale well with the ever increasing storage system volumes. For example, as noticed in [275], in disk based storage systems that operate in a millisecond scale, the overhead of the kernel storage stack represents about 0.3% of the latency and 0.4% of the energy, while for flash storage, it represents about 20% of the latency (see Figure 68) and more than 70% of the energy. As it has been observed in [276], in a Linux kernel, the block layer cannot scale beyond one million IOPS (I/O operations per second) regardless of the number of CPUs. This is a real bottleneck to consider for future storage systems.

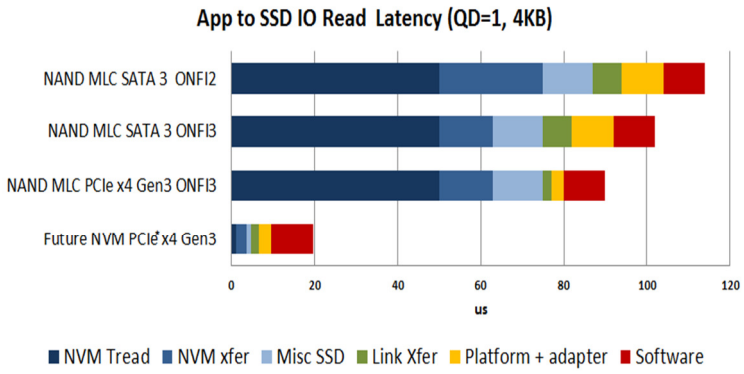


Figure 68. Software latency vs other overheads [279]

4.4 Non-volatile memory, a paradigm shift

In this context, emerging resistive Non-Volatile solid-state Memories (NVMs) promise to revolutionize the memory hierarchy pyramid. Many NVMs have emerged during this last decade, such as Phase Change Memory (PCM), Resistive Memory (ReRAM), and Spin Torque Transfer Memory (STT-RAM). They all present some peculiar and very interesting characteristics making them good candidates to climb on the memory hierarchy pyramid. Intel and Micron 3D XPoint [277] (chased by Samsung [278]) have already begun mass production of NVMs.

NVMs have some peculiar characteristics different from currently used memories such as DRAM and SRAM. Their performance and energy consumption is asymmetric: reads outperform writes. Their endurance may vary from an NVM to another and is more impacted by write operations than reads. Finally, NVMs consume energy only when accessed, which make them very appealing for energy constrained systems (see Table 18).

Table 18. Characteristics of NVMs according to state-of-the-art studies [280][281][282][283][284][285]

	SRAM	DRAM	HDD	NAND flash	STT-RAM	ReRAM	PCM
Cell size (F^2)	120-200	60-100	N/A	4-6	**6-50**	**4-10**	**4-12**
Write endurance	10^{16}	$>10^{15}$	$>10^{15}$	10^4-10^5	**10^{12}-10^{15}**	**10^8-10^{11}**	**10^8-10^9**
Read Latency	~0.2-2ns	~10ns	3-5ms	15-35 µs	**2-35ns**	**~10ns**	**20-60ns**
Write Latency	~0.2-2ns	~10ns	3-5ms	200-500µs	**3-50ns**	**~50ns**	**20-150ns**
Leakage Power	High	Medium	(mechanical)	Low	**Low**	**Low**	**Low**
Dyn. Energy (R/W)	Low	Medium	(mechanical)	Low	**Low/High**	**Low/High**	**Medium/High**
Maturity	Mature	Mature	Mature	Mature	**Test chips**	**Production**	**Test chips**

4.5 NVM integration

Memories are a key component in computer systems, and both their function and integration into systems have evolved considerably over time. In effect, a large set of devices and technologies have been developed.

The design of modern computers introduced the first division in memory hierarchy management: a fast memory that supports the high-speed movement of data, which is indispensable for running applications which is called primary or main memory; and a memory used for mass storage. The latter supplies the former, thanks to a lower cost data storage property, which is called secondary memory or storage memory. This hardware classification of memories has translated into different management policies at the operating system level, as different services manage those memories. Indeed, a specific service manages the primary memory and shares this resource among different application processes, while the storage system is managed as a peripheral, more precisely, as a block device.

The position of a given memory technology in the memory hierarchy (see Figure 69) reflects the distance to the main processing unit. The better its performance in terms of bandwidth and endurance, the closer it is architected with respect to the processing unit; the poorer its performance, the farther it is architected with respect to the processing unit.

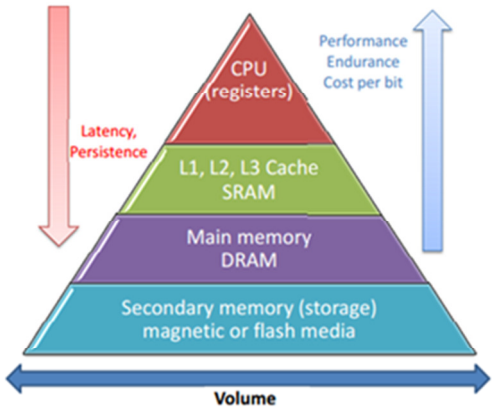


Figure 69. The memory hierarchy pyramid

From an architectural point of view, one might think about inserting a given NVM technology in any location within the memory hierarchy solely according to its characteristics in terms of technology process, bandwidth, and endurance. From a quantitative point of view, the integration would depend on some cost considerations. Generally speaking, the greater the performance of a given memory technology, the higher the cost and the closer to the CPU it is integrated. For instance, an example of integration is under the form of a cache level. SRAM caches are memories that are closer to the CPU.

However, the situation is different from both the applicative and operating system points of view. For instance, from the operating systems perspective, even though many memory components are integrated in a computer, it is mainly the following three that are recognized:

1) The cache system, which can be activated or deactivated, but for which no explicit management is provided, as the CPU cache is managed by a specific hardware component.

2) The main memory, which is byte addressable and is managed as a resource with a dedicated service in the operating system. With regard to the latter, the system relies on some hardware components to accelerate its management (for instance, the Translation Lookaside Buffer or TLB).

Finally, the storage system, be it a traditional magnetic disk or a flash-based storage drive. The storage device is a block-addressed device managed as a peripheral from the operating system perspective.

From the architectural point of view, the integration of NVMs can be performed **vertically** or **horizontally**. Vertical integration means that the integrated NVM will shift a given memory technology down in the memory hierarchy and replace it, while horizontal integration means supplementing a given memory technology in the hierarchy [67]. In horizontal integration, the newly integrated NVM is generally interfaced in the same way as the existing memory, at the

same level. Whether the integration is horizontal or vertical, hardware (controller) and/or software (operating system and/or applications) components must be aware of the new memory stack for performance optimization reasons.

In order to fully integrate NVMs, one must consider hardware integration and software integration. The former answers the question about the horizontal or vertical integration and how NVM should interface with the rest of the system, while the latter is related to the system and to application upgrades to make full profit of the additional memory technology.

4.6 NVM, why and how?

From the architectural and system points of view, a new NVM can be inserted in the memory hierarchy in one of the three main subclasses of memory: processor caches, main memory, or storage systems. NVMs can be vertically or horizontally integrated.

4.6.1 As a storage system

From the storage system point of view, NVM can be integrated horizontally. Therefore, similar to flash memory, some constraint management mechanisms should be implemented and abstracted to higher levels so that the new NVM-based storage system can be integrated in the traditional storage software stack. NVM can also be integrated vertically such as PCIe-based flash devices. Another possibility is to have some hybrid devices such as HDDs integrating flash memories, but this time with NVM technologies other than flash.

4.6.2 As a main memory

As many NVM technologies have very appealing performance characteristics (see Table 18), they can be integrated into a higher level in the memory hierarchy than the storage system. NVMs have some peculiar characteristics. Read performance is higher than the write performance. Their endurance varies according to the technology and is more impacted by write operations than by read operations. Finally, most NVMs use only power when accessed. All of these characteristics must be dealt with and some new services must be included when integrating NVM in place of DRAM or next to DRAM.

Again, NVM can be integrated vertically with DRAM or horizontally [286]. In the first case, DRAM can be regarded as a cache for the NVM in order to reduce the access latency to the main memory. NVM can also be integrated horizontally on the same bus as the memory. Data placement issues could be managed by hardware and thus abstracted to the operating system, or managed by the operating system and abstracted to the application. Finally, some interface could be made available to the application by the operating system to facilitate decisions on placements at the applicative layer. In such cases, operating system support is a critical issue when dealing with memory heterogeneity.

4.6.3 As a processor cache

From the CPU cache perspective, the first level cache is generally accessed at a high frequency, so very low latency and high endurance is required. This is hard to achieve with most NVMs, even if some of them can be considered as candidates for such integration. On the other hand, last level caches are designed more to reduce off-chip data movements and thus must have high density (to achieve large capacity). NVMs integrated in CPU caches are intended more for use in the last level cache than in other levels of cache [67]. However, even in this case, NVMs will absorb very large amounts of write operations, so issues of cell wear out must be mitigated. In addition, the compatibility of the technology with the fabrication process is also a very important issue.

4.7 Research challenges

In future research, I would like to explore the integration of NVM in memory/storage systems and its impact on the overall system and application design. Integration at the main memory level would allow to take full advantage of NVM performance. Indeed, NVMs: (1) allow for a higher density/scalability than DRAM, (2) have performance approaching DRAM, and (3) are more energy efficient than DRAM.

Within this context, many challenging questions arise:

4.7.1 Architectural integration

How to architect such NVMs with DRAM? In order to benefit from the performance of NVM, one should think of integrating it on the memory bus, side by side with DRAM. In such a configuration, many issues need to be investigated: how to manage heterogeneity within the main memory, how to make the MMU (Memory Management Unit) work on both memories, how should the memory be addressed by the processor and its relation with the CPU cache. In addition to this, some specific NVM issues must be dealt with as endurance of NVM is lower than DRAM (see Table 18). This includes wear leveling techniques used to balance write operations over all the NVM cells to increase its lifetime, and write optimization techniques to reduce write operations load.

4.7.2 System Integration

How to integrate NVM at the operating system level? Historically, the main memory has always been volatile while storage systems were non-volatile. The memory (volatile) and storage (non-volatile) stacks were developed based on this assumption and a clear separation was made at the kernel level. With the introduction of NVMs, things should be managed differently: (1) because of the asymmetric properties of read and write operations, (2) because of their non-volatility. While the first property should be mitigated in a hybrid memory by performing, for instance, smart data placement, the second property pushes to rethink the storage/memory frontier from an operating system perspective, for instance by avoiding data duplication between NVM and the storage system. Memory management and file system services must be revisited to benefit from the low latencies of such memories. Indeed, burdening NVMs with the kernel (storage) block layer would be catastrophic for performance, while considering it simply as non-volatile would be wasteful and unrealistic. This is because it would force data duplication and consistency issues thus lowering performance too. If one needs to fully exploit NVM capabilities, revisiting operating system stack is mandatory.

From a system level perspective, mainly two operating system services need to be revisited: memory and storage file system.

The storage system should take into account both the traditional memory/storage system and the byte addressable NVM on the memory bus. Many issues need to be considered when designing such file system:

- **Storage address space management**: as the file storage system spans two different media (NVM and traditional storage system), one needs to handle this heterogeneity and abstract it (at least partially) to the Virtual File System (VFS). Dealing with address space management implies to manage dynamically the growing/shrinking of the storage size. Indeed, in case of high memory load, the system needs to shrink the NVM space dedicated to storage to recycle some space. Conversely, in case of underutilized memory, the system may migrate some data from the storage system for performance and energy optimization.
- Within this context, file **indexing scheme** is a critical issue for many reasons: to optimize file system search operation (read, write, update files), and to manage efficiently the change in granularity from file to memory part.
- A direct consequence of dynamic change of the size of the storage system is the **design of migration policies.** Data can be migrated from traditional HDD/SDD storage to NVM (and vice versa) either explicitly or implicitly. In the first case, the user can ask to place data to NVM for instance, for performance reasons (equivalent to the memory mapping for NVM). Otherwise the kernel can perform data migration according to memory load and application I/O patterns to optimize performance/energy metrics.
- **File system interfaces** with other kernel layers should be revisited. For instance, the page cache mechanism for NVM should be redesigned/removed/replaced to take into consideration files stored in NVM. The connection with block layer should also be reviewed as in case of NVM, requests need to be directed to the memory management unit.

The memory management unit services for hybrid NVM/DRAM memories should also be revisited. In this context, one interesting issue to consider is the semantic switch from file operations to memory load/store for operating system overhead reduction:

- With NVM memories, **the memory mapping facility (*mmap*)** needs to be reconsidered. In traditional systems, the *mmap* system call creates a file mapping in memory in order to access a file data with memory

load/store semantics. When performing a *mmap*, there are two main changes: a performance change as the file is moved to the memory, and a semantic switch as the file is accessed through memory load/store operations. With hybrid memory, the memory mapping could operate from traditional storage to main memory (DRAM or NVM) which is the classical *mmap*, or from NVM to DRAM (performance and semantic change), or from NVM to NVM (by simply changing semantics). The latter case is very interesting as it allows to bypass the operating system overheads. Those novel issues related to memory mapping will be investigated.

- In addition to the *mmap*, which is an explicit way to change semantics of file operations and switch from the I/O stack to the memory management unit, one could think of implicitly **switching the mode of NVM** (transparently) at the kernel level. This can be performed by maintaining a cache for metadata allowing to translate file operations to memory load/store when some I/O patterns are met. The objective would be to reduce the operating system latencies by switching to the memory management unit without explicit request from the user to avoid I/O stack latencies.
- **Dynamic memory space management** inherent to NVM sharing will be considered. The idea is to study mechanisms that would allow to move the slider toward more main memory or more storage according to memory and storage workloads to take into account performance and energy issues.

4.7.3 Application integration

How to expose the NVM to applications? As the NVM provides high performance, and non-volatile storage, one could rethink the file and memory API to expose some characteristics of such a memory. Many challenging questions arise from this point of view. What is the good abstraction level and how to provide information to the user (throughout APIs) about the memory characteristics and usage?

- **Completely abstracted** to support legacy applications for full portability. The developer does not need to take into account or even to know about the heterogeneity of the memory while the system will try to use the NVM efficiently according to application's workload patterns (see system level proposition).
- **Exposed** through more or less simple API, for a more flexibility and manual tuning. One interesting question that may arise is how to expose the memory/storage heterogeneity to the application developer. Simple solutions consist in exposing the fact that files will be stored in NVM or DRAM, while more complex ones could give performance hints for instance by privileging some kind of best effort performance, response time predictability, energy efficiency, or other properties. In this regard, the API could expose different system properties: architectural properties, performance/energy properties, security properties, etc.

Some standardization organisms such as SNIA had already begun working on the question throughout the "**NVM Programming Model (NPM)**" document [287]. This document claims that this kind of new architecture needs some specific API, but still POSIX traditional API should be supported as well. This means the system needs to provide two APIs, one allowing the programmer to be aware of the presence of NVM while the other (standard POSIX) hides the memory heterogeneity. The overall system should provide good performance/energy/security in both cases.

How applications can take profit of this technology: historically, when designing data intensive applications, one has to pay a particular attention to the I/Os generated so that to reduce the execution time. Historically, minimizing the mechanical movements of HDDs was one of the most important issues (see previous section 3.4.2 page 135 on sorting algorithms). With the advent of SSDs, the asymmetry in performance was more related to the operation types. As a matter of fact, many I/O intensive algorithms (such as sorting) were redesigned to minimize the number of write operations at the expense of more read operations. With the emerging NVMs, HDDs and SSDs will cohabit with NVMs that are byte addressable and that have different performance properties than DRAM. Many state-of-the-art established algorithms will need to be redesigned. NVM performance model and non-volatility push toward rethinking most I/O intensive algorithms.

