Vers une meilleure utilisation des énergies renouvelables : application à des bâtiments scientifiques

Inès De Courchelle

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *Date de défense (20/11/2017)* **par :**

Inès de Courchelle

Vers une meilleure utilisation des énergies renouvelables : application à des bâtiments scientifiques

---

JURY

Marie-Pierre Gleizes	Professeur IRIT Toulouse	Président du Jury
Laurent Lefèvre	Chargé de recherche INRIA Lyon	Rapporteur
Jean-Marc Menaud	Professeur Éc. des Mines Nantes	Rapporteur
Anne-Cécile Orgerie	Chargée de recherche IRISA Rennes	Examinateur
Philippe Roose	MCF HDR LIUPPA Anglet	Examinateur
Yann Labit	Professeur LAAS-CNRS Toulouse	Examinateur
Thierry Monteil	Professeur LAAS-CNRS	Directeur de thèse
Georges Da Costa	MCF HDR IRIT Toulouse	Directeur de thèse
Tom Guérout	Chargé de recherche LAAS-CNRS Toulouse	Membre invité

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

IRIT (UMR 5505), LAAS - CNRS (UPR 8001), neOCampus

Directeur(s) de Thèse :

Georges Da Costa, *Thierry Monteil*

Rapporteurs :

Laurent Lefèvre et *Jean-Marc Menaud*

À la mémoire de mon père,

« *Il y a Jimmy Page, Robert Plant, John Paul Jones et Jéro Boam.* »
— P. J

Remerciements

Je tiens tout d'abord à remercier mes encadrants de thèses, Georges Da Costa, Yann Labit et Thierry Monteil. Durant ces trois ans, vous m'avez donné l'opportunité de travailler dans la recherche et l'enseignement, je vous remercie de m'avoir aidée, encouragée, et guidée.

Je remercie également Tom Guérout pour tous ses conseils avisés durant ces trois années. J'ai beaucoup apprécié de partager et discuter avec toi, de ces travaux de recherches.

Un grand merci à l'ensemble des membres de l'équipe SEPIA de l'IRIT, merci à Jean-Marc Pierson et Patricia Stolf et l'équipe SARA du LAAS de m'avoir accueillie.

Je remercie l'opération neOCampus et l'Université de Paul Sabatier qui m'ont permis de réaliser ces travaux de thèses. Je tiens à remercier particulièrement, Marie-Pierre Gleizes pour la confiance qu'elle m'a accordé durant ces trois années, ainsi que dans l'organisation de la journée scientifique de juillet 2017.

Ensuite, beaucoup de remerciements aux enseignants de l'université Paul Sabatier pour m'avoir accordée leur confiance en me laissant participer aux enseignements. Des remerciements tout particulier à Ileana Ober, Wahiba Bahsoun, et Armelle Bonenfant pour votre gentillesse et pour m'avoir donné ces opportunités d'encadrements et d'enseignements.

Mes collègues qui ont enduré mes monologues sur mes chats et, les récits palpitants de mes chats, merci à vous, Rémy, Léo, Imane, Juliette, Gilles, Quentin, Zong (« Go Gunners »), Chaopeng, Gustavo, Nicolas avec un S, Nicola sans S, Santi, Guillaume, François, Ghada, Iovy, Benoît, Hang, Soufian, Hélène ainsi que tous les docteurs et doctorants de l'opération neOCampus, de l'équipe SEPIA et de l'équipe SARA. De plus, je remercie Pascal Berthou pour ces discussions cinématographiques durant les pauses café.

Je tiens à remercier les enseignants de l'Université de Pau et de Pays de l'Adour d'Anglet qui m'ont donné envie de continuer dans l'informatique puis dans la recherche. Durant, ces années d'études à Montaury vous m'avez toujours encouragée dans mon projet professionnel.

Je dédie cette thèse à mon papa et ma maman, qui m'ont soutenue continuellement durant ses années. Sans vous, je n'en serais pas là. Maman tu es incroyable et d'un très grand soutient. Des remerciements aussi à mon grand frère Laurent, Cynthia, mon neveu adoré, Jules et la petite Misty.

Enfin de joyeux remerciements, à Elizabeth qui m'a énormément aidée l'année passée, tu m'as écoutée et supportée. Je tiens à remercier ma sœur Leia qui pendant mes années à Bassussarry m'a beaucoup apportée et qui a accepté sa petite nièce, Elizabeth, au sein de la famille.

Résumé

De nos jours, le *cloud* devient indispensable pour utiliser les services informatiques, mais tous ces services en ligne ont un impact sur notre économie et notre environnement. L'utilisation de centres de calcul est devenue incontournable pour le traitement de l'information et le stockage à grande échelle (*Big Data*). Cependant, de tels systèmes distribués sont énergivores, et ont une influence directe sur l'environnement et leur coût de fonctionnement. Afin de diminuer l'empreinte écologique et la facture d'électricité, l'utilisation d'énergie renouvelable est une solution adoptée. Cependant, l'intégration des énergies renouvelables dans de tels systèmes est complexe en raison de leur forte dépendance à l'environnement naturel. Effectivement, le rendement de production énergétique dépend essentiellement, de la météo, et des horaires de production. Par exemple, les panneaux solaires ne produisent pas d'énergie la nuit, ou lorsqu'un nuage cache le soleil, et ils possèdent un rendement plus ou moins fort lorsque les températures varient. Par contre, les éoliennes peuvent produire le jour comme la nuit mais dépendent de l'intensité du vent ainsi que de leur emplacement géographique. Afin d'améliorer l'efficacité des centres de calcul, de nombreux travaux ont utilisé le stockage d'énergie renouvelable. Cette forme de stockage peut être considérée comme un cas de sécurité lié au mauvais rendement de telles sources renouvelables. L'utilisation de différentes sources d'énergie introduit une nouvelle problématique liée à leur complémentarité. Pour cela la création de réseau intelligent introduit une meilleure intégrité des flux énergétiques, et permet la communication entre les différents systèmes : producteurs d'énergie, et consommateurs d'énergie.

Les travaux de cette thèse portent sur l'optimisation des flux énergétiques et informatiques dans un réseau intelligent ayant pour but d'alimenter un centre de calcul via des énergies renouvelables. Dans cette thèse sont traités les problèmes liés à la mise en commun des informations de types énergétique et informatique dans une contrainte de réactivité forte à travers la création d'une architecture pour un réseau intelligent. La modélisation d'un tel réseau doit permettre la prise de décision de manière dynamique et autonome. L'objectif de cette modélisation, via un réseau intelligent, est l'optimisation des ressources renouvelables afin de diminuer l'empreinte écologique.

La première contribution est la création de RenewSim. C'est un simulateur qui distingue les

différents types de flux informatique et énergétique. RenewSim est basé sur l'utilisation de plusieurs sources d'énergie (fournisseur d'électricité, énergies renouvelables et espaces de stockages). Il s'agit d'un simulateur dédié à l'alimentation d'un centre de calcul via des sources d'énergie renouvelables et non renouvelables. L'objectif de ce simulateur est de proposer la co-simulation afin d'approfondir les modèles de puissance, de consommation des différents composants du réseau intelligent. RenewSim permet de prendre des décisions à travers un module de contrôle unique. C'est un outil paramétrable.

La deuxième contribution porte sur l'analyse d'une charge de travail réelle. Le choix des traces (données d'entrée du système) a un effet décisif sur l'observation des performances d'un centre de calcul. Il doit permettre de garantir l'utilisation des ressources, de mesurer et de respecter le niveau de QoS requis par les utilisateurs, par exemple via des priorités d'exécution. Nous nous sommes basés, pour cela, sur des traces issues des centres de calcul de Google. Son utilisation tend à se rapprocher d'une charge d'utilisation réelle et actuelle. Une étude a permis de créer un ensemble de traces synthétiques représentatives.

La troisième contribution est la mise en place d'une charge de travail modulable à travers l'utilisation et l'adaptation d'algorithmes d'ordonnancement sous contraintes. La contrainte principale est le placement d'une charge à un instant où l'énergie renouvelable est disponible. Les différents algorithmes réalisés sont comparés à l'aide de métriques classiques et spécifiques comme le *GEC* (*Green Energy Coefficient*) qui quantifie la part des énergies renouvelables consommées, au cours d'un scénario, par un centre de calcul.

La quatrième contribution est l'impact environnemental et la performance de la batterie par rapport à sa capacité de stockage et de la production d'énergies renouvelables sur une charge de travail. Les simulations effectuées ont permis de faire augmenter la capacité des espaces de stockage et la production d'énergie renouvelable afin de connaître le ratio où le *GEC* était le plus performant.

Mots-clés : Réseau intelligent - Énergie renouvelable - Centre de Calcul - Ordonnancement

Abstract

Nowadays, the *cloud* becomes essential to use IT services, but all these online services have an impact on our economy and our environment. The use of data centers has become essential for information processing and large scale storage (*Big Data*). However, such distributed systems are energy intensive, and have a direct influence on the environment and their operation cost. In order to reduce the ecological footprint and the electricity bill, the use of renewable energy is a solution adopted. However, the integration of renewable energies in such systems is complex because of their strong dependence on the natural environment. Indeed, the energy production yield depends essentially on the weather and production schedules. For example, solar panels do not produce energy at night, or when a cloud hides the sun, and they perform more or less when temperatures vary. On the other hand, wind turbines can produce both day and night but depend on the intensity of the wind and their geographical location. In order to improve the efficiency of data centers, many works have used renewable energy storage. This form of storage can be considered as a security case related to the poor performance of such renewable sources. The use of different energy sources introduces a new problem related to their complementarity. For that, the creation of intelligent network introduces a better integrity of the energy flows, and allows the communication between the different systems : energy producers, and consumers of energy.

The work of this thesis deals with the optimization of energy and computer flows in an intelligent network aiming to supply a data center via renewable energies. In this thesis are treated the problems related to the pooling of energy and information type information in a strong reactivity constraint through the creation of an architecture for an intelligent network. The modeling of such a network must allow the decision making in a dynamic and autonomous way. The objective of this modeling, via an intelligent network, is the optimization of renewable resources in order to reduce the ecological footprint.

The first contribution is the creation of RenewSim. It is a simulator that distinguishes different types of IT and energy flows. RenewSim is based on the use of several sources of energy (electricity supplier, renewable energy and storage space). It is a simulator dedicated to feeding a data center via renewable and non-renewable energy sources. The objective of this simulator is to propose co-simulation in order to deepen the models of power and consumption of the various components of

the intelligent network. RenewSim allows you to make decisions through a single control module. It is a configurable tool.

The second contribution deals with the analysis of a real workload. The choice of traces (input data of the system) has a decisive effect on the observation of the performances of a data center. It must make it possible to guarantee the use of the resources, to measure and to respect the level of QoS required by the users, for example via execution priorities. To do this, we relied on traces from Google's data centers. Its use tends to be closer to a real and current usage load. A study has created a set of representative synthetic traces.

The third contribution is the establishment of a flexible workload through the use and adaptation of constraint scheduling algorithms. The main constraint is placing a load at a time when renewable energy is available. The different algorithms are compared using classical and specific metrics such as the *GEC* (*Green Energy Coefficient*) which quantifies the share of renewable energies consumed, during a scenario, by a data center.

The fourth contribution is the environmental impact and performance of the battery in relation to its storage capacity and the production of renewable energy on a workload. The simulations made it possible to increase the capacity of the storage spaces and the production of renewable energy in order to know the ratio where the *GEC* was the most efficient.

Keywords : Smart grid - Renewable energy - Data center - Scheduler

Abréviations

CPU **C**entral **P**rocessing **U**nit - Unité centrale.

DC/DC : **D**irect **C**urrent / **D**irect **C**urrent - Courant Continu vers Courant Continu, convertisseur électronique permettant de transformer un courant continu d'un niveau de voltage à un autre.