4.8 Conclusion

Flash memories are just the tip of the big iceberg of NVMs. It can be considered as the first solid state memory entering the storage system stack. Since its emergence, flash memory has been a disruptive technology that put into question the memory hierarchy and more particularly the storage software stack.

In this chapter, we have seen that many very promising NVM technologies such as: PCM, MRAM, FeRAM, and ReRAM, are trying to carve out a place in the memory hierarchy. Those NVMs are in different stages of development, while some are already manufactured (such as FeRAM in some MCUs), others are still in a prototyping phase.

The integration of NVMs has been proposed in mainly two ways, horizontally or vertically. Horizontal integration consisted in complementing existing memory level in the memory hierarchy pyramid and so providing at least two options for storing data (with generally no redundancy). Data placement in this case obeys to some pattern related properties mainly based on two characteristics: operation types, and access patterns. The other type is vertical integration; in this case, an additional level is inserted into the memory hierarchy. The inserted level acts as a cache or a buffer to cope with the performance gap between two memory technologies.

From the applicative or even the operating system points of view, there are mainly two options for the integration. First, it is possible to hide the introduced memory to the operating system and/or the application just like different cache memory levels. The operating systems knows about the existence of a cache, but does not know how much levels nor does it have the possibility to manage it as this is done in hardware. The second option consists in making the newly integrated memory visible to the operating system and eventually to the applications so that, for instance choosing the memory where to store data. In my future work I would like to investigate many of the issues introduced in this chapter, mainly from applicative and system perspective.

Appendix 1 – brief CV in English

1 PERSONAL INFORMATION

1.1 General Information

Jalil BOUKHOBZA, 38 years old,

Nationality: Algerian.

Citizenship: French.

Professional address: Université de Bretagne Occidentale (UBO), Laboratoire Lab-STICC, Département d'Informatique
20 Avenue Victor Le Gorgeu, BP808 Brest 29285, FRANCE

☎ +33 2 98 01 69 73 (Office), +33 6 60 20 28 32 (Mobile)
+33 2 98 01 80 11 (fax)
@ jalil.boukhobza@univ-brest.fr

http://syst.univ-brest.fr/boukhobza

1.2 Current position

Associate professor in Computer science since the 1st of September 2006, Univ. Bretagne Occidentale, Brest, France.

Researcher at the IRT (Institute of Technological Research) B<>COM, 20% of working time, since December 2013, Rennes, France.

- Since 2010: **Head** of the 2nd year Master degree in Software for Embedded Systems.
- Since 2011: **Liaison officer** for foreign student registration at the department of computer science.
- Since 2012: **Member** of the computer science departmental board
- Since 2012: **Member** of the scientific curriculum departmental board
- Since 2013: **Representative** of the computer science department at the faculty's international committee.
- Since 2014: **Deputy Chairperson** of the scientific curriculum at the computer science department.

1.3 Education

Oct. 2000 – Dec. 2004: PhD thesis in computer science at the University of Versailles PRiSM Laboratory, France. Distinction: very honorable
PhD thesis subject: « Performance analysis and simulation of file I/Os on PC systems »
Advisor: Pr Claude Timsit

Oct.1999 – Sept.2000: Master of Science in Computer Science (in parallel systems) with honors at the University of Versailles, France.
Master thesis subject: « Storage system configuration management: SAN vs NAS, a comparative study »
Advisor: Pr. Claude Timsit

Sept.1994 – Oct.1999: Engineering degree in electronics with honors, National Institute of Electronics (I.N.E.L.E.C), Algeria.
Thesis subject: « Parallel Processing In Graphics »
Advisor: Dr. Abbes Bouklachi.

1.4 Academic positions and affiliations

Since December 2013: Researcher at the IRT (Institute of Technological Research) b<>com, 20% working time.

Since September 2006 : Associate professor in computer science at the Univ. Bretagne Occidentale, Computer Science department, Faculty of Science and Techniques, Lab-STICC CNRS UMR 6285.

Sept. 2004 – Aug. 2006: Research and teaching assistant at the University of Versailles, Computer Science department, PRiSM Laboratory, France.

Sept. 2000 – Dec. 2004: PhD student and teaching assistant at the University of Versailles Computer Science department, PRiSM Laboratory, France.

1.5 Awards

- **Awarded** by the national council of Universities (CNU) for scientific achievement since September 2013.
- **Best paper award** : Jalil Boukhobza, Pierre Olivier, Stéphane Rubini, "A Cache Management Strategy to Replace Wear Leveling Techniques for Embedded Flash Memory", in proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (**SPECTS**), pp. 1-8, The Hague, 2011
- **Prestige Grant Award** for mobility in 2016

2 RESEARCH

2.1 Research interest

- **Main topics of interest**: Storage systems, memory hierarchy, operating systems, computer architecture
- **Application domains**: Embedded and mobile systems, real-time systems, Cloud, IoT
- **Objectives**: Energy consumption and performance optimization.

2.2 Research Projects

This section provides a list of national, international, and industrial collaborative research projects in which I participated.

Table 3. Summary of the projects

Projets	Années de participation :	Type de projet	Rôle	Partenaires
FALCON	2017-2020	IRT B<>COM	Member	IRT b<>com
GHEEMaS	2016-2019	PHC Tassili (International project)	Project manager and co-initiator	USTHB Algérie, CERIST Algérie
BaD FLASH	2015-2018	Research contract	Project manager and co-initiator	KoDe Software
Orange	2015-2018	Research contract	Project manager and co-initiator	Orange Labs
INDEED	2013-2016	IRT B<>COM	Member	IRT b<>com
Open PEOPLE	2011-2012	ANR	Member	INRIA Lilles, LORIA, Univ. Bretagne Sud, Univ. Rennes 1, Univ. Nice SA, Thalès Comm., InPixal
Think FPGA	2010-2011	Exploratory project (university)	Project co-manager	Laboratoire LISyC (maintenant intégré au Lab-STICC)
XanKom	2009-2010	Research contract	Project manager	PME XanKom
MORPHEUS	2007-2008	European FP6	Member	Thales France, Ace associated compiler experts b.v. Netherlands, Alcatel-Lucent Germany, Universita di Bologna Italy, Arttic France, CEA France, Critical blue UK, Thomson Germany, Intracom s.a. telecom solutions Greece, M 2000 France, Pact xpp technologies Germany, STMicroelectronics Italy, Technische Universitaet Braunschweig Germany, Technische Universitaet chemnitz Germany, Technische Universiteit Delft Netherlands, Universitaet Karlsruhe Germany[16]

[16] http://cordis.europa.eu/project/rcn/80670_en.html

- **Falcon - IRT b<>com** (2017-2020): Resource isolation in a container-based virtualization in the Cloud.
- **PHC Tassili GHEEMaS** (2016-2019): A Green and Highly Efficient and Evolutionary data and storage Management System for Cloud Data Base as a Service
- **Project with Orange** (2015-2018): Cloud Storage for the IoT, a Content-centric Approach for Consistency and Performance Optimization
- **FLASHBaD - KoDe Software** (2015-2018) : Optimizing high performance DBMS for flash storage systems
- **INDEED - IRT b<>com** (2013-2016): Toward a Green Distributed Cloud Infrastructure
- **ANR Open PEOPLE** (2011-2012) : Open-Power and Energy Optimization PLatform and Estimator
- **Think FPGA** (2010-2011): A Component Based Approach (THINK) for RTOS optimization on reconfigurable FPGA architectures
- **XanKom Project** (2009-2010): Porting Embedded Linux on router board
- **FP6 MORPHEUS** (2007-2008): Multipurpose Dynamically Reconfigurable Platform for Intensive and Flexible Heterogeneous Processing: A technology breakthrough for Embedded Computing

2.3 Research student supervision

2.3.1 PhD students

1. Pierre Olivier (2011-2014) **thesis defended**
   Supervised with: Dr. Eric Senn (Associate professor, UBS Lorient, France)
   Subject: Performance and energy consumption estimation of embedded flash storage systems
2. Yahia Benmoussa (2011-2015) **thesis defended**
   Supervised with: Dr. Eric Senn (Associate professor, UBS Lorient, France), Pr. Djamel Benazzouz (University of Boumerdes, Algeria).
   Subject: Performance and energy consumption characterization and modeling of video decoding on multi-core heterogeneous SoC and their applications
3. Hai Nam Tran (2013-2017) **thesis defended**
   Supervised with: Pr. Frank Singhoff (UBO), Dr. Stéphane Rubini (UBO).
   Subject: Real-time embedded system analysis with cache
4. Cheikh Salmi (2011 -2017) **thesis defended**
   Supervised with: Pr. Ladjel Bellatrèche (ENSMA Poitiers, France).
   Subject: Optimizing Data Warehouse storage systems
5. Djillali Boukhelef (2012 -)
   Supervised with: Dr. Kamel Boukhalfa (Associate professor, University of Algiers USTHB, Algeria).
   Subject: Data placement for hybrid storage and data warehouses optimization for the Cloud
6. Hamza Ouarnoughi (2013-)
   Supervised with: Pr. Frank Singhoff (UBO), Dr. Stéphane Rubini (UBO).
   Subject: Distributed, heterogeneous and adaptive storage system in IaaS Cloud context.
7. Arezki Laga (2014-)
   Supervised with: Pr. Frank Singhoff (UBO).
   Subject: Optimizing I/O performance of high performance DBMS
8. Mohammed Islam Naas (2015-)
   Supervised with: Pr. Alain Plantec (UBO).
   Subject: Cloud storage for Internet of Things, a content-centric approach for data consistency and performance optimization.
9. Assia Ydroudj
   Supervised with: Dr. Kamel Boukhalfa (USTHB, Algieria).
   Subject: Non-Volatile Memory for optimizing Big Data base/warehouses in Cloud context.
10. Jean-Emile Dartois
    Supervised with : Pr. Olivier Barais (IRISA, Univ. Rennes 1).

Subject: Heterogeneous hardware resource management and Cloud service warranty, the case of the containers.

2.3.2 Master students

1. Khalifa Rouis, June 2010, « Performance analysis of flash storage systems ».
2. Bi Shen Yi, September 2010, « Porting embedded Linux on Router Board 411 ».
3. Ilyes Khetib, September 2011, « Simulation of flash storage systems with cache ».
4. Pierre Olivier, September 2011, «Design of a hardware time manager for a component based RTOS» .
5. Hamza Ouarnoughi, September 2013, « Performance cost models for embedded databases ».
6. Chakib Belgaïd, June 2016, « Energy consumption and modeling of SSDs ».
7. Mohammed Bey Ahmed Khernache March 2017, « Optimizing sorting algorithms on NVM ».
8. Oussama Elhamer March 2017, « Qualifying SSD Endurance and Performance ».

2.4 Academic research collaborations

2.4.1 Initiated academic collaborations

- **Polytechnic University of Hong Kong (PolyU), Hong Kong, Chine**: initiated on August 2015 with Pr. Zili Shao on integrating NVMs in operating system stack.
- **Institute of Computer Science (Ecole Supérieure d'Informatique, ESI Algiers, Algeria)**: initiated in 2015 with Pr. Mouloud Koudil, on research and teaching issues related to embedded systems.
- **University of Science and Technology of Houari Boumediene (USTHB), Algiers, Algeria**: initiated in 2014 with Dr. Kamel Boukhalfa on database storage optimizations.
- **University of Mhamed Bougara, Boumerdès (UMBB), Boumerdès, Algeria**: initiated in 2011 with Pr. Djamel Benazouz on energy consumption and video applications for mobile computing
- **University Abou Bakr Belkaïd, Tlemcen, Algeria**: initiated in 2012 with Pr. Abdelkrim Benamar and concerned some co-diploma collaboration with our University.
- **University of Mérida, Mérida, Venezuela:** initiated on 2011 with Pr. Armando Borrero on I/O workload modeling.

2.5 Professional activities

2.5.1 Participation to PhD thesis review

- Reviewer of **M. Nezer Jacob Zaidenberg** PhD thesis, on "Applications of Virtualization in Systems Design", **University of Jyväskylä**, **Finland**. Defended on June, 12th 2012.

2.5.2 General chair

- **Co-chair and initiator: EWiLi** (2011, 2012, 2013, 2014, 2015, 2016, 2017), The Embedded operating systems workshop
- **Co-chair and co-initiator of the WOPSSS** (2016, 2017) **workshop on performance and scalability of storage systems (WOPSSS)** in conjunction with the ISC conference.
- **Co-chair and initiator of the French workshop PerSSS** (2015, 2016, 2017), Performance and Scalability of Storage Systems, co-organized with DDN, CEA, and UVSQ (http://syst.univ-brest.fr/persss/in French).
- **Steering committee member of the ECOFAC doctoral school** (in 2015), Doctoral school on low power design for embedded and real-time systems.

2.5.3 Technical program committee chair

- **Program co-chair: EWiLi** (2011, 2012, 2013, 2014, 2015, 2016), The Embedded operating systems workshop
- **Program co-chair: WOPSSS** (2016, 2017), workshop on performance and scalability of storage systems.

2.5.4 Track chair

- **Track program co-chair in IEEE/IFIP EUC** (2014), The International Conference on Embedded and Ubiquitous Computing.
- **Track program co-chair in ECSI DASIP** (2012), Conference on Design & Architectures for Signal & Image Processing

2.5.5 Publicity chair

- **Publicity co-chair IEEE NVMSA** (2015, 2016), Non-Volatile Memory Systems and Applications symposium.

2.5.6 Technical program committee member

- **EWiLi** (2011, 2012, 2013, 2014, 2015, 2016, 2017), The Embedded operating systems workshop
- **SPECTS** (2012, 2013, 2014, 2015, 2016, 2017), The International Symposium on Performance Evaluation of Computer and Telecommunication Systems
- **IEEE MC-SoC**, (2014, 2015, 2016, 2017) International Symposium on Embedded Multicore/Many-core System-on-Chip.
- **IEEE NVMSA** (2015, 2016, 2017), Non-Volatile Memory Systems and Applications symposium.
- **ASP-DAC** (2017, 2018), Asia and South Pacific Design Automation Conference
- **WOPSSS** (2016, 2017), workshop on performance and scalability of storage systems.
- **CITS** (2015, 2016), The International Conference on Computer, Information and Telecommunication Systems
- **IEEE/IFIP EUC** (2013, 2014), The International Conference on Embedded and Ubiquitous Computing.
- **ECSI DASIP** (2011, 2012), Conference on Design & Architectures for Signal & Image Processing
- **LCTES** (2017), ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems
- **ICESS** (2017), IEEE International Conference on Embedded Software and Systems.
- **INFLOW** (2016), Workshop on INteractions of NVM/FLash with Operating-systems and Workloads.
- **COMPAS** (2016), Conférence d'informatique en Parallélisme, Architecture, Système.
- **SETIT** (2016), Sciences of Electronics, Technologies of Information and Telecommunications.
- **ISPS** (2015), International Symposium on Programming and Systems
- **CADS** (2015), The CSI International Symposium on Computer Architecture & Digital Systems.
- **ICWIT** (2013), The International Conference on Web and Information Technologies.
- **CIIA** (2013), The International Conference on Computer Science and its Applications.
- **SDIWC DICTAP** (2011), The International Conference on Digital Information and Communication Technology and its Applications.

2.5.7 External reviewer

- **DAC** (2017), Design Automation Conference.
- **ICDT** (2017), International Conference on Database Theory
- **IEEE/ACM MASCOTS** (2006, 2013), International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.

2.5.8 Journal guest editor

- **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 12, Number 1, December 2015 Special Issue the 5th Workshop on Embed With Linux (EWiLi 2015) ACM, pp. 65, Jan.. 2016, 1551-3688
- **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 11, Number 4, December 2014 Special Issue the 4th Workshop on Embed With Linux (EWiLi 2014) ACM, pp. 72, Dec. 2014, 1551-3688
- **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 11, Number 1, February 2014 Special Issue the 3rd Workshop on Embed With Linux (EWiLi 2013) ACM, pp. 79, Feb. 2014, 1551-3688.

- **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 9, Number 2, June 2012 Special Issue the 2nd Workshop on Embed With Linux (EWiLi 2012) ACM, pp. 79, Feb. 2014, 1551-3688.

2.5.9 Journal review

- IEEE Transactions On Computers (2014, 2015, 2016)
- IEEE Transaction On VLSI Systems (2014, 2015, 2016)
- Elsevier Journal of System Architecture (2014, 2015, 2016)
- ACM Transactions on Embedded Computing System (2008, 2016, 2017)
- ACM Transactions On Storage (2015, 2016)
- Elsevier Future Generation Computer Systems (2015, 2016)
- IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2016, 2017)
- Wiley Concurrency and Computation: Practice and Experience (2017)
- IET Electronics Letters (2016)
- Wiley Journal of Software: Practice and Experience (2015)
- IEEE Embedded Letters (2014)
- Springer Distributed and Parallel Database (2014)
- Springer Journal of Zhejiang University - Science C (2014)
- MDPI Computers (2014)
- Springer Journal of Supercomputing (2013)
- Springer Journal of Real-Time Image Processing (2012)

2.5.10 Book chapter review

- **Computer Organization and Architecture**, *10th Edition*, 2015, **William Stallings**, Pearson, 2 chapters.
- **Organization, Design, and Architecture**, *5th Edition,* 2013, **Sajjan G. Shiva,** Taylor and Francis**,** 2 chapters.
- **Embedded Computing Systems: Applications, Optimization, and Advanced Design, 2013, IGI,** 1 chapter.

2.5.11 Scientific expertise

- **Agence Nationale de la Recherche** ANR (National Research Agency), Expert for the program «Digital engineering and security» - INS 2013.
- Member of ad-hoc Committee for the assignment of academic positions at the University 2013.

3 TEACHING ACTIVITY

Table 4. Summary of taught courses. The courses followed by a star are those for which I developed most or all of the course materials.

Students' level	Year	Course title	Teaching Department	Nb hours per year/ frequency (nb years)/years
Undergraduate	1st year	Computer and Internet	Faculty of law (UBO)	~20h/2/2007-2009
			Faculty of science and techniques (UBO)	~20h/2/2007-2009
		Computer applications (network and internet)	Faculty of science and techniques (UBO)	~10h/2/2007-2009
	2nd year	**Architecture and Systems 1*** (Microprocessor, assembly language, micro controller, operating systems)	Computer science department (UBO)	~60h/10/2006-*

		Arithmetic operators		~25h/1/2006
		Programming languages		~30h/2/2006-2008
	3rd year	**Architecture and Systems 2*** (Operating systems, Microprocessor architecture and VHDL language)	Computer science department (UBO)	~30h/10/2006-* (System part) ~20h/2/2006-2008 (Architecture part)
		Algorithms and C language		~20h/1/2007
		Microprocessors*		~20/7/2008-*
		Digital circuits	Computer science department (UVSQ)	~40h/5/2000-2005
Graduate	2nd year	**Embedded Operating Systems***	Computer science department – 2nd year Master	~50h/8/2008-*
		Embedded Systems *		3h/3/2013-*
Engineering schools	2nd year	Introduction to Operating Systems and C language	ENSTA Bretagne	~30h/2/2008-2010
			ISTY-Versailles	~60h/5/2000-2005
	3rd year	Operating System and Embedded Linux *	ENSTA Bretagne	~15h/3/2012-2015
	2nd year	Embedded Operating Systems*	Dept. Embedded Systems, ENSEIRB, Bordeaux	~15h/5/2011-*
International Universities	1st year Master	Real-time Systems* (**in English**)	Univ. Of Science and Technology of Hanoï, Vietnam	~25h/1/2015
	2nd year Master	Operating Systems*	Univ. of Tlemcen, Algeria	~20h/1/2012

4 PUBLICATIONS

4.1 Book (1)

[l1] **Jalil Boukhobza**, Pierre Olivier, **Flash Memory Integration, Performance and Energy Considerations**, 266 pages, ISTE- Elsevier edition, March 2017.

4.2 Journal guest editions (4)

[e1] **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 13, Number 1, January 2016 Special Issue of the 5th Workshop on Embedded operating systems (EWiLi 2015) ACM, Dec. 2015, 1551-3688

[e2] **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 11, Number 4, December 2014 Special Issue the 4th Workshop on Embedded operating systems (EWiLi 2014) ACM, pp. 72, Dec. 2014, 1551-3688

[e3] **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 11, Number 1, February 2014 Special Issue the 3rd Workshop on Embedded operating systems (EWiLi 2013) ACM, pp. 79, Feb. 2014, 1551-3688.

[e4] **Jalil Boukhobza**, Jean-Philippe Diguet, Frank Singhoff, Pierre Ficheux, SIGBED Review, Volume 9, Number 2, June 2012 Special Issue the 2nd Workshop on Embedded operating systems (EWiLi 2012) ACM, pp. 47, May. 2012, 1551-3688.

4.3 Invited conferences (6)

[inv1] **Jalil Boukhobza**, Flash and the NVM team to save the data storage world!, **Les Journées Scientifiques de l'Université de Nantes 2016, Colloque 8 - Stockage informatique : usages, évolutions techniques, impact énergétique, sécurité et droit à la vie privée,** Nantes, Juin 2016.

[inv2] **Jalil Boukhobza,** A Methodology for Estimating Performance and Power Consumption of Embedded Flash File Systems, **Journée Modélisation de la consommation d'énergie,** organisée par le laboratoire CRISTAL, Univ. de Lilles, Juin 2016.

[inv3] **Jalil Boukhobza,** "Flash and the NVM"**, Tech Talk IRT b<>com,** Rennes, Juin 2016.

[inv4] **Jalil Boukhobza**, "Flash storage systems: performance and power consumption issues", école thématique COnception FAible Consommation pour les systèmes embarqués temps réels **ECOFAC**, Lorient, France, May 2014.

[inv5] **Jalil Boukhobza**, Stéphane Rubini, "Flashing the memory hierarchy: an overview on flash memory internals", Journée Logiciels Embarqués et Architectures Matérielles du **GDR SoC-SiP,** Paris, France, Nov 2012.

[inv6] **Jalil Boukhobza**, "Hiérarchie mémoire avec un focus sur les mémoires non volatiles de type Flash", école thématique Architectures des systèmes matériels enfouis et méthodes de conception associées **ARCHI**, Perpignan, France, Jun. 2011.

4.4 International journals (15)

[ri1] Arezki Laga, **Jalil Boukhobza**, Frank Singhoff, Michel Koskas, "MONTRES : Merge ON-The-Run External Sorting Algorithm For Large Data Volumes On SSD Based Storage Systems", accepted in **IEEE Transactions on Computers**, 2017.

[ri2] Hamza Ouarnoughi, **Jalil Boukhobza**, Frank Singhoff, Stéphane Rubini, "Integrating I/Os in Cloudsim for Performance and Energy Estimation", **ACM SIGOPS Operating Systems Review**, Volume 50 Issue 1, pp. 27-36, Dec. 2016

[ri3] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, Hamza Ouarnoughi, "A Methodology for Estimating Performance and Power Consumption of Embedded Flash File Systems in **ACM Transactions on Embedded Computing Systems**, Volume 15, issue 5, August 2016.

[ri4] Hai Nam Tran, Frank Singhoff, Stéphane Rubini, **Jalil Boukhobza**., "Cache-aware real-time scheduling simulator: implementation and return of experience", **ACM SIGBED Review**, Volume 13, Issue 1, special issue on Embedded operating systems workshop (EWiLi'15), pp. 22-28, Feb. 2016.

[ri5] **Jalil Boukhobza**, Pierre Olivier, Stéphane Rubini, Laurent Lemarchand, Yassine Hadjadj-Aoul, Arezki Laga, "MaCACH: An adaptive cache-aware hybrid FTL mapping scheme using feedback control for efficient page-mapped space management", **Journal of Systems Architecture**, Elsevier, Volume 61, Issues 3-4, pp.157-171, Mar. 2015.

[ri6] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Yassine Hadjadj-Aoul, Djamel Benazouz, "A methodology for performance/energy consumption characterization and modeling of video decoding on heterogeneous SoC and its applications", **Journal of Systems Architecture**, Elsevier, Volume 61, Issue 1, pp.49-70, Jan. 2015.

[ri7] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Djamel Benazzouz, "On the energy efficiency of parallel multi-core vs hardware accelerated HD video decoding", **ACM SIGBED Review**, Volume 11, Issue 4, pp. 25-30, special issue on Embedded operating systems workshop (EWiLi'14), Dec. 2014.

[ri8] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "Revisiting read-ahead efficiency for raw NAND flash storage in embedded Linux", **ACM SIGBED Review**, Volume 11, Issue 4, pp. 43-48, special issue on Embedded operating systems workshop (EWiLi'14), Dec. 2014.

[ri9] **Jalil Boukhobza**, Pierre Olivier and Stéphane Rubini, "A Scalable and highly configurable cache-aware hybrid flash translation layer", **MDPI Computers**, Volume 3, Issue 1, pp. 36-57, Mar. 2014.

[ri10] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Yassine Hadjadja-Aoul, Djamel Benazzouz, "DyPS: dynamic processor switching for energy-aware video decoding on multi-core SoCs", **ACM SIGBED Review,** Volume 11, Issue 1, pp. 56-61, special issue on Embedded operating systems workshop (EWiLi'13), Feb. 2014.

[ri11] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "Flashmon V2: Monitoring raw NAND Flash memory I/O requests on embedded Linux", **ACM SIGBED Review,** Volume 11, Issue 1, pp. 38-43, special issue on Embedded operating systems workshop (EWiLi'13), Feb. 2014.

[ri12] Hamza Ouarnoughi, **Jalil Boukhobza**, Pierre Olivier, Loïc Plassart, Ladjel Bellatrèche, "Performance analysis and modeling of SQLite embedded databases on flash file systems", **Design Automation for Embedded Systems**, Springer, Volume 17, Issue 3, pp 507-542, Sept. 2013.

[ri13] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "On benchmarking embedded Linux Flash File Systems", **ACM SIGBED Review**, Volume 9, Issue 2, pp. 43-47, special issue on Embedded operating systems workshop (EWiLi'12), Jun. 2012.

[ri14] Pierre Olivier, **Jalil Boukhobza**, "A hardware time manager implementation for the Xenomai real-time kernel of embedded Linux", **ACM SIGBED Review**, Volume 9, Issue 2, pp. 38-42, , special issue on Embedded operating systems workshop (EWiLi'12), Jun. 2012.

[ri15] **Jalil Boukhobza**, Pierre Olivier, "An efficient hierarchical dual cache system for nand flash memories", **International Journal of Digital Information and Wireless Communications** (IJDIWC), The Society of Digital Information and Wireless Communications, Volume 1, Issue 1, pp. 175-194, 2011

4.5 National journals (2)

[rn1] Pierre Olivier, **Jalil Boukhobza**, "Un système de cache hiérarchique pour les E/S présentant des motifs d'accès séquentiels pour les mémoires Flash NAND", **Techniques et Sciences Informatique,** Volume 32, Issue 2, pp. 203-228, 2013.

[rn2] **Jalil Boukhobza**, Claude Timsit, " Analyse des performances des accès séquentiels aux fichiers sous Windows", **Techniques et Sciences Informatique**, Volume 24, Issue 2-3, pp. 333 -358, 2005.

4.6 International conferences[17](28)

[ci1] Islam Naas, Philippe Raipin, **Jalil Boukhobza**, Laurent Lemarchand, "iFogStor: an IoT Data Placement Strategy for Fog Infrastructure", the IEEE International Conference on Fog and Edge Computing (**ICFEC**), Short paper, May 2017.

[ci2] Djillali Boukhelef, Kamel Boukhalfa, **Jalil Boukhobza**, Hamza Ouarnoughi, Laurent Lemarchand, "COPS: Cost Based Object Placement Strategies on Hybrid Storage System for DBaaS Cloud", The 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**CCGRID**), Short paper, May 2017.

[ci3] Yahia Benmoussa, Eric Senn, Nicolas Derouineau, Nicolas Tizon, **Jalil Boukhobza**, "Green Metadata Based Adaptive DVFS for Energy Efficient Video Decoding", Power And Timing Modeling, Optimization and Simulation (**PATMOS**) Bremen, 2016. (T.A.: 41%)

[ci4] Djillali Boukhelef, **Jalil Boukhobza**, Kamel Boukhalfa, "A Cost Model for DBaaS Storage", 27th International Conference on Database and Expert Systems Applications (**DEXA**), pp. 223-240, Porto, Sep. 2016. (T.A.: 28%)

[ci5] Arezki Laga, **Jalil Boukhobza**, Michel Koskas, Frank Singhoff, "Lynx: A Learning Linux Prefetching Mechanism For SSD Performance Model", The 5th Non-Volatile Memory Systems and Applications Symposium (**NVMSA**), Daegu, Aug. 2016.

[ci6] Hamza Ouarnoughi, **Jalil Boukhobza**, Frank Singhoff, Stéphane Rubini, "A cost model for virtual machine storage in Cloud IaaS", in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**PDP**), pp. 664-671, Heraklion, Feb. 2016. (T.A.: 37%)

[ci7] Hai-Nam Tran, Frank Singhoff, Stéphane Rubini, **Jalil Boukhobza**, "Addressing cache related preemption delay in fixed priority assignment", in proceedings of the 20th IEEE International Conference on Emerging Technologies and Factory Automation (**ETFA**), pp. 1-8, Luxembourg, Sept. 2015. (T.A.: 61%)

[ci8] Pierre, Olivier, **Jalil Boukhobza**, Mathieu Soula, Michelle Le Grand, Ismet Chaib Draa and Eric Senn, "A tracing toolset for embedded Linux flash file systems", in proceedings of the 8th EAI International Conference on Performance Evaluation Methodologies and Tools (**ValueTools)**, pp. 153-158, Bratislava, Dec. 2014.

[ci9] Cheikh Salmi, Abdelhakim Nacef, Ladjel Bellatreche, **Jalil Boukhobza**, "What can emerging hardware do for DBMS buffer?", in proceedings of the ACM 17th International Workshop on Data Warehousing and OLAP (**DOLAP**), Short paper, pp. 91-94 , Shanghai, Nov. 2014.

[ci10] Yahia Benmoussa, Eric Senn, **Jalil Boukhobza**, Djamel Benazzouz and Mickael Lanoe. "Open-PEOPLE, A collaborative platform for remote & accurate measurement and evaluation of embedded systems power consumption", in proceedings of the IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (**MASCOTS**), demo paper, pp. 498-501 Paris, Sept. 2014.