DC/AC : **D**irect **C**urrent / **A**lternating **C**urrent - Courant Continu vers Courant alternatif, convertisseur électrique permettant de transformer le courant continu vers un courant alternatif.

DVFS : **D**ynamic **V**oltage and **F**requency **S**caling - Ajustement dynamique de la tension.

FF : **F**irst **F**it - algorithme.

FF_LPT : **F**irst **F**it avec **L**ong **P**rocessing **T**ime - algorithme.

GA : **G**enetic **A**lgorithme - algorithme génétique.

GEC : **G**reen **E**nergy **C**oefficient - coefficient d'énergie vert.

IaaS **I**nfrastructure **a**s **a** **S**ervice - infrastructure en tant que service.

NTIC : **N**ouvelles **T**echnologies de l'**I**nformation et de la **C**ommunication.

QoS **Q**uality **O**f **S**ervice - qualité de service.

PE : **P**rimary **E**nergy - énergie primaire.

PUE : **P**ower **U**sage **E**ffectivenes - efficacité d'utilisation de la puissance.

RR : **R**ound **R**obin - algorithme.

RR_CLASSES : **R**ound **R**obin avec **CLASSES** - algorithme.

RR_LPT : **R**ound **R**obin avec **L**ong **P**rocessing **T**ime - algorithme.

UML : **U**nified **M**odeling **L**anguage - langage de modélisation unifié.

vCPU : **V**irtual **C**entral **P**rocessing **U**nit - unité centrale virtuelle.

VM : **V**irtual **M**achine - machine virtuelle.

Table des matières

Table des figures

Chapitre 1

Introduction

Sommaire

1.1 Contexte

1.1.1 Quelques chiffres

De nos jours, le nuage (*cloud*) devient indispensable pour utiliser les services informatiques, mais tous ces services en ligne ont un impact sur notre économie et notre environnement. En outre, cette consommation d'énergie augmente d'année en année. Le fonctionnement d'un centre de calcul est de plus en plus coûteux. Le taux d'utilisation et les besoins des utilisateurs augmentent jour après jour (diffusion, réseau social, et *cloud* avec les services à la demande). La commission européenne a estimé la consommation des centres de calcul dans la zone européenne à 56 milliards de kW en 2008 et a prédit 104 milliards pour 2020 [1].

1. « Code de conduite sur l'efficacité énergétique du centre de données » par la commission européenne en octobre 2008

De nombreuses entreprises se concentrent sur les énergies renouvelables pour réduire leurs factures d'électricité et l'empreinte carbone. De 2010 à 2014, la consommation d'énergie des centres de calcul a augmenté d'environ 4 pour cent aux États Unis [9]. Par exemple, Google a investi 2,5 milliards de dollars dans des projets d'énergie renouvelable pour atteindre 100% d'énergie renouvelable en 2017 [10]. Aujourd'hui, Google a des centres de calcul alimentés par l'usine photovoltaïque dans le désert d'Atacama, le plus grand de ce type en Amérique latine depuis janvier 2017 [11]. En outre, l'utilisation de panneaux solaires est complexe et imprévisible étant donné que la production d'énergie dépend des variations climatiques. Pour réduire l'empreinte carbone, certains travaux ont utilisé le prix de revente pour réduire l'achat d'énergie non renouvelable [12].

Afin d'améliorer l'efficacité des centres de calcul, de nombreux travaux ont utilisé le stockage d'énergie grâce à la batterie [1]. L'utilisation de différentes sources d'énergie introduit une nouvelle problématique liée au stockage de l'énergie. De nombreux travaux existent sur les différents équipements de stockage d'un point de vue de charge/décharge et leur utilisation dans un centre de calcul [13, 14]. Cette forme de stockage peut être considérée comme un cas de sécurité contre la pénurie.

Ces domaines d'alimentation et de consommation d'un centre de calcul constituent le contexte de ses travaux de thèse. Mes travaux s'inscrivent dans la diminution de l'empreinte écologique et la réduction des coûts de fonctionnements des centres de calcul.

1.1.2 neOCampus

L'élaboration de cette thèse s'est effectuée au sein de l'opération neOCampus. Cette opération a été initiée en juin 2013 par le président de l'Université Paul Sabatier (UPS), Bertrand Monthubert, et fait appel aux compétences de plusieurs laboratoires travaillant conjointement avec l'Université de Paul Sabatier. Des domaines tels que la domotique, les bâtiments intelligents et les villes veulent fonctionner en grande partie avec des énergies renouvelables. Les réseaux intelligents font partie de la mise en œuvre de ces bâtiments intelligents à travers la création de réseaux d'électricité interconnectés. Le campus de Paul Sabatier peut être assimilé à une petite ville mais est composé de systèmes complexes, par exemple des capteurs, du matériel pédagogique, des matériaux innovants dans le bâtiment ou du stockage de données à grande échelle. La démarche de neOCampus est donc évolutive, et est fondée sur le croisement des compétences pluridisciplinaires afin de réduire l'empreinte écologique. Par exemple, la facture d'électricité de l'Université de Paul Sabatier III s'élève à 3M/an [2].

2. http://www.amue.fr/fileadmin/amue/patrimoine/actualites/WEBCONF_Ecocampus/Guide_SDEP_-_web.pdf (p. 53)

Les trois objectifs principaux de neOCampus sont :

- Améliorer la qualité de vie ;
- Réduire l'empreinte écologique ;
- Réduire les coûts.

FIGURE 1.1 – Logo de l'opération neOCampus

Le site internet de neOCampus : https://www.irit.fr/neocampus/fr/

Mes travaux de thèse s'inscrivent dans la diminution de l'empreinte écologique réalisé par un centre de calcul sur un campus universitaire.

1.1.3 ADREAM

L'axe stratégique du bâtiment ADREAM est pluridisciplinaire. Ils sont fondés sur quatre domaines : informatique, robotique, automatique et micro et nano systèmes. L'objectif du bâtiment ADREAM est d'obtenir une plateforme expérimentale pour la recherche. Il offre un contexte d'expérimentation pour le développement formel de systèmes, environnements de co-simulation et co-validation, systèmes informatiques autonomiques, systèmes énergétiques, robotique et optimisation et commande.

Le bâtiment est muni d'un ensemble de capteurs et d'actionneurs. Actuellement, il dispose d'un mini centre de calcul dévoué à l'expérimentation. La superficie totale du bâtiment est de $1700m^2$ et la superficie des panneaux photovoltaïques est de $720m^2$.

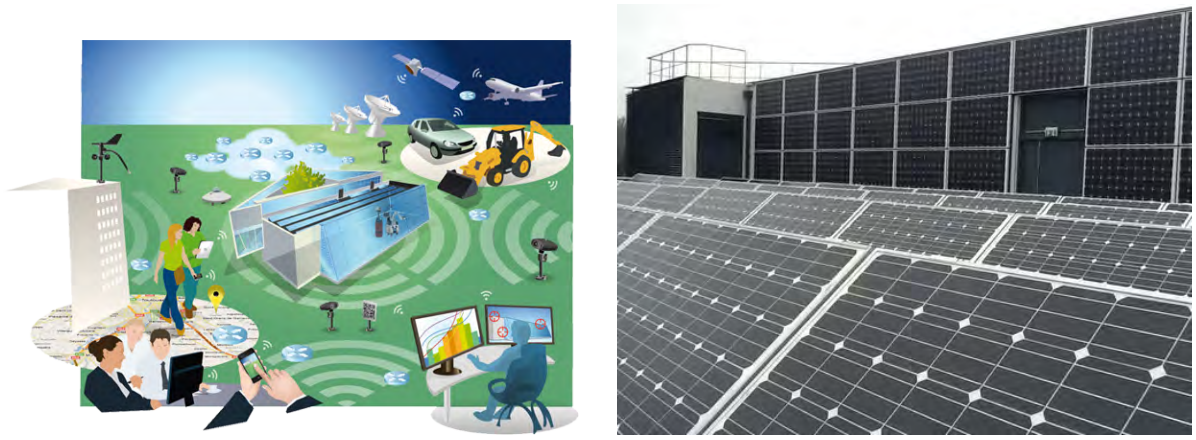


FIGURE 1.2 – Illustrations et panneaux solaires du bâtiment ADREAM

Le site internet du bâtiment ADREAM : https://www.laas.fr/public/fr/adream

Dans ces travaux de thèse, le bâtiment ADREAM m'a permis d'obtenir des informations, techniques de composants et des données d'entrées sur la production photovoltaïque.

1.2 Problématiques de recherche

Les problématiques de cette thèse s'articulent autour de la gestion des énergies renouvelables et leur utilisation pour réduire la consommation énergétique d'un centre de calcul.

Question 1. ***Comment utiliser au mieux les ressources énergétiques renouvelables ?***

Afin, de palier aux rendements incertains des énergies renouvelables, l'utilisation d'espace de stockages peut permettre de sauvegarder et d'utiliser de l'énergie à des instants plus favorables au centre de calcul. Cependant, la prise en compte de plusieurs sources d'énergie dans un tel réseau peut entraîner des problèmes de compatibilité électrique. Par exemple, les puissances des différentes sources d'énergies ne sont pas les mêmes. De plus, l'utilisation des énergies renouvelables et d'espaces de stockage, ne permet pas d'obtenir un centre de calcul autosuffisant. C'est pourquoi, le branchement au réseau d'électricité central est indispensable. L'utilisation de différentes sources peut entraîner des conflits de priorités et influencer la prise de décision. Effectivement, l'achat d'électricité ou la décharge de la batterie, peut entraîner des coûts énergétiques, et influencer le fonctionnement du centre de calcul. Il faut donc être capable de représenter le choix de la source d'énergie dans un modèle de décision.

Question 2. ***Comment diminuer l'empreinte écologique d'un centre de calcul ?***

Le principal défi avec les énergies renouvelables est la production d'énergie non constante qui dépend des conditions météorologiques. Nos défis sont de respecter les contraintes d'ordonnancement et d'énergie des Machines Virtuelles (VMs), de satisfaire la demande des utilisateurs et d'optimiser et rendre les serveurs composants le centre de calcul plus rentables en fonction du temps, de l'énergie disponible et des exigences. Nous pouvons utiliser d'une meilleure façon l'énergie renouvelable grâce à la sauvegarde de stockage (batteries), à la rentabilité de l'énergie produite (ventes) ou à des algorithmes permettant de réduire la consommation et les coûts d'achat. Dans cette thèse, nous nous concentrerons sur une approche d'ordonnancement pour atteindre une corrélation entre la production d'énergie renouvelable et la consommation demandée par la charge de travail (VMs). Cela permet de tendre vers un équilibre entre l'offre (production solaire) et la demande (consommation centre de calcul) et de surcroît diminuer, l'achat d'énergie non renouvelable. Le but est de se rapprocher d'une charge travail agile (adaptable dans le temps et dans l'espace).

Question 3. ***Comment mettre en place un niveau de validation ?***

La validation par des expérimentations réelles est envisageable. Cependant, de telles expérimentations demandent un certain nombre d'équipements physiques de mesure, pas forcement manipulables et modulables. Un moyen de validation est de réaliser une analyse par simulation, permettant d'avoir un aperçu temporel de la consommation énergétique. Plusieurs simulateurs de *Cloud* sont utilisés pour modéliser des infrastructures distribuées. Cependant, de tels simulateurs n'intègrent pas, forcement, les mêmes caractéristiques en termes de fonctionnalités. L'utilisation de données d'entrées réelles apporte de la légitimité aux expériences réalisées. C'est pourquoi une solution est d'utiliser des données réelles de production d'énergie renouvelable et des données réelles de charge de travail de centre de calcul. L'utilisation de données réelles est un moyen de validation.

1.3 Contributions scientifiques

Cette partie présente les différentes contributions réalisées durant ce doctorat.

La création d'un simulateur dédié à l'alimentation d'un centre de calcul via plusieurs sources d'énergie : RenewSim

Cette contribution répond aux questions 1 et 3

Afin de faire face à la consommation énergétique des centres de calcul qui augmentent d'année en année, l'utilisation d'un réseau intelligent permet de diminuer l'empreinte écologique. Le

choix s'est porté vers la création d'un réseau intelligent permettant de gérer un ensemble de flux hétérogènes à travers un module de contrôle. Dans un premier temps, nous avons identifié les différents composants de ce réseau. Après analyse de l'état de l'art, l'utilisation de batteries dans un centre de calcul est fréquemment utilisée afin de réduire les coûts d'électricité. En analysant de telles architectures, plusieurs sources d'énergies dites vertes sont également utilisées dans l'alimentation de tels centres de calcul. Cependant, l'intégration des énergies renouvelables dans de ces systèmes est complexe en raison de leur forte dépendance à l'environnement naturel. C'est pourquoi, ce réseau doit être connecté aux réseaux d'électricité pour palier au rendement des énergies renouvelables. Trois types de sources ont été identifiées : les énergies renouvelables, les espaces de stockages, et le réseaux d'électricité. Cet aspect multi-sources dans ce réseau doit être géré à travers un module de contrôle permettant de régler les conflits de priorités d'utilisation mais aussi de permettre de prendre une décision éco-responsable. La première contribution de cette thèse repose sur une modélisation à haut niveau d'un réseau intelligent, incluant un ensemble de définitions et de modèles permettant d'obtenir un simulateur polymorphe autorisant à terme la co-simulation avec d'autres simulateurs.

L'analyse de traces de centre de calcul à grande échelle pour la création de traces synthétiques

Cette contribution répond à la question 3

La modélisation du réseau intelligent a permis de définir plusieurs simulations avec un module de contrôle utilisant un ensemble de politiques pour l'énergie. Cependant, celui-ci doit être aussi capable de comprendre une charge de travail. Pour cela, une étude d'une charge de travail à grande échelle a été réalisée, afin de permettre de modéliser des besoins et des flux relatifs aux centres de calcul. L'analyse de telles traces, a permis d'étudier le comportement d'une charge de travail en terme de temps d'attente d'un service, de durée d'exécution d'un service, de la qualité de service (QoS), de consommation et d'utilisation de ressources. À travers cette analyse, nous avons modélisé une machine virtuelle ainsi que des traces synthétiques. Elles sont un élément de validation pour la simulation.

Proposition d'utilisation, d'adaptation et de simulation d'algorithmes d'ordonnancement sous contraintes

Cette contribution répond à la question 2

L'utilisation et l'adaptation d'algorithmes ont permis de réaliser un ensemble de simulation afin d'obtenir une charge de travail agile. Cette charge de travail pour le centre de calcul est obtenue via un ensemble d'algorithmes possédant plusieurs contraintes. La première contrainte est la production d'énergie verte disponible et la deuxième contrainte correspond au nombre de

ressources disponibles pour chaque serveur du centre de calcul. Par la suite, des simulations ont été réalisées afin d'évaluer l'impact des différents ordonnancements. La validation a été réalisée à travers le simulateur RenewSim, en dotant, son module de contrôle d'un ordonnanceur. Ce dernier a permis de mettre en corrélation les flux énergétiques et informatiques entre eux. Plusieurs indicateurs ont été mis en place afin de comparer les différents algorithmes, des indicateurs sur le calcul de l'empreinte écologique ainsi que sur la QoS du centre de calcul. Le coefficient d'énergie vert (GEC) a permis de quantifier la part des énergies renouvelables consommées, au cours d'un scénario, par un centre de calcul. De plus, l'utilisation d'algorithmes et de modèles doit nous aider à répondre à des spécifications pour connaître l'algorithme le plus ou moins favorable à une politique de décision par rapport à la QoS attendue, l'aspect environnemental (GEC) ou l'aspect lucratif.

Dimensionnement d'un stockage énergétique par rapport à une production d'énergie verte

Cette contribution répond aux questions 2 et 3

À travers les simulations réalisées, nous avons aussi analysé l'impact environnemental et la performance des éléments d'un réseau intelligent. Ce dimensionnement a été illustré avec une batterie et une production solaire. Il a permis d'étudier les performances de la batterie par rapport à sa capacité de stockage et de la production solaire sur une charge de travail. Les simulations effectuées ont permis de faire évoluer la capacité des espaces de stockage et la production photovoltaïque afin de connaître le ratio où le *GEC* était le plus performant.

1.4 Plan du manuscrit

Ce document est constitué de sept chapitres. Suite à ce chapitre 1, les 6 chapitres suivants s'articulent comme suit :

- **Chapitre 2**
  État de l'art : gestion de l'énergie renouvelable dans un centre de calcul
  Ce chapitre présente de manière assez large un ensemble des domaines abordés dans cette thèse, et le positionnement lié aux défis de cette thèse. Tout d'abord, les principes d'architectures de réseau intelligent pour l'alimentation de centre de calcul. Puis, une présentation des concepts d'algorithmes d'ordonnancements. Ensuite, une présentation des simulateurs ayant pour but d'alimenter des centres de calcul. Enfin, une synthèse des différents concepts abordés est proposée.
- **Chapitre 3**
  Environnement de simulation pour la gestion des flux dans un réseau intelligent
  Ce chapitre présente une analyse d'un réseau intelligent permettant de créer et modéliser

un réseau polymorphe. Cette analyse a donc permis de créer un simulateur RenewSim qui a pour vocation de simuler les flux dans un tel réseau intelligent et de mettre en place un niveau de validation et d'expérimentation. Dans un premier temps est présentée l'approche de conception d'un réseau intelligent et ses objectifs. Dans un deuxième temps, sont analysées les spécifications d'une approche par simulation pour un tel réseau. Dans un troisième temps est introduit les définitions et les différentes caractéristiques de notre réseau intelligent polymorphe. Dans un quatrième temps sont proposés la conception du module de contrôle et de son système de décision. Enfin, une dernière partie expose le bilan et les limites de cette modélisation.

- **Chapitre 4**

  Analyse de charges de travail de serveurs à grande échelle

  Ce quatrième chapitre, va permettre de caractériser une charge de travail d'un centre de calcul à grande échelle, son comportement dans le temps, et dans l'espace (ordonnancement). À travers l'étude de traces de centres de calcul, il est proposé un modèle de traces permettant de mettre en place les algorithmes d'ordonnancements et d'être implémentés dans les différentes expérimentations. Dans une première partie, il est présenté les caractéristiques essentielles afin de générer des traces cohérentes pour valider les différentes contributions. Dans une deuxième partie, l'utilisation des traces Google dans la littérature est exposée. Dans une troisième partie et quatrième partie, les caractéristiques, et l'analyse des traces Google sont exposées. Dans une cinquième partie, une proposition de traces synthétiques est détaillée. Enfin, une dernière partie expose le bilan et les limites des traces Google et des traces synthétiques.

- **Chapitre 5**

  Ordonnancement sous contraintes : maximiser l'utilisation de l'énergie renouvelable

  Ce chapitre introduit la notion de charge de travail agile pour réduire la consommation d'un centre de calcul. Ici, sont présentés un ordonnanceur permettant d'utiliser deux niveaux de contraintes pour la prise de décision, les contraintes énergétiques et les contraintes liées à l'ordonnancement de charges. Il est présenté l'utilisation d'un ensemble d'algorithmes d'optimisation sous contraintes permettant d'ordonnancer une charge de travail sur un centre de calcul en fonction de la production solaire. Afin d'évaluer ces algorithmes un ensemble de métriques est présenté. Dans un premier temps, l'approche d'ordonnancement sous contraintes et les objectifs sont présentés. Dans un deuxième temps, les spécifications d'un tel ordonnanceur sont analysées. Dans un troisième temps, notre approche et les différentes propositions d'adaptation d'algorithmes sous contraintes sont introduites. Dans un quatrième temps, plusieurs métriques existantes d'évaluation de QoS et d'impact sur l'environnement pour les différents algorithmes sont analysées. Enfin, dans une dernière

partie sont exposés le bilan et les limites de tels algorithmes.

- **Chapitre 6**

  Expérimentations : évaluations et comparaisons

  Ce chapitre présente les résultats expérimentaux et se divise en cinq parties. La première partie spécifie l'environnement des expérimentations. La deuxième partie évalue les différentes politiques de décision et l'implémentation du modèle de décision présenté dans le chapitre 3. La troisième partie compare les différents algorithmes présentés dans le chapitre 5 à travers un ensemble de métriques. La quatrième partie est une étude sur le dimensionnement des éléments du réseau intelligent, afin d'établir une corrélation entre eux et leurs influences sur le rendement énergétique vert. Enfin, la dernière partie dresse un bilan des différentes expérimentations réalisées.

- **Chapitre 7**

  Conclusions et perspectives

  Ce dernier chapitre est dédié aux conclusions et perspectives de ces travaux. Il dresse le bilan et les limites des travaux de cette thèse. Les perspectives d'évolution de ces travaux de recherches sont détaillées de l'amélioration des modèles proposés à leur implémentation en environnement réel non maîtrisable.

Chapitre 2

État de l'art : gestion de l'énergie renouvelable dans un centre de calcul

Sommaire

Ce chapitre présente l'état de l'art dans lequel s'inscrit les travaux de cette thèse. Il propose un aperçu de trois domaines : les réseaux intelligents alimentés par des énergies renouvelables, les algorithmes d'ordonnancement liés aux problématiques de l'énergie, et les simulateurs modélisant les flux d'énergies dans des centres de calcul alimentés par des énergies vertes. Les sections de ce chapitre sont :

- Dans une première partie sont décrits les différents réseaux intelligents, qui fournissent à un centre de calcul une consommation plus respectueuse de l'environnement. Ces réseaux intelligents rendent possible le contrôle des flux et aident à connaître les différentes familles de composants qui construisent un tel système.
- Dans une deuxième partie sont présentés différents algorithmes de placement et d'ordonnancement, mettant en avant la réduction de la consommation d'énergie dans un centre de calcul. De tels algorithmes répartissent la charge de travail dans l'espace et dans le temps afin de diminuer les coûts de consommation d'énergie.
- Dans une troisième partie sont analysés différents simulateurs présents dans la littérature qui alimentent des centres de calcul. Cette section permet d'évaluer les différents outils proposés dans la littérature.

Ces trois domaines sont liés entres eux et permettent de situer le positionnement des travaux de cette thèse. Dans un premier temps, les réseaux intelligents aident à atteindre des objectifs pluridisciplinaires, plus particulièrement les objectifs informatiques et énergétiques. La structure d'un réseau intelligent influe directement sur la consommation et l'optimisation. De tels réseaux vont contrôler le cycle complet des flux, de la production d'énergie à la consommation. Dans un deuxième temps, les algorithmes présents dans les ordonnanceurs ne sont qu'une partie de ce cycle. Cependant, ils ont aussi leurs rôles à jouer dans l'optimisation de l'utilisation d'énergie. Dans un troisième temps, il faut un environnement d'expérimentation afin de tester à la fois une structure d'un réseau intelligent et des algorithmes d'ordonnancement. L'utilisation de simulateurs semble plus adéquate pour réaliser diverses expérimentations car elle est moins coûteuse et obtient des résultats plus rapidement que des expérimentations en situations réelles. Les simulateurs vont servir à inscrire et modéliser, plus précisément les algorithmes dans le cycle complet présenté dans les réseaux intelligents. De tels simulateurs aident à modéliser et légitimer le comportement des différents flux informatiques et énergétiques à travers un cycle complet, et par ailleurs, d'être un outil de validation avant l'implémentation dans un environnement réel.