[ci11] Hai Nam Tran, Frank Singhoff, Stéephane Rubini, **Jalil Boukhobza**, "Instruction cache in hard real-time systems: modeling and integration in scheduling analysis tools with AADL", in proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing (**EUC**), pp. 104-111, Milano, Aug. 2014. (T.A.: 39.2%).

[ci12] **Jalil Boukhobza**, Pierre Olivier, Loic Plassart, Hamza Ouarnoughi, Ladjel Bellatreche, "Embedded databases on flash memories: performance and lifetime issues, the case of SQLite", in Proceedings of Embedded Real-time Software and Systems (**ERTSS),** Toulouse, Feb. 2014. (T.A.: 69%)

[ci13] Ladjel Bellatreche , Cheikh Salmi, Sebastian Bress , Amira Kerkad , Ahcene Boukorca , **Jalil Boukhobza**, "How to exploit the device diversity and database interaction to propose a generic cost model", in proceedings of the 17th International Database Engineering & Applications Symposium (**IDEAS**), pp. 142-147, Short paper, Barcelona, Oct. 2013

[ci14] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Djamel Benazzouz, "GPP Vs DSP: A performance/energy characterization and evaluation of video decoding", in proceedings of the 21st IEEE International Symposium on Modeling and Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 273-282, San Francisco, Aug. 2013. (T.A.: 27%)

[17] Acceptance rate is given when available (T. A).

[ci15] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Djamel Benazzouz, "Energy consumption modeling of H264/AVC video decoding for GPP and DSP", in proceedings of the EUROMICRO Digital System Design (**DSD**) conference, Santander, pp. 890-897, Sept. 2013. (T.A.: 44%)

[ci16] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "modeling driver level NAND Flash memory I/O performance and power consumption for embedded Linux", in proceedings of the IEEE 11th International Symposium on Programming and Systems (**ISPS**), pp. 143-152, Algiers, Apr. 2013. (T.A.: 25%)

[ci17] **Jalil Boukhobza**, Pierre Olivier, Stéphane Rubini, "CACH-FTL: a cache aware configurable hybrid flash translation layer", in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**PDP**), pp. 94-101, Belfast, Feb. 2013. (T.A.: 39%)

[ci18] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "Micro-benchmarking Flash memory file-system wear leveling and garbage collection: a focus on initial state impact", in proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (**EUC**), pp. 437-444, Paphos, Dec. 2012 (T.A.: 31%).

[ci19] Stéphane Rubini, Jean-Philippe Babau, **Jalil Boukhobza**, "A hardware/software CBSE framework for RTOS Services: the timing service case study", In proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (**EUC**), pp. 570-577, Paphos, Dec. 2012 (T.A: 31%).

[ci20] **Jalil Boukhobza**, Ilyes Khetib, Pierre Olivier, "Characterization of OLTP I/O workloads for dimensioning embedded write cache for flash memories: a case study", in proceedings of the International conference on Model and Data Engineering **MEDI**, Springer LNCS 6918, pp. 97-109, Obidos, Sept. 2011 (T.A: 27%).

[ci21] **Jalil Boukhobza**, Pierre Olivier, "C-lash: a cache system for optimizing NAND flash memory performance and lifetime", in proceedings of the International Conference on Digital Information and Communication Technology and its Applications (**DICTAP**), LNCS Springer CCIS 167, pp. 519-613, Dijon, Jun. 2011 (T.A: 39%).

[ci22] **Jalil Boukhobza**, Pierre Olivier, Stéphane Rubini, "A cache management strategy to replace wear leveling techniques for embedded flash memory", in proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (**SPECTS**), pp. 1-8, IEEE, SCS, The Hague, Jun. 2011 (T.A: 44%). Ce papier a reçu le prix du **BEST PAPER AWARD**.

[ci23] **Jalil Boukhobza**, Khalifa Rouis, " A sequential workload performance study of embedded NAND flash memories ", in proceedings of the International Conference on Electrical Engineering (**ICEE)**, pp. 85-90, Boumerdes, Dec. 2009. (T.A: 64%).

[ci24] Bernard Pottier, **Jalil Boukhobza**, Thierry Goubier, "An integrated platform for heterogeneous reconfigurable computing", in proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (**ERSA)**, Invited paper , pp. 25-36, Las Vegas, Jun. 2007.

[ci25] **Jalil Boukhobza**, Claude Timsit, "Toolbox for dimensioning Windows storage systems", in proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (**QEST**), pp. 31-40, Riverside, Sept. 2006. (T.A.: 49%).

[ci26] **Jalil Boukhobza**, Claude Timsit, "On Windows file access modes: a performance study", in proceedings of the Winter International Symposium on Information and Communication Technologies (**WISICT**), pp. 44-49, Cape Town, Jan. 2005. (T.A.: 40%).

[ci27] **Jalil Boukhobza**, Claude Timsit "An I/O simulator for Windows systems", in proceedings of the European Simulation and Modeling Conference (**ESM**), EUROSIS, pp. 116-121, Paris, Oct. 2004.

[ci28] **Jalil Boukhobza**, "A parameter extraction tool for I/O Windows system characterization", in proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (**MASCOTS**), Poster paper, pp. 626-629, Volendam Oct. 2004.

4.7 Book chapters (2)

[ch1] **Jalil Boukhobza**, "Flashing in the Cloud: shedding some light on NAND flash memory storage systems", in book Data Intensive Storage Services for Cloud Environments, , pp. 241-266, IGI Global Editor, ISBN13: 9781466639348, Apr. 2013

[ch2] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "Flash based storage in embedded systems", in book Embedded Computing Systems: Applications, Optimization, and Advanced Design, pp. 439-455, IGI Global Editor, ISBN13: 9781466639225, Apr. 2013.

4.8 National conferences (7)

[cn1] Frank Singhoff, Alain Plantec, Stéphane Rubini, Hai-Nam Tran, Vincent Gaudel, **Jalil Boukhobza**, Laurent Lemarchand, Shuai Li, Etienne Borde, Laurent Pautet, Jérôme Hugues, Pierre Dissaux, Jérôme Legrand, Christian Fotsing, Blandine Djika, "Teaching real-time scheduling analysis with Cheddar", Actes de l'Ecole d'Eté Temps Réel (**ETR**), Rennes, Aug. 2015.

[cn2] Pierre Olivier, **Jalil Boukhobza**, "FFSMark : Un benchmark pour systèmes de fichiers dédiés aux mémoires flash", Actes des Conférences d'Informatique en parallèlisme, architecture et système (**COMPAS**), Lille, Jul. 2015.

[cn3] Pierre Olivier, **Jalil Boukhobza**, Jean-Philippe Babau, Damien Picard, Stéphane Rubini, "Implémentation matérielle des services d'un RTOS sur circuit reconfigurable", Actes de l'Ecole d'Eté Temps Réel (**ETR**), Brest, Sept. 2011.

[cn4] **Jalil Boukhobza**, Ilyes Khetib, Pierre Olivier, "Flashmon: un outil de trace pour les accès à la mémoire flash NAND", Actes du 1er workshop Embed With Linux (**EWiLi)**, St Malo, May 2011.

[cn5] Pierre Olivier, **Jalil Boukhobza**, "Un mécanisme de cache pour les E/S séquentielles en mémoires flash", Actes du Symposium d'Architectures nouvelles de machines (**SympA**), Saint Malo, May 2011.

[cn6] Loïc Lagadec, **Jalil Boukhobza**, Alain Plantec, "Chaîne de programmation pour architecture hétérogène reconfigurable", Actes du Symposium d'Architectures nouvelles de machines (**SympA**), Fribourg, Feb., 2008.

[cn7] **Jalil Boukhobza**, Claude Timsit, " Simulation de traces réelles d'E/S disque de PC", Actes des Conférence Française en Systèmes d'Exploitation (**CFSE**), Perpignan, Oct. 2006.

4.9 Workshops (13)

[w1] Hamza Ouarnoughi, **Jalil Boukhobza**, Frank Singhoff, Stéphane Rubini and Erwann Kassis, Considering I/O Processing in CloudSim for Performance and Energy Evaluation, The Workshop on Performance and Scalability of Storage Systems (**WOPSSS**), Frankfurt, 2016.

[w2] Mourad Bouache, John L Glover, **Jalil Boukhobza**, Analysis of Memory Performance: Mixed Rank Performance across Microarchitectures, The Workshop on Performance and Scalability of Storage Systems (**WOPSSS**), Frankfurt, 2016.

[w3] Hai Nam Tran, Stéphane Rubini, Frank Singhoff and **Jalil Boukhobza**, Adapting a Fixed Priority Assignment Algorithm to Real-time Embedded Systems with Cache Memory, in proceedings of the **GDR SoC-SiP Workshop**, Nantes, Jun. 2016

[w4] Arezki Laga, Michel Koskas, **Jalil Boukhobza**, Frank Singhoff, Claude Brisson, "Revisiting External Sorting Algorithm On Flash Based Storage Systems", Performance and Scalability of Storage Systems (**Per3S**) workshop, Versailles, Jan. 2016.

[w5] Pierre Olivier, **Jalil Boukhobza**, "FFSMark: a Postmark extension for dedicated flash file systems", the 4th IEEE Non-Volatile Memory System and Applications Symposium (**NVMSA**), Poster, Hong Kong, Aug. 2015.

[w6] Hamza Ouarnoughi, **Jalil Boukhobza**, Frank Singhoff, and Stéphane Rubini, "A multi-level I/O tracer for timing and performance storage systems in IaaS Cloud, Real-time and distributed computing in emerging applications", 3rd IEEE Real-Time and Distributed Computing in Emerging Applications (**REACTION**), pp.1-8, Rome, Dec. 2014.

[w7] Hai Nam Tran, Frank Singhoff, Stéphane Rubini, **Jalil Boukhobza**., "Integration of cache related preemption delay analysis in priority assignment algorithm", the 4th Embedded Operating System Workshop (**EWiLi**), short paper, Lisbon, Nov. 2014.

[w8] Yahia Benmoussa, **Jalil Boukhobza**, Eric Senn, Djamel Benazzouz, "Evaluation of the performance/energy overhead in DSP video decoding and its implications", in proceedings of the **GDR SoC-SiP** Workshop, Lyon, Jun. 2013.

[w9] Pierre, Olivier, **Jalil Boukhobza**, Eric Senn, "Toward a unified performance and power consumption NAND flash memory model of embedded and solid state secondary storage systems", in proceedings of the **GDR SoC-SiP** Workshop, Lyon, Jun. 2013.

[w10] Yahia Benmoussa, **Jalil Boukhobza**, Yassine Hadjadj Aoul, Loïc Lagadec, Djamel Benazzouz, "Behavioral system level power consumption modeling of mobile video streaming applications", in proceedings of the **GDR SoC-SiP** Workshop, Paris, Jun. 2012

[w11] Pierre Olivier, **Jalil Boukhobza**, Eric Senn, "performance evaluation of flash file-systems", in proceedings of the **GDR SoC-SiP** Workshop, Paris, Jun. 2012

[w12] **Jalil Boukhobza**, Loic Lagadec, Alain Plantec, Jean-Christophe Lelann, "CDFG platform in MORPHEUS", AETHER - MORPHEUS Workshop **(AMWAS)**, Paris, Oct. 2007.

[w13] **Jalil Boukhobza**, Claude Timsit « Performance des lectures de fichiers de grande taille sur des systèmes Windows » 8th Atelier d'Evaluation de Performance (**AEP**), Reims, May 2003.

5 REFERENCES

[1] F. Thoma, M. Kuhnle, P. Bonnot, E. M Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schuler, K. D. Müller-Glaser, J. Becker, MORPHEUS: Heterogeneous Reconfigurable Computing, in Proceedings of the International Conference on Field Programmable Logic and Applications (**FPL**)., pp.409-414, Aug. 2007

[2] G. De Micheli, Synthesis and Optimization of Digital Circuits. McGraw-Hill, Jan. 1994.

[3] S. Gupta, N. D. Dutt, R. Gupta, Spark: A Parallelizing Approach to the High-Level Synthesis of Digital Circuits. Springer, Jun. 2004.

[4] B. Pottier, J. Boukhobza, T. Goubier, An integrated platform for heterogeneous reconfigurable computing, in proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (**ERSA**), pp. 25-36, Las Vegas, Jun. 2007.

[5] L. Lagadec, J. Boukhobza, A. Plantec, Chaîne de programmation pour architecture hétérogène reconfigurable, Actes du Symposium d'Architectures nouvelles de machines (**SympA**), Fribourg, Feb., 2008.

[6] J. Boukhobza, L. Lagadec, A. Plantec, J-C. Lelann, CDFG platform in MORPHEUS, AETHER - MORPHEUS Workshop (**AMWAS**), Paris, Oct. 2007.

[7] ISO 10303-1. Part 1: Overview and fundamental principles, 1994.

[8] ISO 10303-11. Part 11: edition 2, EXPRESS Language Reference Manual, 2004.

[9] What can you do with Platypus?. http://dossen.univ-brest.fr/apl/index.php/Platypus, Accessed on May 2017.

[10] E. Lenormand and G. Edelin, An industrial perspective: pragmatic high-end signal processing environment at Thales, in Proceedings of the 3rd International Workshop on Synthesis, Architectures, Modeling and Simulation (**SAMOS**), 2003.

[11] International Technology Roadmap for Semiconductors. Design. (2012). https://www.dropbox.com/sh/49tu7ip2lsf4922/AAALux-uU0oD70A6-1QTV46za/2012Chapters?dl=0&preview=2012Overview.pdf, Accessed on May 2017.

[12] I. Trashanski, SSD revolution in client devices and data centers, Flash memory Summit, Santa Clara, Aug. 2013.

[13] M. Broussely, G. Archdale, Li-ion batteries and portable power source prospects for the next 5-10 years. Journal of Power Sources, Volume 136, issue 2, pp. 386-394, Oct. 2004.

[14] Cisco, Cisco visual networking index: Global mobile data traffic forecast update. 2013. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Accessed on May 2017.

[15] OOYALA, Ooyala Global Video Index Q2 2013. 2013. http://go.ooyala.com/wf-video-index-q2-2013.html. Accessed on May 2017

[16] Y. Benmoussa, J. Boukhobza, E. Senn, D. Benazouz, GPP Vs DSP: A performance/energy characterization and evaluation of video decoding, in proceedings of the 21st IEEE International Symposium on Modeling and Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 273-282, San Francisco, Aug. 2013.

[17] Y. Benmoussa, J. Boukhobza, E. Senn, D. Benazouz, Energy consumption modeling of H264/AVC video decoding for GPP and DSP, in proceedings of the EUROMICRO Digital System Design (**DSD**), Santander, pp. 890-897, Sept. 2013.

[18] Y. Benmoussa, J. Boukhobza, E. Senn, Y. Hadjadj-Aoul, D. Benazouz, A methodology for performance/energy consumption characterization and modeling of video decoding on heterogeneous SoC and its applications, Journal of Systems Architecture, Elsevier, Volume 61, Issue 1, pp.49-70, Jan. 2015.

[19] Y. Benmoussa, J. Boukhobza, E. Senn, Y. Hadjadja-Aoul, D. Benazouz, DyPS: dynamic processor switching for energy-aware video decoding on multi-core SoCs, ACM SIGBED Review, Volume 11, Issue 1, pp. 56-61, Feb. 2014.

[20] P. Olivier, J. Boukhobza, E. Senn, Revisiting read-ahead efficiency for raw NAND flash storage in embedded Linux, ACM SIGBED Review, Volume 11, Issue 4, pp. 43-48, Dec. 2014.

[21] P. Olivier, J. Boukhobza, E. Senn, On benchmarking embedded Linux Flash File Systems, ACM SIGBED Review, Volume 9, Issue 2, pp. 43-47, Jun. 2012.

[22] P. Olivier, J. Boukhobza, E. Senn, Micro-benchmarking Flash memory file-system wear leveling and garbage collection: a focus on initial state impact, in proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (**EUC**), pp. 437-444, Paphos, Dec. 2012 .

[23] P. Olivier, J. Boukhobza, E. Senn, modeling driver level NAND Flash memory I/O performance and power consumption for embedded Linux, in proceedings of the IEEE 11th International Symposium on Programming and Systems (**ISPS**), pp. 143-152, Algiers, Apr. 2013.

[24] P. Olivier J. Boukhobza, E. Senn, Flashmon V2: Monitoring raw NAND Flash memory I/O requests on embedded Linux, ACM SIGBED Review, Volume 11, Issue 1, pp. 38-43, Feb. 2014.

[25] P. Olivier, J. Boukhobza, M. Soula, M. Le Grand, I. Chaib Draa and E. Senn, A tracing toolset for embedded Linux flash file systems, in proceedings of the 8th EAI International Conference on Performance Evaluation Methodologies and Tools (**ValueTools**), pp. 153-158, Bratislava, Dec. 2014.

[26] D. Woodhouse, JFFS2: the journalling flash file system version 2, in Proceedings of the Linux Symposium, Ottawa, Canada, 2001.

[27] P. Olivier, J. Boukhobza, E. Senn, H. Ouarnoughi, A Methodology for Estimating Performance and Power Consumption of Embedded Flash File Systems, in ACM Transactions on Embedded Computing Systems, Volume 15, issue 5, August 2016.

[28] J. Boukhobza, Flashing in the Cloud: shedding some light on NAND flash memory storage systems, in book Data Intensive Storage Services for Cloud Environments, pp. 241-266, IGI Global Editor, ISBN13: 9781466639348, Apr. 2013

[29] J. Boukhobza, P. Olivier, Flash Memory Integration, Performance and Energy Considerations, 266 pages, ISTE-Elsevier edition, March 2017.

[30] D. Ma, J. Feng, G. Li. A survey of address translation technologies for flash memories. ACM Comput. Surv. Volume 46, issue 3, Jan. 2014.

[31] S. J. Kwon, A. Ranjitkar, Y. Ko, T. Chung, FTL Algorithms for NAND-type flash memories, Design Automation for Embedded Systems (DAEM), Springer, pp. 191-224, Dec. 2011.

[32] H. Jo, J. Kang, S. Park, J. Kim, and J. Lee, FAB: Flash-Aware Buffer Management Policy for Portable Media Players, IEEE Trans. Consumer Electron., Volume 52, Issue 2, pp. 485-493, May. 2006.

[33] S. Kang, S. Park, H. Jung, H. Shim, and J. Cha, Performance Trade-Offs in Using NVRAM Write Buffer for Flash Memory-Based Storage Devices, IEEE Trans. Computers, Volume 58, Issue 6, pp. 744-758, Jun. 2009.

[34] H. Kim, and S. Ahn, BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage, in Proceedings of the 6th USENIX Conference on File and Storage Technologies (**FAST**), Feb. 2008.

[35] J. Boukhobza, P. Olivier, C-lash: a cache system for optimizing NAND flash memory performance and lifetime, in proceedings of the International Conference on Digital Information and Communication Technology and its Applications (**DICTAP**), LNCS Springer CCIS 167, pp. 519-613, Dijon, Jun. 2011.

[36] J. Boukhobza, P. Olivier, An efficient hierarchical dual cache system for nand flash memories, International Journal of Digital Information and Wireless Communications (IJDIWC), The Society of Digital Information and Wireless Communications, Volume 1, Issue 1, pp. 175-194, 2011

[37] J. Boukhobza, P. Olivier, S. Rubini, CACH-FTL: a cache aware configurable hybrid flash translation layer, in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**PDP**), pp. 94-101, Belfast, Feb. 2013.

[38] J. Boukhobza, P. Olivier, S. Rubini, A Scalable and highly configurable cache-aware hybrid flash translation layer, MDPI Computers, Volume 3, Issue 1, pp. 36-57, Mar. 2014.

[39] J. Boukhobza, P. Olivier, S. Rubini, L. Lemarchand, Y. Hadjadj-Aoul, A. Laga, MaCACH: An adaptive cache-aware hybrid FTL mapping scheme using feedback control for efficient page-mapped space management, Journal of Systems Architecture, Elsevier, Volume 61, Issues 3-4, pp.157-171, Mar. 2015.

[40] S. Lee, D. Park, T. Chung, D. Lee, S. Park, and H. Song, A Log Buffer Based Flash Translation Layer Using Fully Associative Sector Translation, ACM Trans. Embedded Computing Sys., Volume 6, Issue 3, pp. 1-27, Jul. 2007.

[41] S. Lee, D. Shin, Y. Kim, and J. Kim, LAST: Locality Aware Sector Translation for NAND Flash Memory Based Storage Systems, ACM SIGOPS Operating Systems Review, Volume 42, Issue 6, pp. 36-42. Oct. 2008.

[42] Y. Guan, G. Wang, Y. Wang, R. Chen, Z. Shao, Block-level Log-block Management for NAND Flash Memory Storage Systems, in Proceedings of the ACM Conference on Languages, Compilers and Tools for Embedded Systems (**LCTES**), pp. 111-120, Jun. 2013.

[43] Q. Wei, B. Gong, S. Pathak, B. Veeravalli, L. Zeng, and K. Okada, WAFTL: A Workload Adaptive Flash Translation Layer with Data Partition, in Proceedings of 27th IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), May. 2011.

[44] A. Gupta, Y. Kim, B. Urgaonkar, DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings, in Proceedings of the 14th ACM International conference on Architectural support for programming languages and operating systems (**ASPLOS**), 2009.

[45] A. Leventhal, Flash Storage Memory, Communications of the ACM, Volume 51, Issue 8, Jul. 2008.

[46] S. Rubini, J-P. Babau, J. Boukhobza, A hardware/software CBSE framework for RTOS Services: the timing service case study, In proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (**EUC**), pp. 570-577, Paphos, Dec. 2012 .

[47] M. Anne, R. He, T. Jarboui, M. Lacoste, O. Lobry, G. Lorant, M. Louvel, J. Navas, V. Olive, J. Polajovic, J. Pulou, M. Poulhies, S. Seyvoz, J. Tous, T. Watteyne, Think: View-based support of non-functional properties in embedded systems, in Proceedings of the 6th International Conference on Embedded Software and Systems (**ICESS**), pp. 147 – 156, May. 2009.

[48] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, J. Stéfani, An Open Component Model and its Support in Java, in Proceedings of the 7th International Symposium on Component-Based Software Engineering (**CBSE**), Springer, Ed., 2004, p. 7-22, May 2004.

[49] S. Hauck, A. DeHon. Reconfigurable Computing. Morgan Kauffman publishers, Elseiver, 2008.

[50] R. David, D. Lavenier, S. Pillement. Du microprocesseur au circuit FPGA : Une analyse sous l'angle de la reconfiguration. TSI. Technique et science informatiques Volume 24, Issue 4, pp. 395-422, 2005.

[51] P. Ranganathan, From microprocessors to nanostores: rethinking data-centric systems, Computer 44 (1) (2011) 39–48.

[52] R. Winter, Why Are Data Warehouses Growing So Fast?, 2008. from http://searchdatamanagement.techtarget.com/ news/2240111227/Why-Are-Data-Warehouses-Growing-So-Fast, Accessed on May 2016.

[53] D. Farmer, Study Projects Nearly 45-Fold Annual Data Growth by 2020, EMC press release. 2010. from http://www.emc.com/about/news/press/2010/20100504-01.htm. Accessed on May 2017.

[54] S. Park, Y. Kim, B. Urgaonkar, J. Lee, E. Seo, A Comprehensive Study on Energy Efficiency of Flash Memory Storages, Journal of Systems Architecture (JSA), Volume 57, Issue 4, pp. 354-365, 2011.

[55] A. J. Shah, C. D. Patel, Cost model for planning, development and operation of a data center, HP Labs Palo Alto, Tech. Rep., 2005.

[56] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter, in Proceedings of International conference on Algorithms and architectures for parallel processing (**ICA3P**), pp. 372-384, Oct. 2011.

[57] H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, A multi-level I/O tracer for timing and performance storage systems in IaaS Cloud, Real-time and distributed computing in emerging applications, 3rd IEEE Real-Time and Distributed Computing in Emerging Applications (**REACTION**), pp.1-8, Rome, Dec. 2014.

[58] H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, A cost model for virtual machine storage in Cloud IaaS", in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**PDP**), pp. 664-671, Heraklion, Feb. 2016.

[59] D. Boukhelef, J. Boukhobza, K. Boukhalfa, A Cost Model for DBaaS Storage", 27th International Conference on Database and Expert Systems Applications (**DEXA**), pp. 223-240, Porto, Sep. 2016.

[60] D. Boukhelef, K. Boukhalfa, J. Boukhobza, H. Ouarnoughi, Laurent Lemarchand, "COPS: Cost Based Object Placement Strategies on Hybrid Storage System for DBaaS Cloud", The 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**CCGRID**), May 2017.

[61] I. Naas, P. Raipin, J. Boukhobza, L. Lemarchand, iFogStor: an IoT Data Placement Strategy for Fog Infrastructure, the IEEE International Conference on Fog and Edge Computing (**ICFEC**), May 2017.

[62] H. Ouarnoughi, J. Boukhobza, P. Olivier, L. Plassart, L. Bellatreche, "Performance analysis and modeling of SQLite embedded databases on flash file systems", Design Automation for Embedded Systems, Springer, Volume 17, Issue 3, pp 507-542, Sept. 2013.

[63] L. Bellatreche , C. Salmi, S. Breb , A. Kerkad , A. Boukorca , J. Boukhobza, "How to exploit the device diversity and database interaction to propose a generic cost model", in proceedings of the 17th International Database Engineering & Applications Symposium (**IDEAS**), pp. 142-147, Barcelona, Oct. 2013

[64] A. Laga, J. Boukhobza, F. Singhoff, M. Koskas, "MONTRES : Merge ON-The-Run External Sorting Algorithm For Large Data Volumes On SSD Based Storage Systems", in IEEE Transactions on Computers, Volume 66, Issue 10, pp. 1689-1702, 2017.

[65] A. Laga, J. Boukhobza, M. Koskas, F. Singhoff, "Lynx: A Learning Linux Prefetching Mechanism For SSD Performance Model", The 5th Non-Volatile Memory Systems and Applications Symposium (**NVMSA**), Daegu, Aug. 2016.

[66] C. J. Xue, Y. Zhang, Y. Chen, G. Sun, J. J. Yang, H. Li, Emerging non-volatile memories: opportunities and challenges, in Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (**CODES+ISSS**), pp. 325-334, 2011.

[67] J. S. Vetter, S. Mittal, Opportunities for Nonvolatile Memory Systems in Extreme-Scale High-Performance Computing. Computing in Science and Engineering, Volume 17, Issue 2, pp. 73-82, 2015.

[68] A. Carroll, G. Heiser, An analysis of power consumption in a smartphone, in Proceedings of the USENIX Annual Technical Conference (**ATC**), pp. 21–28, Jun. 2010.

[69] A. Carroll, G. Heiser, The systems hacker's guide to the galaxy energy usage in a modern smartphone, in: Proceedings of the 4th Asia-Pacific Workshop on Systems (**APSys)**, pp. 5:1–5:7, 2013.

[70] T. Burd, R. Brodersen, Energy efficient CMOS microprocessor design, in Proceedings of the 28th Hawaii International Conference on System Sciences (**HICSS**), 1, 1995, pp. 288–297, 1995.

[71] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, V. Narayanan, Leakage current: Moore's law meets static power, Computer 36 (12), pp. 68–75, 2003.

[72] K. Moiseev, A. Kolodny, S. Wimer, Timing-aware power-optimal ordering of signals, ACM Trans. Des. Autom. Electron. Syst. 13 (4) pp. 65:1–65:17, Oct. 2008.

[73] A. Chandrakasan, S. Sheng, R. Brodersen, Low-power CMOS digital design, IEEE J. Solid-State Circuits 27 (4) pp. 473–484, Apr. 1992.

[74] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, R. Brodersen, Methods for true energy-performance optimization, IEEE J. Solid-State Circuits 39 (8) pp. 1282–1293, 2004.

[75] Texas Instruments, Codec Engine Overhead, 2010. http://processors.wiki.ti.com/index.php/ Codec_Engine_Overhead. Accessed on May 2017..

[76] Z. Ma, M. Xu, Y.-F. Ou, Y. Wang, Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization step-size and its applications, IEEE Transact. Circuits Syst. Video Technol. 22 (5), pp. 671–682, May 2012.

[77] K. Xu, T.-M. Liu, J.-I. Guo, C.-S. Choy, Methods for power/throughput/area optimization of H.264/AVC decoding, J. Signal Process. Syst.60 (1), pp. 131–145, 2010.

[78] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B.C. Lee, S. Richardson, C. Kozyrakis, M. Horowitz, Understanding sources of inefficiency in general-purpose chips, SIGARCH Comput. Archit. News 38 (3), pp. 37–47, Jun. 2010.

[79] G.J. Smit, A.B. Kokkeler, P.T. Wolkotte, M.D. van de Burgwal, Multi-core architectures and streaming applications, in Proceedings of the 2008 international workshop on System level interconnect prediction (**SLIP**), pp. 35–42, Apr. 2008.

[80] A. Wang, A. Chandrakasan, Energy-efficient DSPs for wireless sensor networks, IEEE Signal Process. Mag. 19 (4), pp. 68–78, 2002.

[81] S. Kim, M.C. Papaefthymiou. Reconfigurable low energy multiplier for multimedia system design, in Proceedings of IEEE Computer Society Workshop on **VLSI**, pp. 129–134, Apr. 2000.

[82] S. Borkar, Thousand core chips: a technology perspective, in Proceedings of the 44th Annual Design Automation Conference (**DAC**), pp. 746–749, Jun. 2007.

[83] ARM, big.little processing. http://www.arm.com/products/processors/technologies/ biglittleprocessing.php. Accessed on May 2017.

[84] Y. Benmoussa, J. Boukhobza, E. Senn, D. Benazouz, On the energy efficiency of parallel multi-core vs hardware accelerated HD video decoding, ACM SIGBED Review, Volume 11, Issue 4, pp. 25-30, Dec. 2014.

[85] Z. Shao, Q. Zhuge, C. Xue, E.H.-M. Sha, Efficient assignment and scheduling for heterogeneous dsp systems, IEEE Transact. Parallel Distrib. Syst. 16 (6), pp. 516–525, 2005.

[86] Y.-H. Wei, C.-Y. Yang, T.-W. Kuo, S.-H. Hung, Y.-H. Chu, Energy-efficient realtime scheduling of multimedia tasks on multi-core processors, in Proceedings of the 2010 ACM Symposium on Applied Computing (**SAC**), pp. 258–262, Mar. 2010.

[87] D. Bautista, J. Sahuquillo, H. Hassan, S. Petit, J. Duato, A simple power-aware scheduling for multicore systems when running real-time applications, in Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (**IPDPS**), pp. 1–7, Apr. 2008.