2.1 La composition des centres de calcul verts

Cette partie présente les éléments composants un centre de calcul alimenté via des énergies dites vertes. Pour réduire l'empreinte écologique, certains travaux se portent sur un meilleure utilisation de *framework* et d'infrastructure en tant que service (*infrastructure as a service*, IaaS) [15, 16, 17]. Dans un premier temps, des architectures réelles qui alimentent ces parcs informatiques via des énergies renouvelables des réseaux situés sur le même lieu géographique sont présentées, puis dans un deuxième temps, celles qui sont distribués géographiquement. Dans un troisième temps, les notions de stockage d'énergie sont introduites.

2.1.1 Des architectures de centre de calcul mono site

Lei *et al.* [12] présente une stratégie d'ordonnancement afin d'utiliser au mieux l'énergie verte dans un centre de calcul. Leur ordonnanceur est modélisé dans un réseau intelligent composé de plusieurs sources d'énergie : de la production solaire, de la production éolienne et le réseau d'électricité conventionnel. Leur système vise à diminuer les coûts énergétiques, et à obtenir un taux de QoS satisfaisant pour le centre de calcul. Les résultats démontrent que l'élasticité du coût d'électricité permet d'augmenter le taux d'utilisation de l'énergie renouvelable et de réduire le coût de la consommation d'électricité totale. Les tâches arrivent dans une file d'attente. Elles ont deux niveaux de priorité dites *cruciales* et *non cruciales*. Les tâches *cruciales* ont besoin d'un temps de réponse plus court que les tâches *non cruciales*. Le but final de l'ordonnanceur est d'éviter au maximum les violations de dates limites (*deadlines*). Chaque tâche est placée sur un nœud informatique et ne peut être placée sur plusieurs nœuds. L'ordonnanceur détermine où et quand la tâche peut être exécutée.

Goiri *et al.* [1] ont développé un prototype d'un centre de calcul vert (*Parasol*) ainsi qu'un ordonnanceur (*GreenSwitch*). Ce dernier ordonnance dynamiquement une charge de travail et sélectionne la source énergétique. *Parasol* est composé de batteries, de climatiseur, et de panneaux solaires. Le centre de calcul est composé de 64 serveurs consommant entre 22 à 30 W. Parasol utilise Ganglia [18, 19], un système de surveillance distribué évolutif permettant de suivre l'évolution d'un *cluster* et prendre des décisions sur celui-ci. De plus, Parasol est constitué d'un module de climatisation. La figure 2.1 présente l'architecture de *GreenSwitch*. Il est composé de 3 modules : un pour la prévision (production d'électricité, charges de travail, niveau de la batterie), un pour l'ordonnancement de la charge de travail et l'utilisation des différentes sources d'énergie, et enfin, un pour la configuration d'Hadoop [20].

GreenCloud [2] est une architecture qui a pour but de réduire la consommation d'un centre de calcul tout en garantissant les performances d'un point de vue utilisateur. Le centre de calcul utilise des machines virtuelles (VM) permettant la migration en direct via Xen [21]. Les VMs sont exécutées sur une machine physique et peuvent être migrées sur un autre serveur sans arrêter l'exécution de celles-ci. La figure 2.2 présente l'architecture de GreenCloud. Elle est composée de quatre modules. Un module pour le contrôle des migrations des machines virtuelles (*Migration Manager*), un module pour le contrôles des ressources (*Monitoring Services*), un module de gestion de l'environnement (*Managed Environment*) et un module pour le service des données (*Data Services*). Le module *Migration Manager* autorise la migration en temps réel et la prise de décisions sur le placement des machines virtuelles sur les machines physiques via une heuristique de placement. Le module *Monitoring Services* collecte les données sur les applications de la charge de travail (VMs), et sur l'utilisation des ressources occupées et libres, c'est-à-dire l'état des machines physiques et leur consommation. Par exemple, ce module sait si les serveurs sont

allumés ou éteints, connaît la consommation des serveurs, et quelles sont leurs disponibilités en termes de ressources. Le module *Managed Environment* représente l'ensemble des applications et par surcroît les machines virtuelles, puis les machines physiques et leurs capteurs réalisant les mesures de consommation. Enfin, le module *Data Services*, contient E-Map qui est un service web de type frontal (*front-end*) permettant de connaître en temps réel l'état de GreenCloud. Par exemple, il montre le statut des serveurs, la consommation, les informations relatives à la charge de travail, ou encore les températures. L'objectif de GreenCloud est l'utilisation d'une architecture orientée pour le fonctionnement des migrations de VMs en cours d'exécution afin d'économiser efficacement de l'énergie.

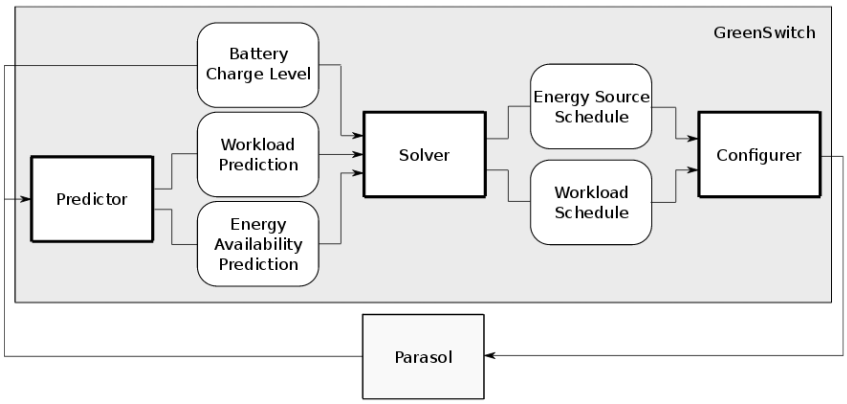


FIGURE 2.1 – Architecture GreenSwitch [1]

SolarCore [22] est une architecture multi-cœurs afin de gérer l'alimentation de processeurs. Ces derniers sont fournis en électricité par des panneaux solaires et le réseau d'électricité. Ce dernier est utilisé que lorsque les panneaux solaires ne produisent pas assez d'énergie.

Stewart *et al.* [23] proposent une architecture multi-sources : batteries, diverses énergies renouvelables et le réseau d'électricité. La source principale est l'énergie renouvelable. Elle peut être de priorité 1 ou 2 (1 étant la plus forte). La deuxième source est l'ensemble des batteries et le réseau d'électricité qui est prioritaire par rapport aux batteries, qui ne sont utilisées qu'en mode secours sur des courtes durées.

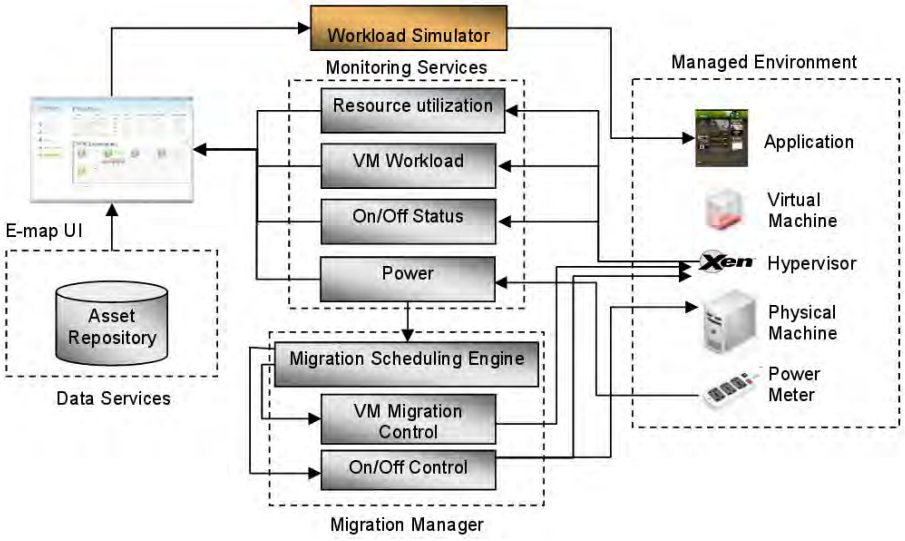


FIGURE 2.2 – Architecture GreenCloud [2]

2.1.2 Des architectures de centres de calcul distribués géographiquement

Zhang et al. [24] ont proposé un intergiciel (*middleware*) appelé GreenWare. Il vise à augmenter le pourcentage d'énergie renouvelable utilisée pour alimenter un réseau de centres de calcul distribués. L'énergie renouvelable utilisée est l'énergie éolienne et l'énergie solaire. Dans cet article, le nuage (*cloud*) est composé comme un ensemble de centres de calcul distribués géographiquement. Les auteurs ont étudié les services afin de minimiser l'empreinte carbone de certaines demandes dans le cadre d'un coût budgétaire fixé par l'opérateur du service Internet.
Dans [25], les auteurs utilisent de nombreux petits centres de calcul géographiquement distribués qui peuvent ajuster dynamiquement la charge de travail où il existe plus de sources renouvelables disponibles.
Liu et al. [26] ont proposé trois algorithmes pour atteindre un équilibre optimal de la charge géographique, améliorer l'utilisation de l'énergie verte et réduire l'énergie des combustibles fossiles.

2.1.3 Les composants de stockage pour améliorer l'utilisation d'un centre de calcul

Les composants de stockages de type Alimentation Sans Interruption (ASI), *Uninterruptible Power Supply* (UPS) ont été utilisés dans le domaine de l'électronique, afin de palier au manque d'énergie dans les centres de calcul. Ils peuvent être utilisés principalement en cas de démarrage ou de pénurie d'énergie sur un court instant [27, 28]. De nos jours, avec l'utilisation des énergies renouvelables, plusieurs types de systèmes de stockages d'énergies sont utilisés [29, 13].

Les batteries sont de plus en plus utilisées dans les centres de calcul afin d'économiser de l'énergie [1, 30]. Cependant, plusieurs études du domaine de l'électronique ont mis en évidence certains facteurs influençant le stockage des batteries :

1. Le lieu de stockage de la batterie (température, humidité ...) influe sur la capacité de stockage et déstockage et sur la durée de vie de la batterie [31, 32] ;
2. La puissance de la batterie influe sur le stockage ainsi que le déstockage de cette dernière ;
3. Le stockage et le déstockage engrangent des pertes d'énergie [33] ;
4. La durée de vie d'une batterie varie suivant son taux utilisation.

Plusieurs approches d'utilisation de batteries sont présentent dans la littérature. Li *et al.* [34] présentent deux approches pour améliorer l'utilisation de l'énergie renouvelable dans les centres de calcul de petites et moyennes tailles. La première approche consiste à placer le maximum de tâches lorsque de l'énergie solaire est disponible. La deuxième approche consiste à stocker le surplus d'énergie solaire et de décharger ce surplus lorsqu'il n'y a plus d'énergie. Le modèle de charge et décharge présenté par les auteurs, est réalisé en fonction de la courbe de production solaire et la demande du centre de calcul. Lorsque la production solaire est supérieure à la demande du centre de calcul, la batterie se charge. Lorsque la production solaire est inférieure à la demande du centre de calcul, la batterie est déchargée pour répondre à la demande.

Les centres de calcul ont besoin d'énergie 24 heures sur 24 et 7 jours sur 7, mais les panneaux solaires fournissent de l'énergie par intermittence et seulement lorsque la météo est satisfaisante. Stewart *et al.* [23] proposent d'allumer et d'éteindre les différentes machines composant le centre de calcul. Leur centre de calcul est donc utilisé avec une batterie de type secours, pour palier à des ressources manquantes.