[88] D. Brooks, V. Tiwari, M. Martonosi, Wattch: a framework for architectural-level power analysis and optimizations, in Proceedings of the 27th International Symposium on Computer Architecture (**ISCA**), pp. 83–94, Jun. 2000.

[89] T. Austin, E. Larson, D. Ernst, Simplescalar: an infrastructure for computer system modeling, Computer 35 (2), pp. 59–67, 2002.

[90] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, N. Jouppi, Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures, in Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (**MICRO**), pp. 469–480, Dec. 2009.

[91] S. Hong, H. Kim, An integrated GPU power and performance model, SIGARCH Comput. Archit. News 38 (3), pp. 280–289, Jun. 2010.

[92] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, H.264/AVC baseline profile decoder complexity analysis, IEEE Transact. Circuits Syst. Video Technol. 13 (7), pp. 704–716, 2003.

[93] Z. Ma, H. Hu, Y. Wang, On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding, IEEE Transact. Multimedia 13 (6), pp. 1240–1255, Dec. 2011.

[94] X. Li, Z. Ma, F. Fernandes. Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming. IEEE Visual Communications and Image Processing (**VCIP**), pp. 1–6, Nov. 2012

[95] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the h.264/avc standard, IEEE Transact. Circuits Syst. Video Technol. (2007) 1103–1120

[96] K. Choi, R. Soma, M. Pedram, Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade off based on the ratio of offchip access to on-chip computation times, IEEE Trans. Comput. Aided Des. Integrated Circuits Syst. 24 (1), pp. 18–28, Jan 2005.

[97] J. Choi, H. Cha, Memory-aware dynamic voltage scaling for multimedia applications, IEEE Proc. Comput. Digital Tech. 153 (2), pp. 130–136, March 2006.

[98] Y. Benmoussa, E. Senn, J. Boukhobza, D. Benazouz,M. Lanoe. Open-PEOPLE, A collaborative platform for remote & accurate measurement and evaluation of embedded systems power consumption, in proceedings of the IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 498-501 Paris, Sept. 2014.

[99] T. Wiegand, G. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (7), pp. 560–576, Jul. 2003.

[100] Texas Instruments, OMAP3530 Power Estimation Spreadsheet, 2012a. http://processors.wiki.ti.com/index.php/OMAP3530PowerEstimationSpreadsheet. Accessed on Jan. 2016.

[101] V. Pallipadi, A. Starikovskiy, The ondemand governor: past, present and future, in Proceedings of the Linux Symposium, pp. 223–238, Ottawa, Jul. 2006.

[102] C.M. Don Darling, B. Singh, Gstreamer on texas instruments OMAP35x processors, in Proceedings of the Linux Symposium, pp. 69–78, Ottawa, Jul. 2009.

[103] L. Merritt, R. Vanam. x264: A High Performance H.264/AVC Encoder, 2006 M. Alvarez, E. Salami, A. Ramirez, M. Valero, A performance characterization of high definition digital video decoding using h.264/avc, in proceedings of the IEEE International Workload Characterization Symposium (IISWC), pp. 24–33, Austin, Oct 2005.

[104] K. Livingston, N. Triquenaux, T. Fighiera, J. C. Beyler, W. Jalby, Computer using too much power? Give it a REST (Runtime Energy Saving Technology). Computer Science - R&D, 29(2): 123-130, 2014.

[105] Z. Xu, S. Sohoni, R. Min, Y. Hu, An analysis of cache performance of multimedia applications, IEEE Trans. Comput. 53 (1), pp. 20–38, Jan. 2004.

[106] N.T. Slingerland, A.J. Smith, Cache performance for multimedia applications, in Proceedings of the 15th International Conference on Supercomputing (**ICS)**, pp. 204–217, Jun. 2001.

[107] M. Alvarez, E. Salami, A. Ramirez, M. Valero, A performance characterization of high definition digital video decoding using h.264/avc, in: Workload Characterization Symposium, 2005. Proceedings of the IEEE International, pp. 24–33, Oct 2005.

[108] M.J. Holliman, E.Q. Li, Y. kuang Chen, Mpeg decoding workload characterization, in Proceedings of Workshop on Computer Architecture Evaluation Using Commercial Workloads (**CAECW**), 2003.

[109] G. Keramidas, V. Spiliopoulos, S. Kaxiras, Interval-based models for run-time DVFS orchestration in superscalar processors, in Proceedings of the 7th ACM International Conference on Computing Frontiers (**CF**), pp. 287–296, 2010.

[110] V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth, S. Kaxiras, Introducing dvfs-management in a full-system simulator, in Proceedings of the 21st IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 535–545, Aug 2013.

[111] T.E. Carlson, W. Heirman, L. Eeckhout, Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation, in: Proceedings of the International Conference for High Performance computing, Networking, Storage and Analysis (**SC),** pp. 52:1–52:12, Nov. 2011.

[112] J. Levon, P. Elie, Oprofile: A System Profiler for Linux, 2004. URL: http://oprofile.sf.net. Accessed on May. 2017.

[113] V. Pallipadi, cpuidle – do nothing, efficiently, in Proceedings of the Linux Symposium, Ottawa, Jul. 2007.

[114] T. Stockhammer, Dynamic adaptive streaming over HTTP: standards and design principles, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 133–144, 2011.

[115] V. Gutnik, A. Chandrakasan, Embedded power supply for low-power dsp, IEEE Trans. Very Large Scale Integration (VLSI) Syst. 5 (4), pp. 425–435, 1997.

[116] Z. He, Y. Liang, L. Chen, I. Ahmad, D. Wu, Power-rate-distortion analysis for wireless video communication under energy constraints, IEEE Trans. Circuits Syst. Video Technol. 15 (5), pp. 645–658, 2005.

[117] Y. Benmoussa, E. Senn, N. Derouineau, N. Tizon J. Boukhobza, Green Metadata Based Adaptive DVFS for Energy Efficient Video Decoding, Power And Timing Modeling, Optimization and Simulation (**PATMOS**) Bremen, 2016.

[118] X. Wang, K. Ma, Y. Wang, Adaptive power control with online model estimation for chip multiprocessors, IEEE Transact. Parallel Distrib. Syst. 22 (10), pp. 1681–1696, 2011.

[119] Y. Benmoussa. Performance and Energy Consumption Characterization and Modeling of Video Decoding on Multi-core Heterogenous SoC and their Applications. Multimedia. Université de Bretagne Occidentale, 2015.

[120] A. Bityutskiy, JFFS3 design issues. 2005. http://idke.ruc.edu.cn/people/dazhou/Papers/JFFS3design.pdf. Accessed on May 2017

[121] M. Bjørling, L. Le Folgoc, A. Mseddi, P. Bonnet, L. Bouganim, B. Jonsson, Performing sound flash device measurements: some lessons from uFLIP. In Proceedings of the ACM SIGMOD International Conference on Management of data. ACM, pp. 1219–1222, Jun. 2010.

[122] D. P. Bovet M. Cesati. Understanding the Linux kernel. O'Reilly Media, Inc., 2005.

[123] A. M. Caulfield, L. M. Grupp, S. Swanson, Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. ACM Sigplan Notices 44, 3, pp. 217–228. 2009.

[124] F. Chen, S. Jiang, X. Zhang. SmartSaver: turning flash drive into a disk energy saver for mobile computers. In Proceedings of the IEEE International Symposium on Low Power Electronics and Design (**ISLPED**), pp. 412–417, Oct. 2006.

[125] S. Choudhuri. Macromodeling and characterization of filesystem energy consumption for diskless embedded systems. Ph.D. Dissertation. Texas A&M University. http://oaktrust.library.tamu.edu/handle/1969.1/295, 2003.

[126] T.-S. Chung, D.-J Park, S. Park, D.-H Lee, S. –H. Lee, H.-J. Song, A survey of Flash Translation Layer. Journal of Systems Architecture 55, 5–6, pp. 332–343, May 2009.

[127] N. Dayan, M. K. Svendsen, M. Björling, P. Bonnet, L. Bouganim. Eagle-Tree: exploring the design space of SSD-based algorithms. Proceedings of the VLDB Endowment 6, 12, pp. 1290–1293. 2013.

[128] I. H. Doh, H. J. Lee, Y. J. Moon, E. Kim, J. Choi, D. Lee, S. H. Noh, Impact of NVRAM write cache for file system metadata on I/O performance in embedded systems, In Proceedings of the 2009 ACM symposium on Applied Computing (**SAC**), pp. 1658–1663, Mar. 2009.

[129] X. Dong, C. Xu, Y. Xie, N. P. Jouppi, Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 31, 7, pp. 994–1007, 2012.

[130] E. Gal, S. Toledo, Algorithms and Data Structures for Flash Memories. ACM Comput. Surv. 37, 2, pp. 138–163. Jun. 2005.

[131] L. M. Grupp, A. M. Caulfield, JCoburn, S. Swanson, E. Yaakobi, P. H. Siegel, J. K. Wolf, Characterizing flash memory: anomalies, observations, and applications, in Proceedings of 42nd Annual IEEE/ACM International Symposium on.Microarchitecture (**MICRO**), pp. 24–33, dec. 2009.

[132] T. Homma, Evaluation of Flash File Systems for Large NAND Flash Memory. in Proceedings of the Embedded Linux Conference (**CELF**), Apr. 2009

[133] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, S. Zhang, Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity, in Proceedings of the international conference on Supercomputing (**ICS**), pp. 96–107, May 2011.

[134] A. Hunter, A brief introduction to the design of UBIFS. (2008). http://www.linux-mtd.infradead.org/doc/ubifs whitepaper.pdf. Accessed on Jan. 2016.

[135] J.-Y. Hwang, F2FS: A New File System Designed for Flash Storage in Mobile. http://elinux.org/images/8/81/A New File System Designed for Flash Storage in Mobile.pdf. Accessed on Jan. 2016.

[136] R. Jain, The art of computer systems performance analysis. John Wiley & Sons, 2008.

[137] J. Ji, C. Wang, X. Zhou, System-level early power estimation for memory subsystem in embedded systems, in Proceedings of the IEEE International Symposium on Embedded Computing (**SEC**), pp. 370-375, Oct. 2008.

[138] D. Jung, J. Kim, J.-S. Kim, J. Lee, ScaleFFS: A scalable log-structured flash file system for mobile multimedia systems. ACM Transactions on Multimedia Computing, Communications and Applications 5, 1, 2008.

[139] M. Jung, E. H. Wilson, D. Donofrio, J. Shalf, M. T. Kandemir, NANDFlashSim: Intrinsic latency variation aware NAND flash memory system modeling and simulation, in Proceedings on the IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1-12, Apr. 2012.

[140] Y. Kang, E. L. Miller, Adding Aggressive Error Correction to a High performance Compressing Flash File System, in Proceedings of the 7th ACM International Conference on Embedded Software (**EMSOFT**), pp. 305–314, Oct. 2009.

[141] T. Kgil, T. Mudge, FlashCache: a NAND flash memory file cache for low power web servers, in Proceedings of the international conference on Compilers, architecture and synthesis for embedded systems (**CASES**), pp. 103–112, Oct. 2006.

[142] H. Kim, Y. Won, S. Kang, Embedded NAND flash file system for mobile multimedia devices. IEEE Transactions on Consumer Electronics, 55, 2, pp; 545–552, 2009.

[143] J.-M. Kim, J.-S. Kim, Androbench: Benchmarking the storage performance of android-based mobile devices. In Frontiers in Computer Education, vol. 133 of the series Advances in Intelligent and Soft Computing, Springer, pp. 667–674, 2012.

[144] J. Kim, H. Shim, S.-Y. Park, S. Maeng, J.-S. Kim, FlashLight: A Lightweight Flash File System for Embedded Systems, ACM Trans. Embed. Comput. Syst. 11S, 1, 18:1–18:23. Jun. 2012.

[145] Y. Kim, B. Tauras, A. Gupta, B. Urgaonkar, Flashsim: A simulator for nand flash-based solid-state drives, in Proceedings of the 1st International Conference on Advances in System Simulation (**SIMUL**), pp. 125–131, Sept. 2009.

[146] S. Lee, K. Ha, K. Zhang, J. Kim, J. Kim, FlexFS: A Flexible Flash File System for MLC NAND Flash Memory, in Proceedings of the USENIX Annual Technical Conference (**USENIX**), Jun. 2009.

[147] S.-W. Lee, B. Moon, C. Park, Advances in flash memory SSD technology for enterprise database applications, in Proceedings of the ACM SIGMOD International Conference on Management of data, pp. 863–870, Jul. 2009.

[148] J. Li, A. Badam, R. Chandra, S. Swanson, B. Worthington, Q. Zhang, On the Energy Overhead of Mobile Storage Systems, in Proceedings of the 12th USENIX Conference on File and Storage Technologies (**FAST**), pp. 105–118, Santa Clara, Feb. 2014.

[149] S.-H. Lim, S.-H. Baek, J.-Y. Hwang, K.-H. Park., Write Back Routine for JFFS2 Efficient I/O, In Embedded and Ubiquitous Computing, Edwin Sha, Sung-Kook Han, Cheng-Zhong Xu, Moon-Hae Kim, Laurence T. Yang, and Bin Xiao (Eds.). Number 4096 in Lecture Notes in Computer Science. Springer, pp. 795–804.

[150] S.-H. Lim, K.-H. Park, An efficient NAND flash file system for flash memory storage, IEEE Transactions on Computers, 55, 7, pp. 906–912, 2006.

[151] S. Liu, X. Guan, D. Tong, X. Cheng, Analysis and comparison of NAND flash specific file systems, Chinese Journal of Electronics 19, 3, pp 403-408, 2010.

[152] G. Mathur, P. Desnoyers, P. Chukiu, D. Ganesan, P. Shenoy, Ultra-low power data storage for sensor networks. ACM Transactions on Sensor Networks 5, 4, n° 33, 2009.

[153] Micron Inc. NAND Flash and Mobile LPDDR 168-Ball Package-on-Package (PoP) MCP Combination Memory (TI OMAP) Datasheet, 2009.

[154] Microsoft Research, SSD Extension for DiskSim Simulation Environment. 2009. http://research.microsoft.com/en-us/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4/. Accessed on Jan. 2016.

[155] Mistral Solutions. 2013. OMAP35x Evaluation Module. 2013. http://www.mistralsolutions.com/pes-products/development-platforms/omap35x-evm-.html. Accessed on Jan. 2016.

[156] V. Mohan, S. Gurumurthi, M. R. Stan, FlashPower: A detailed power model for NAND flash memory, in Proceedings of the Conference on Design, Automation and Test in Europe (**DATE**), pp. 502–507, Mar. 2010.

[157] MTD Contributors. 2008. Memory Technology Device General Documentation. (2008). http://www.linux-mtd.infradead.org/doc/general.html, Accessed on Jan. 2016.

[158] National Instruments. 2014. Carte d'acquisition NI PXI-4472B. 2014. http://sine.ni.com/nips/cds/print/p/lang/fr/nid/12184. Accessed on Jan. 2016.

[159] M. Opdenacker, Flash Filesystem Benchmarks, in Proceedings of the Embedded Linux Conference Europe (**ELCE**), Oct. 2010.

[160] R. Pai, B. Pulavarty, M. Cao, Linux 2.6 performance improvement through readahead optimization, in Proceedings of the Linux Symposium, Vol. 2., pp. 105-116, Ottawa, Jul. 2004.

[161] C. Park, J.-U. Kang, S.-Y. Park, J.-S. Kim, Energy-aware Demand Paging on NAND Flash-based Embedded Storages, in Proceedings of the International Symposium on Low Power Electronics and Design (**ISLPED**), pp. 338–343, Aug. 2004.