Les travaux présentés par Münzberg *et al.* [35] portent sur l'évaluation de l'utilisation de trois sources d'énergies (photovoltaïque, batterie et réseau d'électricité) pour alimenter des bâtiments d'habitations ou non. Les auteurs proposent un modèle linéaire permettant de minimiser l'utilisation d'énergie tout en minimisant le prix de l'électricité. Le modèle linéaire proposé est expérimenté sur des scénarios sur une longue durée, pendant 1 an, pour voir l'impact du modèle.

2.1.4 Positionnement et objectifs

Actuellement, la gestion d'un centre de calcul s'équilibre par rapport à l'offre et à la demande de services informatiques. Cependant, l'introduction d'énergie verte dans l'alimentation de tels centres de calcul, ne permet pas de produire de l'énergie de façon linéaire. Les objectifs de cette thèse répondent aux questions et points suivants :

- alimenter un centre de calcul via des énergies renouvelables, par exemple, via de l'énergie solaire. Cette production solaire est incertaine.

- utiliser un espace de stockage d'énergie pour diminuer les coûts de consommation. L'utilisation d'éléments de stockage tel que des batteries pour palier à certaines pénuries d'énergie, amène de nouvelles contraintes en terme de capacité et de stockage. Les batteries peuvent être utilisées à divers instants et pas seulement lorsqu'il y a pénurie d'énergie renouvelable. La prise de décision sur l'utilisation de la batterie doit se faire à court terme. De plus, son utilisation dans un réseau intelligent doté de plusieurs sources d'énergies, peut introduire des conflits d'utilisations de ces sources.
- gérer un centre de calcul dans un réseau intelligent via des énergies vertes dont le lieu de production est situé sur le même lieu géographique. Les architectures proposées dans la littérature peuvent être ou non distribuées sur un lieu géographique différent. Cependant, cela induit des coûts de transports d'énergies d'un point A à un point B, et par surcroît, des pertes d'énergies. De plus, l'utilisation du centre de calcul doit être appliqué dans un contexte de campus innovant, soit sur un seul site.
- exploiter la communication entre la partie informatique et les sources d'énergies, afin de mutualiser l'offre et la demande pour prendre une décision optimale.

2.2 Les ordonnancements et politiques de placements

Cette section présente différents types d'algorithmes et de méthodes utilisés pour ordonnancer et placer des charges de travail sur des centres de calcul. La reconfiguration dynamique donne la possibilité de changer un ordonnancement et déplacer des charges de travail alors que celles-ci sont en cours d'exécution. Les algorithmes dit « Gloutons » et les algorithmes « multi-critères » présentent différentes problématiques, en terme d'optimisation, ou de temps d'exécution. De plus, certains algorithmes sont réalisés à travers des ordonnanceurs possédant une architecture qui leur sont propres.

2.2.1 La reconfiguration dynamique

À travers la reconfiguration dynamique, il est possible de diminuer la consommation d'électricité d'un centre de calcul. Ces reconfigurations peuvent intervenir directement sur l'architecture logicielle, via la migration de services [36, 37], le démarrage/extinction de machines, ou la suspension de services. Ces reconfigurations peuvent se faire directement sur l'architecture matérielle à travers par exemple le DVFS [38, 39]. Villebonnet *et al.* [40], proposent de reconfigurer dynamiquement des machines hétérogènes afin de placer des applications de type web pour minimiser l'utilisation d'énergie. À travers des migrations et l'extinction ou non des machines physiques, les décisions et politiques de placement servent à consommer moins d'énergie tout en satisfaisant les contraintes QoS.

2.2.2 L'utilisation d'algorithmes gloutons

Les algorithmes gloutons (*greedy algorithms*) sont des schémas heuristiques. Ils construisent une solution par itération, sans remettre en cause les décisions prises à l'itération antérieure. Dans certains cas, ces algorithmes réalisent des prises de décisions qui semblent les meilleures à une itération mais qui peuvent par la suite être inappropriées. Cependant, de telles heuristiques permettent d'être mises facilement en œuvre afin d'obtenir des résultats de manière rapide, en cas de contraintes de temps liées à leurs contextes d'exécution.

L'algorithme *Round-Robin* (Tourniquet) [41] est souvent utilisé dans l'ordonnancement de tâches dans les systèmes d'exploitation. Il gère une file d'attente afin d'allouer un temps d'exécution à chacune des tâches de manière cyclique. Dans le cadre de planification dans un environnement de type "nuage", l'algorithme *Round-Robin* permet l'ordonnancement de charges de travail. Cet algorithme construit une solution à chaque tour d'itération. Par exemple, dans le cadre du placement d'une charge de travail constitué de n tâches et de s serveurs, à la première itération la tâche $t1$ sera placée sur un serveur $s1$. A l'itération suivante, la tâche $t2$ sera placée sur le serveur $s2$. Les tâches et les serveurs peuvent être classés par ordre d'arrivée ou de priorité.

Li *et al.* [3] présentent PIKA (*oPportunistic schedulIng broKer infrAstructure*), un framework d'ordonnancement, pour économiser de l'énergie dans un centre de calcul en réduisant la consommation des machines physiques. La figure 2.3 présente son architecture. Il utilise un module de prédiction (*Predictor*) pour l'utilisation de l'énergie verte ainsi qu'un module de re-ajustement (*Gap*) pour dynamiquement remanier le nombre de machines physiques allumées par rapport à la production d'énergie verte réellement réalisée. L'algorithme utilisé pour le placement des jobs détecte, dans un premier temps, les machines physiques surchargées en terme de CPU et de RAM. Puis ensuite, le module de re-ajustement (*Gap*) prend une décision de consolidation, puis chaque VM est placée via un algorithme Gloutons *First Fit Decreasing* [42] combiné à l'optimisation des ressources proposées par PIKA. Cependant, l'utilisation de cet algorithme de placement dans ce framework ne s'applique uniquement que pour des jobs dans un centre de calcul de petite/moyenne taille et situé sur un seul et unique site.

Khosravi *et al.* [43] proposent ECE (*Energy and Carbon-Efficient*) une architecture permettant d'assurer la QoS du système, tout en minimisant l'empreinte carbone via une heuristique de placement. L'heuristique utilisée est une dérivation du best-fit. Les VMs sont placées dans le centre de calcul, en fonction du résultat du PUE [44], et l'augmentation minimale de la consommation d'électricité des serveurs physiques.

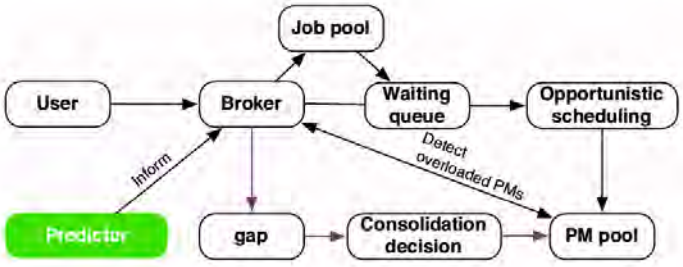


FIGURE 2.3 – Architecture de PIKA (Li *et al.* [3])

2.2.3 Les algorithmes multicritères

L'algorithme génétique, *Genetic Algorithm (GA)*, a été proposé pour la première fois en 1975 par John Holland [45, 46]. L'algorithme génétique s'inspire de la théorie de l'évolution. C'est une méta-heuristique qui évalue diverses solutions à travers une note normalisée. Celle-ci provient du résultat de la fonction objective (fitness) qui étudie chaque individu (solution) qui constitue une population. Les diverses solutions sont appelées « Chromosomes ». L'algorithme génétique a pour principe de faire évoluer la population initiale vers une population qui contiendra de meilleurs chromosomes. Cette évolution se fait à travers divers opérateurs génétiques : mutation, croisement, sélection. la figure 2.4 présente un exemple de chromosome pour un algorithme génétique de placements. Cela génère un ensemble de placements qui peut être modifié à travers les opérateurs génétiques.

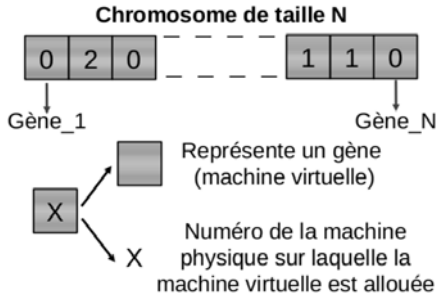


FIGURE 2.4 – Exemple de chromosome dans un algorithme génétique dédié à la problématique de placements de VMs sur un ensemble de machine physique (Guérout *et al.* [4])

Kessaci*et al.* [5] proposent un algorithme génétique multi-objectifs (MO-GA) dont le but est de trouver le meilleur ordonnancement pour des applications à travers le temps. La fonction objectif de cet algorithme est la réduction de la consommation d'énergie et des émissions de CO_2, tout en maximisant le profit et en respectant la QoS des délais de terminaisons des applications (*deadline*). Les centres de calcul sont répartis sur des sites géographiques différents. L'ordonnan-

ceur utilise un cycle de planification décrit dans la figure 2.5. À chaque cycle, l'ordonnanceur attend d'avoir suffisamment d'applications à planifier, puis le MO-GA permet d'identifier les meilleurs ordonnancements possibles sur les centres de calcul. Les résultats sont stockés dans une archive Pareto. Un ordonnancement est choisi dans l'ensemble des solutions Pareto. Cet état proposé par l'ordonnancement, servira de base à l'itération suivante. Leur approche est basée sur le modèle de IaaS. Leurs expérimentations ont été réalisées sur des traces existantes (Parallel Workload archive [47]). La période des traces s'étend sur 6 mois (janvier à juin 2007). Les informations extraites de ces traces correspondent aux caractéristiques de leur application : temps de soumission des applications, leur temps d'exécution et le nombre de processeurs requis. Les traces étant dépourvues d'informations sur le délai de terminaison des applications, ils ont généré cette information. Le nombre de centres de calcul est porté à huit, répartis dans différentes zones géographiques.

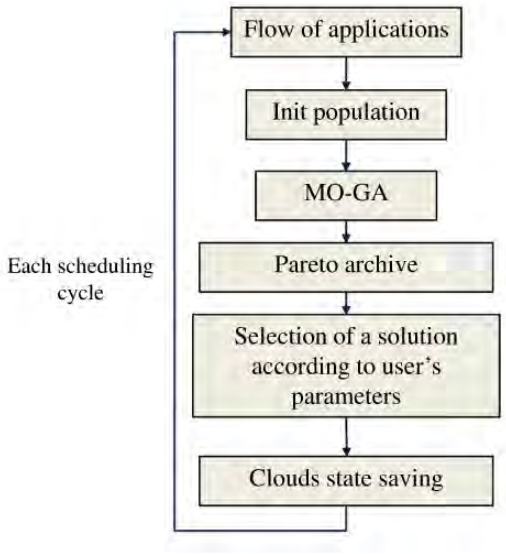


FIGURE 2.5 – Représentation picturale de l'algorithme de (Kessaci*et al.* [5])

Guérout *et al.* [4, 48] proposent un algorithme génétique pour placer des machines virtuelles sur un ensemble de serveur. La fonction objective de cet algorithme est la minimisation de la consommation d'énergie, du temps de réponse, et de la robustesse tout en maximisant le dynamisme. La robustesse est un indicateur de QoS évaluant la défaillance de machine physique (panne logicielle, panne physique, panne réseau, ...). Le dynamisme permet de connaître la capacité de CPU libre sur chacune des machines physiques. Cependant, cet algorithme ne prend pas en compte le type d'énergie utilisé et la production d'énergie réalisée. De plus, l'ensemble des VMs est programmé dès le début de l'algorithme.