[162] S.-H. Park, T.-H. Lee, K.-D. Chung, A Flash file system to support fast mounting for NAND Flash memory based embedded systems. In Embedded Computer Systems: Architectures, Modeling, and Simulation. Springer, pp. 415–424, Volume 4017 of the series Lecture Notes in Computer Science, 2006.

[163] S. O. Park, S. J. Kim, ENFFiS: An Enhanced NAND Flash Memory File System for Mobile Embedded Multimedia System. ACM Trans. Embed. Comput. Syst. 12, 2, pp. 23:1–23:13, Feb. 2013.

[164] Y. Park, S. H. Lim, C. Lee, K. H. Park, PFFS: a scalable flash memory file system for the hybrid architecture of phase-change RAM and NAND flash, in Proceedings of the ACM symposium on Applied computing (**SAC**), pp. 1498–1503, Mar. 2008.

[165] J. Reardon, S. Capkun, D. A. Basin, Data node encrypted file system: Efficient secure deletion for flash memory, in Proceedings of the USENIX Security Symposium, pp. 333–348, Aug. 2012.

[166] E. Seo, S.-Y. Park, B. Urgaonkar, Empirical Analysis on Energy Efficiency of Flash-based SSDs, in Proceedings of the conference on Power aware computing and systems (**HotPower),** San diego, Dec. 2008.

[167] K. Sowa, Choosing a Linux File System for Flash Memory Devices, 2011.

[168] Toshiba Inc. 2009. Evaluation of UBI and UBIFS, CE Linux Forum, Oct. 2009.

[169] H.-W. Tseng, H.-L. Li, Ch.-L. Yang, An energy-efficient virtual memory system with flash memory as the secondary storage. In Proceedings of the 2006 International Symposium on Low Power Electronics and Design (**ISLPED**), pp. 418–423, Oct. 2006.

[170] UBIFS Contributors, MTD Website - Raw flash versus FTL devices. 2009.

[171] UBM Tech. 2013. Embedded Market Study, http://images.content.ubmtechelectronics.com/Web/UBMTechElectronics/%7Ba7a91f0e-87c0-4a6d-b861-d4147707f831%7D2013EmbeddedMarketStudyb.pdf. Accessed on Jan. 2016.

[172] D. Woodhouse, JFFS2: The journaling flash file system version 2. In Proceedings of the Linux Symposium, Jul. 2001.

[173] Wookey, YAFFS - A NAND Flash File System. In Proceedings of the Embedded Linux Linux Conference. Europe, 2007.

[174] YAFFS2 Contributors. 2012. Google Android - YAFFS2 Website, http://www.yaffs.net/google-android, 2012. Accessed on Jan. 2016.

[175] S. Yoo, C. Park, Low Power Mobile Storage: SSD Case Study. In Energy-Aware System Design. Springer, pp. 223–246, 2011.

[176] H. Lee, H. Yun, and D. Lee, HFTL: Hybrid Flash Translation Layer Based on Hot Data Identification for Flash Memory, IEEE Trans. Consumer Electron., vol. 55, no. 4, pp. 2005-2011, Nov. 2009.

[177] D. Park, B. Debnath, and D. Du, A Workload-Aware Adaptive Hybrid Flash Translation Layer with an Efficient Caching Strategy, in Proceedings of the IEEE International Symposium on of Modeling, Analysis & Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 248-255, Jul. 2011.

[178] S. Park, J. Park, S. Kim. C. C. Weems, A Pattern Adaptive NAND Flash Memory Storage Structure, IEEE Trans. on Computers, vol.61, no.1, pp.134,138, Jan. 2012.

[179] C. Wang, W. Wong, ADAPT: Efficient Workload-Sensitive Flash Management Based on Adaptation, Prediction and Aggregation, in Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1-12, Apr. 2012

[180] H. Kwon, E. Kim, J. Choi, D. Lee, S. H. Noh. Janus-FTL: Finding the Optimal Point on the Spectrum between Page and Block Mapping Schemes, in Proceedings of the tenth ACM international conference on Embedded software (**EMSOFT**), pp. 169-178, Oct. 2010.

[181] D. Liu, T. Wang, Y. Wang, Z. Qin, Z. Shao, PCM-FTL: A Write-Activity-Aware NAND Flash Memory Management Scheme for PCM-Based Embedded Systems, in Proceedings of the IEEE 32nd Real-Time Systems Symposium (**RTSS**), pp. 357-366, Dec. 2011

[182] G. Wu, B. Eckart, and X. He, BPAC: An Adaptive Write Buffer Management Scheme for Flash-Based Solid State Drives, in Proceedings of 26th IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1-6, May 2010.

[183] B. Debnath, S. Subramanya, D. Du, and D. J. Lilja, Large Block CLOCK (LB-CLOCK): A Write Caching Algorithm for Solid State Disks, in Proceedings of the IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 1-9, Sept. 2009.

[184] J. Hu, H. Jiang, T. Tian, and L. Xu,, PUD-LRU : An Erase-Efficient Write Buffer Management Algorithm for Flash Memory SSD, in Proceedings of the IEEE International Symposium on of Modeling, Analysis & Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp.69-78, Aug. 2010.

[185] D. Seo, and D. Shin, Recently-Evicted-First Buffer Replacement Policy for Flash Storage Devices, IEEE Trans. Consumer Electron., vol. 54, no. 3, pp. 1228-1235, Aug. 2008.

[186] X. Liao, S. Hu, Bridging the Information Gap between Buffer and Flash Translation Layer for Flash Memory, IEEE Trans. on Consumer Electron., vol.57, no.4, pp.1765,1773, Nov. 2011.

[187] J. E. Brewer, and M. Gill, Nonvolatile Memory Technologies with Emphasis on Flash, IEEE Press Series, Wiley Inter-Science, pp. 22-24, 2008.

[188] A. Ban, Flash File System Optimized for Page-Mode Flash Technologies, United States Patent, No 5,937,425, 1999.

[189] C. Wu, and T. Kuo, An Adaptive Two-Level Management for the Flash Translation Layer in Embedded Systems, in Proceedings of the IEEE/ACM international conference on Computer-Aided Design (**ICCAD**), pp. 601-606 , Nov. 2006.

[190] J. Hsieh, Y. Tsai, T. Kuo, and T. Lee, Configurable Flash-Memory Management: Performance versus Overheads, IEEE Trans. Computers, vol. 57, no. 11, pp. 1571-1583, Nov. 2008.

[191] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, A Space-Efficient Flash Translation Layer for Compactflash Systems, IEEE Trans. Consumer Electron., vol. 48, no. 2, pp. 366-375, May. 2002.

[192] Y. Wang, D. Liu, M. Wang, Z. Qin, Z. Shao, Y. Guan, RNFTL: A Reuse-Aware NAND Flash Translation Layer for Flash Memory, in Proceedings of the ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (**LCTES**), pp. 163-172, Stockholm, Apr. 2010.

[193] H. Cho, D. Shin, and Y. I. Eom, KAST: K-Associative Sector Translation for NAND Flash Memory in Real-Time Systems, in Proceedings of Design, Automation and Test in Europe (**DATE**), pp. 507-512, Apr. 2009.

[194] C. Wang, W. Wong, TreeFTL: Efficient RAM management for high performance of NAND flash-based storage systems, in Proceedings of Design, Automation & Test in Europe Conference & Exhibition (**DATE**), pp.374-379, Mar. 2013.

[195] Z Qin, Y. Wang, D. Liu, Z. Shao, Y. Guan, MNFTL: an efficient flash translation layer for MLC NAND flash memory storage systems, in Proceedings of the 48th Design Automation Conference (**DAC**), pp. 17-22, Jun. 2011.

[196] P. Huang, Y. Chang, and T. Kuo, Joint management of RAM and flash memory with access pattern considerations, in Proceedings of ACM/EDAC/IEEE Design Automation Conference (**DAC**), pp. 882-887, Jun. 2012.

[197] X. Hu, R. Haas, The Fundamental Limit of Flash Random Write Performance: Understanding, Analysis and Performance Modelling, Research Report RZ 3771 IBM, Switzerland, 2010.

[198] J. Boukhobza, P. Olivier, S. Rubini, A cache management strategy to replace wear leveling techniques for embedded flash memory, in proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (**SPECTS**), pp. 1-8, IEEE, SCS, The Hague, Jun. 2011.

[199] H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, Integrating I/Os in Cloudsim for Performance and Energy Estimation, ACM SIGOPS Operating Systems Review, Volume 50 Issue 1, pp. 27-36, Dec. 2016

[200] J. Boukhobza, P. Olivier, L. Plassart, H. Ouarnoughi, Ladjel Bellatreche, Embedded databases on flash memories: performance and lifetime issues, the case of SQLite, in Proceedings of Embedded Real-time Software and Systems (**ERTSS),** Toulouse, Feb. 2014.

[201] S. Park, J. Cha, S. Kang, Integrated write buffer management for solid state drives, Journal of Systems Architecture, vol. 60, no. 4, pp. 329-344, Apr. 2014.

[202] L. Chang; Y. Su, Plugging versus logging: A new approach to write buffer management for solid-state disks, in Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (**DAC**), pp. 23-28, Jun. 2011.

[203] T. Wang, D. Liu, Y. Wang, Z. Shao, FTL2: a hybrid flash translation layer with logging for write reduction in flash memory, in Proceedings of the ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (**LCTES**), pp.91-100, Seattle, Jun. 2013.

[204] K.J. Astrom, T. Hagglund, Advanced PID Control. ISA-The Instrumentation, Systems, and Automation Society, ISBN 1556179421, 2006.

[205] X. Hu, E. Eleftheriou, R. Haas, I. Iliadis, R. Pletka, Write Amplification Analysis in Flash-Based Solid State Drives, in Proceedings of the ACM System and Storage Conference (**SYSTOR**), 2009.

[206] R. Gorez, A survey of PID Auto-Tuning Methods: Feature Issue Controller Tuning, Journal A, vol. 38, no.1, pp. 3-10, 1997.

[207] M. Chiao; D. Chang, ROSE: A Novel Flash Translation Layer for NAND Flash Memory Based on Hybrid Address Translation, IEEE Transactions on Computers, vol.60, no.6, pp.753,766, Jun. 2011

[208] G. R. Ganger, B. Worthington, and Y. N. Patt, The Disksim Simulation Environment Version 3.0 Reference Manual, Tech. Report CMU-CS-03-102, Pittsburgh, 2003.

[209] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, Design Tradeoffs For SSD Performance, in Proceedings of the USENIX Annual Technical Conference (**ATC**), pp. 57-70, Jun. 2008.

[210] OLTP Traces - UMass trace repository, http://traces.cs.umass. edu/index.php/Storage/Storage/. Accessed on Jan. 2016.

[211] Cello99 Traces, HP Labs. http://tesla.hpl.hp.com/opensource /cello99. Accessed on Jan. 2016.

[212] D. Narayanan, A. Donnelly, A. Rowstron, Write Off-Loading: Practical Power Management for Enterprise Storage, in Proceedings of the 6th USENIX Conference on File and Storage Technologies (**FAST**), pp. 253-267, Feb. 2008.

[213] S-W. Lee, J-S. Kim, Understanding SSDs with the OpenSSD Platform, Flash Summit, USA, 2011.

[214] J. Knowles, D. Corne, Approximating the Non-dominated Front Using the Pareto Archived Evolution Strategy, Evolutionary Computation, 8:2, pp 149-172, MIT press, 2000.

[215] C. Fonseca, P. Fleming, An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, 3:1, pp 1-16, MIT press, 1995.

[216] Micron, Small Block vs. Large Block NAND Flash Devices, Micron Technical Report TN-29- 07, https://www.micron.com/~/media/documents/products/technical-note/nand-flash/tn2907.pdf, 2007 (Accessed Nov. 2016).

[217] Q. Zhang, L. Chang, R. Boutaba, Cloud Computing: State-of-the-art and Research Challenges, Journal of Internet Services and Applications, 1(1), pp. 7-18, 2010.

[218] J. Carter, K. Rajamani, Designing Energy Efficient Servers and Data Centers, Computer, 43(7), pp. 76-78, 2010.

[219] P. Kogge et al., ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, DARPA Information Processing Techniques Office 2008.

[220] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et M. Zaharia, A View of Cloud Computing, Communications of the ACM, 53(4), pp. 50-58, 2010.

[221] N. Joukov and J. Sipek.. GreenFS: making enterprise computers greener by protecting them better. in Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems (**Eurosys**), pp. 69-80, 2008.

[222] L. A. Barroso, J. Clidaras, U. Hoelzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1, Morgan & Claypool, 2013, pp.154-.

[223] J. Kim, D. Rotem, FREP: Energy proportionality for disk storage using replication, Journal of Parallel and Distributed Computing, 72(8), pp. 960-974, 2012.

[224] T. Bostoen, S. Mullender, and Y. Berbers. Power-reduction techniques for data-center storage systems. ACM Comput. Surv, (45) 3, 2013.

[225] J. Creasey, Hybrid Hard Drives with Non-Volatile Flash and Longhorn, in Proceedings of. Windows Hardware Engineering Conference (**WinHEC**), 2005.

[226] H. J. Lee, K. H. Lee, and S. H. Noh.Augmenting RAID with an SSD for energy relief, in Proceedings of the conference on Power aware computing and systems (**HotPower**), 2008.

[227] D. Lee and K. Koh. Popularity Based Cache Management Scheme for RAID Which Uses an SSD as a Cache, in Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology (**ICCIT**), pp. 913-915. 2009

[228] S. Byan, J. Lentini, A. Madan and L. Pabón, Mercury: Host-side flash caching for the data center, in Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1-12, 2012.

[229] S. Gurumurthi, Architecting Storage for the Cloud Computing Era, in IEEE Micro, 29(6), pp. 68-71, Nov.-Dec. 2009.

[230] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to SSDs: analysis of tradeoffs, in Proceedings of the 4th ACM European conference on Computer systems (**EuroSys**). pp. 145-158, 2009.

[231] W. Felter, A. Hylick and J. Carter, Reliability-aware energy management for hybrid storage systems, in Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1-13, 2011.

[232] G. Zhang, L. Chiu, and L. Liu. Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment, in Proceedings of the IEEE International Conference on Cloud Computing (**CLOUD**), pp. 148-155, 2010.

[233] R. T. Kaushik, L. Cherkasova, R. Campbell, and K. Nahrstedt. Lightning: self-adaptive, energy-conserving, multi-zoned, commodity green cloud storage system, in Proceedings of the ACM International Symposium on High Performance Distributed Computing (**HPDC**), pp. 332-335 , 2010.

[234] L. Lin, Y. Zhu, J. Yue, Z. Cai and B. Segee, Hot Random Off-Loading: A Hybrid Storage System with Dynamic Data Migration, in Proceedings of the IEEE Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 318-325, 2011.

[235] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam. HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs, in Proceedings of the IEEE Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 227-236, 2011.

[236] B. Teabe, A. Tchana and D. Hagimont, Billing the CPU Time Used by System Components on Behalf of VMs, in Proceedings of the IEEE International Conference on Services Computing (SCC), pp. 307-315, 2016.

[237] Q. Yang and J. Ren, I-CASH: Intelligently Coupled Array of SSD and HDD, in Proceedings of the IEEE International Symposium on High Performance Computer Architecture (**HPCA**), pp. 278-289, 2011.