Lei *et al.* [49] présente un algorithme évolutionnaire à plusieurs objectifs pour ordonnancer des tâches sur un centre de calcul partiellement alimenté par de l'énergie renouvelable. Basé sur des prédictions d'énergie renouvelable, l'algorithme ordonnance une charge de travail (ensemble de tâches) sur un centre de calcul. L'ordonnanceur sélectionne les nœuds informatiques et leur besoin en voltage, pour diminuer la consommation et correspondre à la production d'énergie verte prévisionnelle. Les objectifs sont les suivants : maximiser l'utilisation de l'énergie verte, et le taux de satisfaction des tâches, et simultanément, minimiser la durée d'exécution d'une tâche et la consommation d'énergie totale.

2.2.4 Positionnement et objectifs

L'objectif de cette thèse n'est pas nécessairement la définition d'un algorithme de placement optimal, mais plutôt d'améliorer les moyens permettant de le faire. À travers des validations et des illustrations de ces optimisations, nous allons concevoir un ordonnancement visant à réduire la consommation énergétique non renouvelable. Nos contraintes sont liées aux énergies vertes car elle ne sont pas pérennes dans le temps. Pour cela nous devons obtenir un résultat rapidement afin de trouver un ordonnancement favorable à la minimisation d'énergie non renouvelable. De plus, le nombre de contraintes peut être élevé dans un réseau intelligent possédant un certain nombre d'éléments pouvant fournir de l'énergie. Dans ce manuscrit, nous avons pris la position sur les trois points suivants :

Le type de prise de décisions. Nous avons choisi de prendre des décisions de manière hors ligne et non dynamique car les énergies renouvelables ont un rendement qui dépend principalement de la météo. Or de nombreux travaux se portent sur les prévisions météorologiques. Cependant, dans un espace temps d'une grandeur supérieure au mois, les prévisions ne sont plus précises. L'ordonnancement de charge de travail peut se faire à travers des algorithmes de décision en ligne ou hors ligne. Le centre de calcul peut se situer sur le même site où la production d'énergie renouvelable est réalisée. Tu *et al.* [50] ont montré que l'approche en ligne obtient de meilleurs résultats. Cependant, l'approche hors ligne obtient des résultats d'optimisation de la ressource énergétique non négligeables. C'est pour cela que nous nous sommes d'abord intéressés à des décisions hors ligne, dont les algorithmes sont plus rapidement paramétrables et obtiennent un ensemble de résultats de manière plus rapide.

La famille d'algorithme utilisée. Nous avons préféré l'utilisation d'algorithmes Gloutons. De fait, la programmation linéaire peut être utilisée pour l'optimisation des ressources énergétiques [51, 52, 53], et peut être très performantes pour tendre vers des solutions exactes. Cependant, la complexité en termes de contraintes (temps, différences sources d'énergie, contraintes des machines physiques et des machines virtuelles, ...) est élevée, et le temps de

calcul n'est pas borné et peut-être suivant la complexité du programme prendre du temps en termes d'exécution et de calcul. Les algorithmes gloutons nous ont permis de mettre en place un ensemble de décisions. Ces algorithmes ont mis en avant des contraintes sur des flux hétérogènes pouvant circuler dans un réseau intelligent. De plus, cela nous a aidé à étudier, avec des expérimentations, le comportement d'une charge de travail agile, alimentée par des sources renouvelables ou non et des espaces de stockage.

Le type de charge de travail. Nous avons opté pour l'utilisation d'une charge de travail réelle pour appuyer la réalité des résultats de consommation. Les caractéristiques des charges de travail sont étudiées dans ce manuscrit ainsi que la génération de traces synthétiques comme Kessaci *et al.* [5], via Parallel Workload archive [47] et/ou en utilisant des priorités pour les différentes tâches comme dans Lei *et al.* [12]. L'ordonnancement se fera en fonction de la consommation énergétique et les ressources demandées par une charge de travail et non à travers des nœuds d'un réseau de centre de calcul. Le choix s'est porté vers l'utilisation de machines virtuelles car elles peuvent contenir elles même un ensemble de services. De plus, elles sont fortement présentes dans la littérature sur les travaux de migrations [54] et leurs utilisations pour la sauvegarde d'énergie [55, 56, 57]. Les machines virtuelles possèdent un taux d'abstraction élevé sur le type de services demandé, qui n'est pas une caractéristique des travaux de cette thèse.

2.3 Les simulateurs de gestion de centres de calcul

Les simulateurs sont des outils de développement utilisés afin de tester et évaluer de manière rapide et efficace des algorithmes de placement. Cette section présente les simulateurs reproduisant un environnement électrique pour un centre de calcul et pour les placements de charge de travail dans un contexte d'applications distribuées.

2.3.1 Les simulateurs d'environnement distribué

Simgrid [58] est un simulateur pour les applications distribuées hétérogènes dans un environnement distribué. Il autorise l'exécution d'algorithmes d'ordonnancement d'un centre de calcul [59], la simulation de stockage dans un centre de calcul [60] et l'utilisation du DVFS dans un centre de calcul [61]. Ce simulateur s'inscrit surtout dans l'amélioration d'applications réelles dans le domaine des grilles.

CloudSim[62], historiquement basé sur GridSim [63], est un simulateur de *Cloud Computing* de type IaaS (*Infrastructure as a Service*), implémentée en Java. L'architecture des centres de calcul est finement modélisée et facilement modifiable par l'utilisateur. Il intègre la virtualisation

et la migration, la simulation réseau inter-centre de calcul, l'ajout de politiques de placements ainsi que les coûts énergétiques de fonctionnement induits. Coté énergie, CloudSim permet l'extinction de machine, la migration de machines virtuelles et l'intégration de modèles énergétiques. Une autre version [39] intègre également le DVFS, avec un fonctionnement identique à celui implémenté dans le noyau Linux, favorisant les études énergétiques avec une granularité de gestion très fine des fréquences CPU. Ce simulateur est purement dédié à la gestion interne des centres de calcul.

2.3.2 Les simulateurs dédiés aux réseaux intelligent pour les centres de calcul verts

GreenSolt [64] est un simulateur et un planificateur de tâches pour des centres de calcul alimentés par des panneaux photovoltaïques, mais aussi par le réseau d'électricité. GreenSlot peut prédire la quantité d'énergie solaire qui sera disponible dans un avenir proche. GreenHadoop [65] est une évolution de GreenSolt [64]. C'est un simulateur d'éoliennes et de panneaux solaires. GreenHadoop intègre le principe de réplication dynamique de données afin de garantir la disponibilité de celles-ci, lorsque les serveurs sont éteints. GreenHadoop a un module de prédiction de production d'énergie avec lequel la charge des batteries est gérée. Contrairement à GreenSlot [64] les tâches ont cinq niveaux de priorités (très élevée, élevée, normale, faible, et très faible). Trois états peuvent être affectés aux serveurs : actif, veille et éteint. Ces deux simulateurs [64, 65] permettent d'optimiser la consommation de l'énergie solaire mais ne proposent pas d'optimiser simultanément la consommation d'énergie, et le coût financier.

ReRack [66] est une infrastructure de simulation extensible. Il est composé en deux parties : un simulateur et un optimiseur. Ce simulateur a pour but d'alimenter un centre de calcul monosite via trois sources d'énergies : solaire, vent, et réseau d'électricité. De plus, il utilise un espace de stockage pour réduire les coûts de fonctionnement du centre de calcul. Son optimiseur utilise un GA visant à minimiser les coûts et obtenir un meilleur taux d'utilisation des énergies renouvelables.

DCWorms (Data Center Workload and Resource Management Simulator) [6] est un simulateur évaluant l'impact énergétique via l'utilisation de plusieurs ressources hétérogènes dans un environnement de type nuage. L'architecture générale de DCWorms est décrite par la figure 2.6. Il est basé sur le Simulateur GSSIM [67, 68]. Il propose des fonctionnalités liées à la consommation énergétique. Son objectif principal est de modéliser des infrastructures des centres de calcul et d'estimer leurs performances énergétiques via différents types de charges et de politiques d'ordonnancements. Ce simulateur permet de récupérer les expériences réalisées et d'obtenir des rendus graphiques via un module statistique. Le simulateur autorise l'ajout de plugin par les utilisateurs.

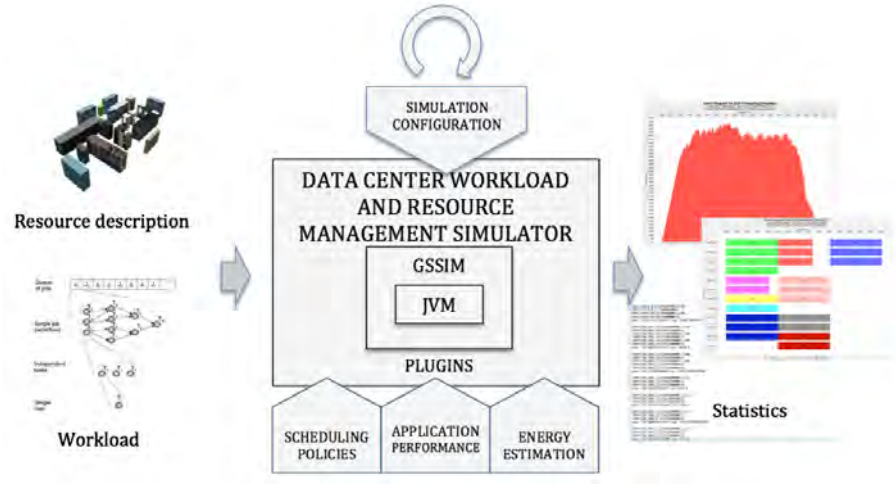


FIGURE 2.6 – Architecture de DCWorms [6]

2.3.3 Positionnement et objectifs

Dans cette thèse, nous voulons un simulateur qui répondent aux objectifs suivants :

Une meilleure utilisation de l'énergie renouvelable et ne pas seulement réduire l'aspect économique. Il existe aussi d'autres simulateurs servant à réaliser des études énergétiques tant sur la consommation et sur l'utilisation d'énergie verte dans des centres de calcul comme [69, 70, 71]. Certains simulateurs comme GreenWare [24] maximisent l'utilisation d'énergie renouvelable et évalue le coût pouvant être généré par l'utilisation d'énergie verte. Ces simulateurs se placent principalement du côté économique, pour la réduction des coûts des opérateurs de services Internet. Les objectifs de nos travaux doivent porter sur la réduction de la consommation énergétique non renouvelable, afin d'obtenir un bâtiment pouvant être à la fois producteur d'énergie et consommateur.

Un simulateur pour la co-simulation. Le simulateur DCWorms [6] autorise l'ajout de *plugins* par les utilisateurs. Cependant, une fois les *plugins* implémentés et utilisés dans le simulateur, ils ne sont plus utilisables, tels quels, dans d'autres environnements et ont un fonctionnement qui est propre à DCworms. L'objectif est de proposé un simulateur ayant une architecture polymorphe permettant la co-simulation. C'est-à-dire, l'utilisation de divers composants logiciels qui peuvent être déconnectés et re-connectés de façon simple (*plug and play*).

La communication entre le réseau physique électrique et la demande du centre de calcul. Les différents simulateurs décrits précédemment, ne prennent pas en compte le type de réseaux (courant continu ou courant alternatif). Certains travaux [72, 73] montrent que l'utilisation d'un réseau en courant continu peut réduire les pertes d'énergies liées à leur transport ou à leur conversion d'un courant à un autre. Les objectifs ne portent pas sur l'utilisation d'un réseau intelligent en courant continu (DC/DC) ou alternatif (DC/AC). Cependant, le simulateur doit agréer l'une ou l'autre situation et voir même l'utilisation des différents types de courants dans un même réseau intelligent. Il doit permettre la prise de décision en fonction des flux d'énergie mais aussi de l'état et des demandes du centre de calcul via un module de contrôle centralisé.

2.4 Conclusion

Ce chapitre propose un état de l'art de différents domaines tant sur l'électronique que sur la gestion de centre de calcul en terme de consommation et de gestion de ressources (ordonnancement). Le but est donc de créer un outil permettant la communication entre les différents domaines présents dans un réseau intelligent afin de baisser la consommation énergétique d'un centre de calcul.

Création d'un simulateur

Pour cela, nous avons créé un simulateur **RenewSim**, dédié à l'utilisation de sources d'énergie hétérogènes : renouvelables, non renouvelables et espaces de stockage. Notre approche, via RenewSim, propose de s'intéresser simultanément à deux problématiques de recherche : l'optimisation du flux énergétique et l'optimisation de l'ordonnanceur d'un centre de calcul dans un réseau alimenté par des sources hétérogènes.

Comme dans GreenCloud [2] ou DCWorms [6], notre simulateur doit permettre de connaître en temps réel ou non les résultats d'une simulation à travers un ensemble de mesures :

- Utilisation des ressources des machines physiques ;
- L'état des espaces de stockage au cours du temps (charge/décharge) ;
- L'évolution de l'achat/vente à un fournisseur d'électricité ;
- Le pourcentage d'utilisation d'énergie verte ou non.

Il doit offrir la possibilité de :

- mettre en place d'algorithme d'ordonnancement (cf 2.2) ;
- utiliser une architecture polymorphe et non fixe (cf 2.1). C'est-à-dire, pouvoir brancher et débrancher facilement d'autres simulateurs, outils, ou modèles sans influencer son fonctionnement ;
- offrir un environnement de simulation et d'expérimentations.

Analyse d'une charge de travail réelle

Ensuite, dans ce manuscrit nous présentons une analyse de charge de travail à grande échelle [8] afin de comprendre le fonctionnement de telles charges et générer un ensemble de traces synthétiques, pour pouvoir les implémenter dans un centre de calcul. Les traces analysées nous permettent d'avoir un aperçu de la QoS, dont le domaine de recherche n'est pas approfondi dans ces travaux de thèse.

Création d'un ordonnanceur

Nous proposons un ordonnanceur directement utilisable dans le simulateur RenewSim à travers l'utilisation et l'adaptation d'un ensemble d'algorithmes de la famille des gloutons. Ces algorithmes ont pour but de placer la charge de travail synthétique, basée sur une analyse réelle, pour générer une charge de travail agile.

Le Chapitre 3 suivant présente notre simulateur RenewSim. Son architecture et son système de prise de décision y sont décrits.

Chapitre 3

Environnement de simulation pour la gestion des flux dans un réseau intelligent

Sommaire

Ces travaux de thèse s'inscrivent en vue de la création d'un centre de calcul pour un bâtiment scientifique dans le cadre de l'opération neOCampus (cf. 1.1.2). La problématique d'un réseau intelligent connecté à une source d'énergie verte est étudiée. La production non régulière des énergies renouvelables introduit des problématiques d'alimentation, car il faut satisfaire la demande variable des centres de calcul. Dans le chapitre précédent, nous avons porté un état de l'art et une analyse sur l'architecture de centres de calcul et sur des simulateurs. Dans ce chapitre, l'approche présentée est la conception d'un réseau intelligent contrôlé par un module central, permettant de prendre des décisions pour satisfaire la demande des centres de calcul. Cette approche doit permettre la création d'un simulateur de co-simulation pouvant rejouer la gestion électrique d'un centre de calcul alimenté via des énergies renouvelables.

La structure de ce chapitre est la suivante : tout d'abord, l'approche de conception d'un réseau intelligent et de ses objectifs sont présentés. Ensuite, les spécifications d'une approche par simulation pour ce type de réseau sont analysées. Puis, la modélisation du réseau intelligent et ses différentes caractéristiques sont introduites. Dans une quatrième partie, le module de contrôle du réseau intelligent et son système de décision sont modélisés. Enfin, dans une dernière partie, le bilan et les limites d'un tel réseau sont exposés.

3.1 Approche de conception d'un réseau intelligent

Dans cette partie 3.1 sont présentées les caractéristiques d'un réseau intelligent afin de le modéliser, et de permettre la création d'un simulateur. Ce dernier doit reproduire le comportement d'un réseau intelligent contrôlé. Dans un premier temps, l'approche consiste à décrire un réseau intelligent afin de donner une définition. Dans un deuxième temps, cela permet de comprendre le fonctionnement et la modélisation d'un tel réseau, pour la création d'un simulateur.

3.1.1 Impact d'un réseau intelligent centralisé

Depuis quelques années, la modernisation des réseaux d'électricité a permis d'amener une flexibilité pour s'adapter à l'offre et à la demande d'énergie. Cela assure les trajets des différents flux d'électricité de manières efficace, pérenne et économique. Les contraintes liées à l'intermittence de production des énergies renouvelable, telles que les énergies éoliennes et solaires, sont plus facilement intégrées dans les réseaux intelligents. Les flux circulants dans un réseau intelligent sont hétérogènes, en supplément des flux d'électricité, le réseau intelligent est capable de faire remonter un certain nombre d'informations sur les éléments qui le composent afin de prendre une décision adaptée et de se contrôler de manière indépendante et automatique.

La figure 3.1 est un exemple d'un réseau intelligent [7]. Ce réseau est constitué d'un ensemble de parcs de production avec des énergies renouvelables (éoliennes, panneaux solaires, ...) ou non (central nucléaire, turbine à gaz). De plus, il possède un réseau de transport pour alimenter les consommateurs. Ce schéma justifie la présence de différents flux dans un tel réseau :

- des **flux d'électricité** qui peuvent être ensuite consommés (*Consommation d'électricité*) et réinjectés dans le réseau (*injection d'électricité*) ;
- des **flux de communication**, ils sont représentés dans le schéma via les *Boîtiers de communication*. Ils envoient des informations partout dans le réseau.

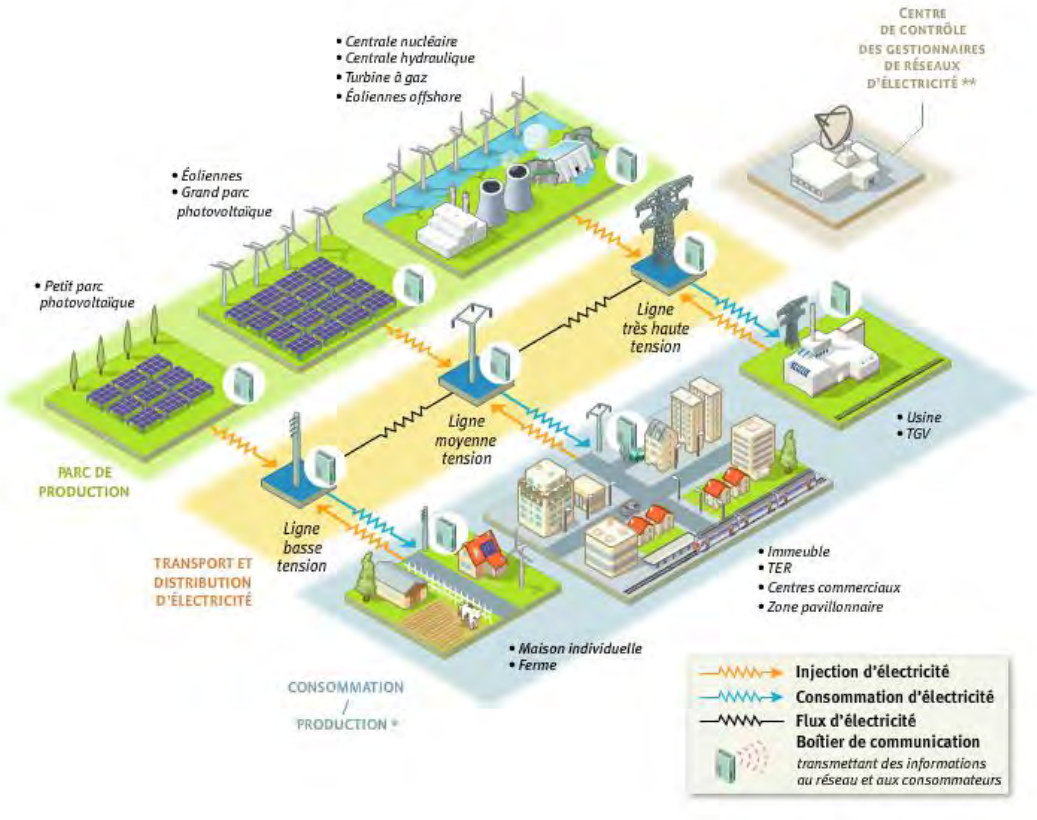


FIGURE 3.1 – Exemple de la composition d'un réseau intelligent [7]

3.1.2 Définition d'un réseau intelligent

Un ensemble de définition est présenté dans la littérature. Elles permettent de cibler les caractéristiques et les objectifs d'un réseau intelligent. Ci-dessous sont données trois définitions.

Définition 1 : Commission de la Régulation Énergétique [7]

≪ Les réseaux électriques intelligents sont aussi appelés *Smart Grids*. Ce sont les réseaux électriques publics auxquels sont ajoutés des fonctionnalités issues des nouvelles technologies de l'information et de la communication (NTIC). Le but est d'assurer l'équilibre entre l'offre et la demande d'électricité à tout instant et de fournir un approvisionnement sûr, durable et compétitif aux consommateurs. ≫

Définition 2 : Union Européenne [74]

≪ Les réseaux intelligents sont des réseaux d'énergie pouvant surveiller automatiquement les flux d'énergie et s'adapter aux changements d'approvisionnement en énergie et de demande. Lorsqu'il est couplé à des systèmes de mesure intelligents, les réseaux intelligents atteignent les consommateurs et les fournisseurs en fournissant des informations sur la consommation en temps réel. Avec les compteurs intelligents, les consommateurs peuvent s'adapter - en temps et en volume - à leur consommation d'énergie à différents prix de l'énergie tout au long de la journée, en économisant de l'argent sur leurs factures d'énergie en consommant plus d'énergie dans des périodes de prix plus bas. ≫

Définition 3 : Le forum du réseau intelligent d'Ontario [75]

≪ Un réseau intelligent est un système électrique moderne. Il utilise des capteurs, des contrôleurs, des outils de communications, de l'automatisation et des ordinateurs pour améliorer la flexibilité, la sécurité, la fiabilité, l'efficacité et la sécurité du système électrique. Cela augmente le choix du consommateur en permettant aux consommateurs de mieux contrôler leur consommation d'électricité en réponse aux prix ou à d'autres paramètres. ≫

3.1.3 Caractéristiques d'un réseau intelligent

À partir des définitions présentées dans la partie 3.1.2, le réseau intelligent possède un ensemble de caractéristiques :

- les capteurs qui le composent sont reliés entre eux ;
- c'est un système auto adaptatif, c'est-à-dire, qu'il s'évalue et est capable de prendre une décision de manière automatique ;
- il y a une régulation de la tension et courant entre les différents éléments qui le composent ;
- la capacité de s'équilibrer entre l'offre et la demande, c'est-à-dire, être capable de fournir de l'énergie afin de satisfaire les utilisateurs.

3.1.4 Objectifs d'un réseau intelligent

À partir des définitions présentées dans la partie 3.1.2, les objectifs d'un réseau intelligent sont multiples :

- **énergétique** : minimiser les pertes d'énergie dans le réseau, ou réduire la consommation électrique ;
- **économique** : diminuer la facture électricité et les coûts de fonctionnement ;
- **sécurité** : garantir la sécurité des informations circulant dans le réseau intelligent ;
- **lucratif** : revendre l'énergie renouvelable non utilisée ;
- **communication** : optimiser la communication entre les différents capteurs le composant ;
- **QoS** : assurer une QoS aux utilisateurs du réseau ;
- **environnemental** : diminuer l'émission de CO_2.