[238] X. Wu and A. L. N. Reddy, Managing storage space in a flash and disk hybrid storage system, in Proceedings of the IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 1-4, 2009.

[239] M. C. Huebscher and J. A. McCann, A survey of autonomic computing–degrees, models, and applications, ACM Comput. Surv., 40(3), pp. 7:1–7:28, 2008.

[240] A. Brunelle and J. Axboe. 2007 blktrace user guide. Available: http://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html, Accessed on Dec. 2016

[241] D. V. Levin, R. McGrath, and W. Akkerman. strace.. Available: http://sourceforge.net/projects/strace/, Accessed on Dec. 2016

[242] B. Jacob, P. Larson, B. H. Leitao, S. A. M. Martins da Silva, SystemTap: Instrumenting the linux kernel for analyzing performance and functional problems, IBM redbooks, Aug. 2009. Available: http://www.redbooks.ibm.com/abstracts/redp4469.html, Accessed on Dec. 2016

[243] G. Aceto, A. Botta, W. De Donato, and A. Pescape, Cloud monitoring: A survey, Comput. Netw., 57 (9), pp. 2093–2115, Jun. 2013.

[244] OpenStack. Ceilometer. Available: https://wiki.openstack.org/wiki/Ceilometer, Accessed on Dec. 2016

[245] N. Enterprises. Nagios overview. Available: http://www.nagios.org/about/overview/, Accessed on Dec. 2016.

[246] O. Cordes. iostat. Available: http://iostat.sourceforge.net, Accessed on Dec. 2016

[247] G. Chazarain. iotop. Available: http://guichaz.free.fr/iotop /, Accessed on Dec. 2016

[248] T. Tso. (2005) The ext2fs library. Available: http://www.giis.co.in/libext2fs.pdf, Accessed on Dec. 2016

[249] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, Virtual machine power metering and provisioning, in Proceedings of the ACM Symposium on Cloud Computing (**SOCC**), pp. 39-50, 2010.

[250] S. Sivathanu, L. Liu, M. Yiduo, and X. Pu, Storage management in virtualized cloud environment, in Proceedings of the International Conference on Cloud Computing (**CLOUD**), pp. 204-211, 2010

[251] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen, Reducing electricity cost through virtual machine placement in high performance computing clouds, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (**SC**), 2011.

[252] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, Process-level power estimation In vm-based systems, in Proceedings of the European Conference on Computer Systems (**EuroSys**), 2015.

[253] Q. Huang, F. Gao, R. Wang, and Z. Qi, Power consumption of virtual machine live migration in clouds, in Proceedings of the International Conference on Communications and Mobile Computing (**CMC**), pp. 122-125, 2011.

[254] N. Kim, J. Cho, and E. Seo, Energy-based accounting and scheduling of virtual machines in a cloud system, in IEEE/ACM International Conference on Green Computing and Communications (**GreenCom**), pp. 176-181, 2011.

[255] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, Vm power metering: Feasibility and challenges, ACM SIGMETRICS Performance Evaluation Review, 38(4), pp. 56-60, 2011.

[256] A. Strunk and W. Dargie, Does live migration of virtual machines cost energy?, in Proceedings of the International Conference on Advanced Information Networking and Applications (**AINA**), pp. 514-521, 2013.

[257] A. Bohra and V. Chaudhary, Vmeter: Power modelling for virtualized clouds, in Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (**IPDPS workshops**), pp. 1-8, 2010.

[258] J. Stoess, C. Lang, and F. Bellosa, Energy management for hypervisorbased virtual machines, in Proceedings of the USENIX Annual Technical Conference (**ATC**), 2007.

[259] D. Gmach, J. Rolia, and L. Cherkasova, Resource and virtualization costs up in the cloud: Models and design choices, in Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (**DSN**), pp. 395-402, 2011.

[260] R. Zhang, R. Routray, D. Eyers, D. Chambliss, P. Sarkar, D. Willcocks, and P. Pietzuch, Io tetris: Deep storage consolidation for the cloud via fine-grained workload analysis, in Proceedings of the International Conference on Cloud Computing (**CLOUD**), pp. 700-707, 2011.

[261] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, A. Sivasubramaniam, Hybridplan: a capacity planning technique for projecting storage requirements in hybrid storage systems, The Journal of Supercomputing, 67(1), 2014.

[262] Z. Li, A. Mukker, and E. Zadok, On the importance of evaluating storage systems' $costs, in Proceedings of the 6th USENIX conference on Hot Topics in Storage and File Systems (**HotStorage**), 2014.

[263] Y. Li and D. D. E. Long, Which storage device is the greenest? Modeling the energy cost of i/o workloads, in Proceedings of the IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (**MASCOTS**), pp. 100-105, 2014.

[264] B. Schroeder and G. A. Gibson, Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you?, in Proceedings of the 5th USENIX conference on File and Storage Technologies (**FAST**), 2007.

[265] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter, J. Network and Computer Applications, 45(C), pp. 108-120, 2014.

[266] Google. Google cloud storage, google prediction api, and google bigquery sla. Available: https://cloud.google.com/storage/sla, Accessed in Sept. 2015.

[267] Microsoft. May 2015, Sla for cloud services. Available: http://azure.microsoft.com/en-us/support/legal/sla/cloud-services/v1_0/, Accessed in Dec. 2016.

[268] A. W. Services. Amazon ec2 service level agreement. Available: https://aws.amazon.com/ec2/sla/, Accessed in Sept. 2015.

[269] W. L. Bircher and L. K. John, Complete system power estimation using processor performance events. IEEE Trans. Computers, 61(4), pp. 563-577, 2012.

[270] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, Modeling hard-disk power consumption, in Proceedings of the USENIX Conference on File and Storage Technologies (**FAST**), pp. 217-230, 2003.

[271] J. Gray, E-Science: The Next Decade Will Be Exciting, may, 2006, from http://signallake.com/innovation/JGrayETH_E_Science.pdf. Accessed on Dec. 2016.

[272] O. Mutlu, Main Memory Scaling: Challenges and Solution Directions, Book Chapter 6 in More than Moore Technologies for Next Generation Computer Design, pp. 127-153, Springer, 2015

[273] J. C. Mogul, E. Argollo, M. Shah, P. Faraboschi, Operating system support for NVM+DRAM hybrid main memory, in Proceedings Of the 12th conference on Hot topics in operating systems (**HotOS**), 2009.

[274] K. Yoongu, R. Daly, J. Kim, C. Fallin, L. Ji Hye, L. Donghyuk, C. Wilkerson, K. Lai, and O.Mutlu, Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors, in Proceeding of the 41st annual international symposium on Computer architecture (**ISCA**), pp. 361-372, 2014.

[275] S. Swanson, A. Caulfield, Refactor, Reduce, Recycle: Restructuring the I/O Stack for the Future of Storage. Computer, 46(8), pp.52-59, Aug. 2013.

[276] M. Bjørling, J. Axboe, D. Nellans, Philippe Bonnet, Linux block IO: introducing multi-queue SSD access on multi-core systems, in Proceedings of the 6th International Systems and Storage Conference (**SYSTOR**), 2013.

[277] Intel. 2015. https://newsroom.intel.com/news-releases/intel-and-micron-produce-breakthrough-memory-technology/. Accessed on March 2016.

[278] J. Cho. 2016. http://www.businesskorea.co.kr/english/news/ict/13818-chasing-intel-samsung-successfully-develops-3d-cross-point-memory?sthash.FqDSwi5t.mjjo, Accessed on March 2016.

[279] A. Rudoff, The Impact of the NVM Programming Model, In Storage Developer Conference (**SDC**) 2013.

[280] S. Mittal, J. S. Vetter, D. Li, A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-volatile On-chip Caches, IEEE Transactions on Parallel and Distributed Systems, 26(6), pp. 1524-1537, 2015.

[281] J. Wang, X. Dong, Y. Xie, N. P. Jouppi, Endurance-aware cache line management for non-volatile caches. ACM Trans. Archit. Code Optim. 11(1), 2014.

[282] F. Xia, D. Jiang, J. Xiong, N. Sun, A Survey of Phase Change Memory Systems, Journal of Computer Science and Technology, 30(1), pp. 121-144, 2015.

[283] A. Suresh, P. Cicotti, L. Carrington, Evaluation of emerging memory technologies for HPC, data intensive applications, in Proceedings of the IEEE International Conference on Cluster Computing (**CLUSTER**), pp. 239-247, 2014.

[284] S. Baek, J. Choi, D. Lee, S. H. Noh, Energy-efficient and high-performance software architecture for storage class memory, ACM Trans. Embed. Comput. Syst. 12(3), 2013.

[285] J. S. Meena, S. M. Sze, U. Chand, T. Tseng, Overview of Emerging Non-volatile Memory Technologies, Nanoscale Research Letters, 9(526), pp 1-33, Sept. 2014.

[286] D. Li, J.S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu, Identifying Opportunities for Byte-Addressable Non-Volatile Memory in Extreme-Scale Scientific Applications, in Proceedings of the IEEE International Parallel and Distributed Processing Symposium (**IPDPS**), pp. 945-956, 2012

[287] SNIA, 2015 http://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.1.pdf. Accessed on March 2016.

[288] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. 41(1), pp. 23-50, 2011.

[289] D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri, Practical Skew Handling in Parallel Joins, in Proceedings of the 18th International Conference on Very Large Data Bases (**VLDB**), pp 27-40, 1992.

[290] G. Graefe. The Value of Merge-Join and Hash-Join in SQL Server, in Proceedings of the 25th International Conference on Very Large Data Bases (**VLDB**), pp. 250-253, 1999.

[291] W. Kang, S. Lee, B. Moon, G. Oh, and C. Min, X-FTL: transactional FTL for SQLite databases, in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (**SIGMOD**), pp 97-108, 2013.

[292] I. Koltsidas and S. D. Viglas, Data management over flash memory, in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (**SIGMOD**), pp. 1209-1212, 2011.

[293] S.S. Lightstone, T. J. Teorey, T. Nadeau, Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[294] S. Nath and A. Kansal, FlashDB: dynamic self-tuning database for NAND flash, in Proceedings of the 6th international conference on Information processing in sensor networks (**IPSN**), pp. 410-419, 2007.

[295] P. O'Neil and G. Graefe, Multi-table joins through bitmapped join indices. SIGMOD Rec. 24(3), pp.8-11, 1995.

[296] S. Park, Flash-aware cost model for embedded database query optimizer. Journal of information science and engineering 29(5), pp. 947–967, 2013.

[297] V. N. Ramarekha Patchigolla, J. Springer, and K. Lutes, Embedded Database Management Performance, in Proceedings of the 2011 Eighth International Conference on Information Technology: New Generations (**ITNG**), 2011.

[298] SQLite Developpers, Journal mode pragma statement. From http://www.sqlite.org/pragma.html#pragma_journal_mode, Accessed Dec. 2016.

[299] UBIFS Developpers UBIFS and linux read-ahead. Fromhttp://www.linux-mtd.infradead.org/doc/ubifs.html#L_readahead, Accessed on Dec. 2016.

[300] D. E. Knuth, The art of computer programming: sorting and searching. Pearson Education, 1998, vol. 3.

[301] G. Graefe, Implementing sorting in database systems, ACM Comput. Surv., 38 (3), 2006.

[302] W. Zhang and P.-A. Larson, Dynamic memory adjustment for external mergesort, in Proceedings of the 3rd International Conference on Very Large Data Bases, ser. (**VLDB**), pp. 376–385, 1997.

[303] P. Andreou, O. Spanos, D. Zeinalipour-Yazti, G. Samaras, P. K. Chrysanthis, Fsort: External sorting on flash-based sensor devices, in Proceedings of the Sixth International Workshop on Data Management for Sensor Networks (**DMSN**), 10:1–10:6, 2009.

[304] V. Estivill-Castro and D. Wood, Foundations for faster external sorting, in Proceedings of the 14th Conference on Foundations of Software Technology and Theoretical Computer Science (**FSTTCS**), pp. 414-425, 1994.

[305] M. Lorrillere, J. Sopena, S. Monnet, and P. Sens, Puma: pooling unused memory in virtual machines for I/O intensive applications. In Proceedings of the 8th ACM International Systems and Storage Conference (**SYSTOR**), Article 1, 11 pages, 2015.

[306] H. Park and K. Shim, Fast: Flash-aware external sorting for mobile database systems, Journal of Systems and Software, 82 (8), pp. 1298–1312, 2009.

[307] Y. Liu, Z. He, Y.-P. P. Chen, and T. Nguyen, External sorting on flash memory via natural page run generation, The Computer Journal, 54(11), pp. 1882–1990, 2011.

[308] T. Cossentine and R. Lawrence, Efficient external sorting on flash memory embedded devices, International Journal of Database Management Systems, 5(1), 2013.

[309] J. Lee, H. Roh, and S. Park, External mergesort for flash-based solid state drives, IEEE Transactions on Computers, 65(5), pp. 1518–1527, 2016.

[310] M. D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, Minmax heaps and generalized priority queues, Communications of the ACM, 29(10), pp. 996–1000, 1986.

[311] B. Jacob, S. Ng, and D. Wang, Memory Systems: Cache, DRAM, Disk. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[312] A. Gulati, C. Kumar, and I. Ahmad, Storage workload characterization and consolidation in virtualized environments, in Workshop on Virtualization Performance: Analysis, Characterization, and Tools (**VPACT**), 2009.

[313] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, Characterization of storage workload traces from production windows servers, in Proceedings of the IEEE International Symposium on Workload Characterization (**IISWC**), pp. 119–128, 2008.

[314] S. Liang, S. Jiang, and X. Zhang, Step: Sequentiality and thrashing detection based prefetching to improve performance of networked storage servers, in Proceedings of International Conference on the Distributed Computing Systems (**ICDCS**), 2007.

[315] B. S. Gill and L. A. D. Bathen, Amp: Adaptive multi-stream prefetching in a shared cache, in Proceedings of the USENIX conference on File and Storage Technologies (**FAST**), 2007.

[316] Y. Joo, J. Ryu, S. Park, and K. G. Shin, Fast: Quick application launch on solid-state drives, in Proceedings of the USENIX conference on File and Storage Technologies (**FAST**), 2011.

[317] A. J. Uppal, R. C. Chiang, and H. H. Huang, Flashy prefetching for high-performance flash drives, in Proceedings of IEEE 28th Symposium on Mass Storage Systems and Technologies (**MSST**), pp. 1–12, 2012.

[318] D. M. Huizinga and S. Desai, Implementation of informed prefetching and caching in linux, in Proceedings. International Conference on Information Technology: Coding and Computing (**ITCC**), pp. 443–448, 2000.

[319] O. Rodeh, H. Helman, and D. Chambliss, Visualizing block IO workloads, ACM Transactions on Storage, 11(2), 2015.

[320] S. Marsland, Machine learning: an algorithmic perspective. 2nd edition, Chapman and Hall/CRC, 2014.

[321] D. Joseph and D. Grunwald, Prefetching using markov predictors, in ACM SIGARCH Computer Architecture News, 25(2), pp. 252–263, 1997.

[322] N. Zhang, J. Tatemura, J. M. Patel, and H. Hacigumus, Yoward cost-effective storage provisioning for DBMSs, The VLDB Journal, vol. 23, no. 2, pp. 329–354, 2014.

[323] M. Canim, G. A. Mihaila, B. Bhattacharjee, K. A. Ross, and C. A. Lang, An Object Placement Advisor for DB2 Using Solid State Storage, Proc. VLDB Endow., vol. 2, no. 2, pp. 1318–1329, 2009.