3.2 Spécifications d'un simulateur permettant la gestion des différents flux évoluant dans un réseau intelligent

Pour contrôler un réseau intelligent, il faut nécessairement identifier les éléments et les flux qui sont contrôlables. Dans cette partie 3.2, il est présenté les différentes motivations de l'utilisation d'un tel réseau dans ces travaux de thèse, et plus particulièrement, l'objectif de la création d'un simulateur pour la gestion des flux dans un réseau intelligent. Ce réseau n'alimente pas un ensemble d'habitations mais un centre de calcul via un ensemble de sources énergétiques hétérogènes dont des énergies renouvelables.

3.2.1 Un réseau intelligent supervisé par un module de contrôle

La figure 3.2 présente un exemple de réseau intelligent. Tous les éléments du réseau sont reliés à un bus. Un bus est un ensemble de liaisons physiques permettant de relier plusieurs appareils ou dispositifs entres eux. Il est destiné à la distribution de la puissance entre les différents composants.

Il est possible de brancher plusieurs sources et destinations. La source principale est la production réalisée par des énergies renouvelables. La destination principale est l'alimentation du centre de calcul. Chaque élément connecté au bus est relié à un système de conversion. Ce dernier est un module permettant la conversion de courant de type DC/DC ou DC/AC. Certains travaux portent sur l'optimisation d'un réseau de type DC/DC alimenté par des panneaux solaires pour stocker de l'énergie dans des batteries [72]. Cependant, l'optimisation des systèmes de conversions n'est pas évoquée dans ce manuscrit. Ce réseau intelligent est supervisé par un module de décision : le contrôleur. Son rôle est de trouver la meilleure configuration possible pour réduire le coût énergétique à travers la redirection et l'utilisation d'énergie ou à travers l'ordonnancement

du centre de calcul.

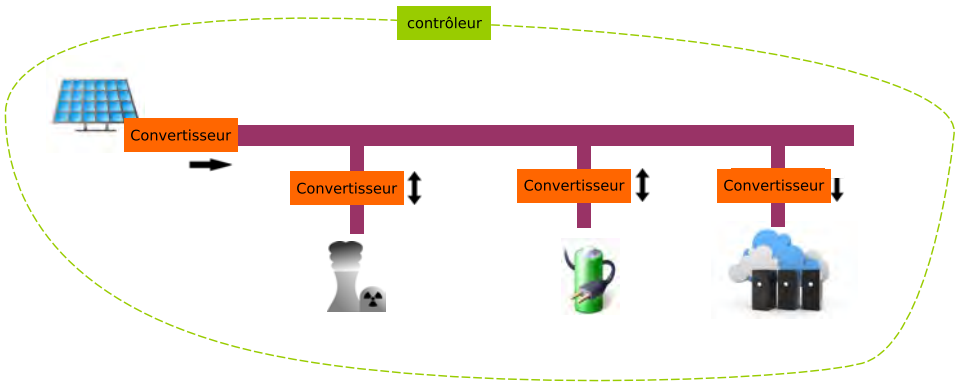


FIGURE 3.2 – Exemple d'architecture d'un réseau intelligent d'un réseau intelligent permettant d'alimenter un centre de calcul

Le défi d'un module de contrôle est :

- de relier les différents flux d'énergies, renouvelables ou non renouvelables ;
- de satisfaire la demande d'énergie du centre de calcul ;
- d'assurer la QoS aux utilisateurs.

Les décisions prises par le module de contrôle sont exécutées à deux niveaux :

- **Énergie** : gestion de la production, stockage et acheminement d'énergie ;
- **Informatique** : gestion des ressources du centre de calcul et son optimisation.

3.2.2 Les motivations de la création d'un simulateur

L'utilisation de différentes sources d'énergies introduit de nouvelles problématiques liées au stockage de l'énergie. De nombreux travaux existent sur les modèles de charge/décharge des batteries et leur utilisation dans un centre de calcul [13, 14]. Cette forme de stockage peut être considérée comme un cas de sécurité contre la pénurie. De plus, l'utilisation d'énergie renouvelable est complexe et imprévisible car elle dépend des variations climatiques. L'optimisation de la distribution des services dans un système informatique est complexe. Un centre de calcul est composé de machines hétérogènes et de services ayant des caractéristiques différentes. Par exemple, la puissance demandée pour un service diffère d'un service à l'autre et les ressources disponibles dans le système informatique évoluant dans le temps.

Comment corréler l'optimisation des flux énergétiques à l'optimisation de la distribution des services dans un système informatique ?

Le module de contrôle dans le réseau intelligent doit centraliser toutes les informations des différents composants capteurs et contrôler les flux de puissance à travers le réseau. Cela per-

met d'utiliser au mieux l'énergie verte et de contrôler son flux dans le réseau intelligent pour économiser de l'énergie ou réaliser moins de perte. Il est tenu aussi optimiser les ressources informatiques.

Le simulateur de ce réseau intelligent doit être un outil de supervision, permettant de concevoir un réseau polymorphe dont l'ensemble des éléments sont paramétrables. Il doit pouvoir gérer un ensemble de traces de centre de calcul à grande échelle (chapitre 4), et créer un environnement pour expérimenter (chapitre 6) un ensemble d'ordonnancement(chapitre 5).

3.2.3 Fonctionnalités attendues

Le simulateur a besoin de posséder un ensemble de fonctionnalités. Il doit :

- être de type greffon : c'est-à-dire, qu'il doit pouvoir communiquer avec d'autres simulateurs du domaine, par exemple, de l'énergie, du photovoltaïque ou de centre de calcul ;
- offrir la possibilité d'être extensible et polymorphe. Les éléments doivent avoir un niveau d'abstraction assez élevé, afin de rendre les possibilités d'extensibilité plus importantes ;
- obtenir un environnement d'expérimentation, afin de, valider différentes politiques d'ordonnancement, de placement ou d'utilisation de l'énergie collectée par des panneaux solaires ;
- créer un historique qui a pour but de modéliser graphiquement ou textuellement les données, afin de comparer un ensemble d'expérimentations entres elles.

3.3 Description et spécification des éléments d'un réseau intelligent

Dans cette partie 3.3.1 l'architecture matérielle d'un réseau intelligent. Les différents flux sont analysés, afin d'obtenir un module de contrôle permettant la prise de décisions.

3.3.1 Modélisation de l'architecture du réseau intelligent

Dans la partie 2.1 et la partie 2.3, un ensemble d'architecture pour gérer de tel type de réseaux est présenté. Plusieurs hypothèses de conception matérielles sont exposées pour se positionner sur une modélisation. L'ordonnanceur occupe une place importante dans la consommation d'énergie, car le résultat de son placement indique la dépense énergétique du centre de calcul. Il doit avoir accès à des informations de consommation et de puissance pour prendre une décision. L'architecture de GreenCloud [2] permet la communication des informations de mesures d'électricité à son système de planification. Il ne peut prendre des décisions sans ses informations. GreenSwitch [1] utilise un module de prédiction pour réaliser son ordonnancement. Il existe donc un

lien entre la prise de décision informatique et d'électricité. La plupart des travaux présentant des problématiques de gestion de centres de calcul avec plusieurs sources d'énergie, ont mis à profit les espaces de stockage d'énergie [1, 34]. Cependant, l'emploi de tels composants engendre des problèmes d'utilisation entre les autres sources énergétiques possibles. Pour résoudre ces conflits, dans la littérature, ces espaces de stockage sont employés seulement s'il n'y a plus d'énergie renouvelable [34, 23].

Nous proposons donc l'architecture proposée dans la figure 3.3 présentant les différents modules du réseau intelligent. Elle est composée :

- d'un bus électrique ;
- d'un ensemble d'éléments producteurs ou consommateurs reliés directement au bus via un convertisseur ;
- d'un module de contrôle central relié à un module d'ordonnancement et de contrôle de bus.

Cette architecture permet la connexion de n'importe quels éléments. Par exemple, dans la figure 3.3, le bus est composé d'un centre de calcul, d'une batterie, de panneaux solaires, de fournisseurs d'électricité et d'éoliennes. Le module de contrôle centrale peut prendre, par exemple, les décisions suivantes : la batterie peut être chargée ou déchargée, les éoliennes et les panneaux solaires produisent de l'énergie, les fournisseurs sont utilisés pour acheter ou vendre de l'énergie, et enfin, le centre de calcul reçoit de l'énergie.

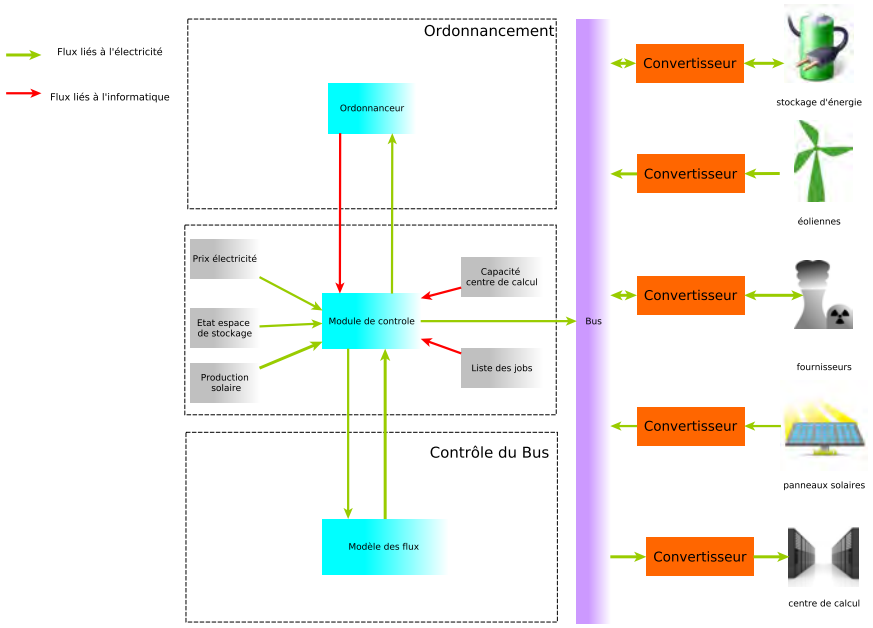


FIGURE 3.3 – Architecture du réseau intelligent et de son centre de calcul

Les parties suivantes décrivent en détails le fonctionnement de l'architecture, les définitions des différents flux et le rôle de chacun des modules.

3.3.2 Le centre de calcul

Un centre de calcul a besoin d'être alimenté 24h/24 et 7j/7, mais il doit satisfaire immédiatement la demande des utilisateurs avec un taux satisfaisant de QoS malgré la production incertaine. De manière générale, la QoS dans un centre de calcul est la capacité à fournir un service conforme à des exigences, par exemple, le temps d'attente des services, ou la capacité à s'adapter à des anomalies.

Un centre de calcul peut recevoir des demandes de services d'autres centres de calcul ou directement par des utilisateurs. Ces demandes de services peuvent être virtualisées ou englobées dans des conteneurs. Un conteneur n'intègre pas de système d'exploitation, il utilise le système d'exploitation du serveur sur lequel il est déployé. Cependant, l'avantage d'une VM à un conteneur est la sécurité car les conteneurs peuvent avoir accès au système d'exploitation sur lequel il est hébergé. Une VM est un composant logiciel, permettant de virtualiser les ressources d'une machine physique. Les machines physiques ont des capacités de puissance électrique et de ressources. Elles permettent l'exécution des différentes VMs. Il existe plusieurs façons de calculer la consommation d'un serveur, par exemple en utilisant les puissances minimum et maximum du serveur [76], ou estimant la consommation d'une VM [77]. L'équation 3.3.1 ([76]) permet d'estimer la consommation d'un serveur à un instant t, avec $P(t)$ la puissance résultante, P_{min} et P_{max} respectivement les puissances minimum et maximum d'un serveur et $L(t)$ la charge d'un serveur à un instant t.

$$p(t) = p_{min} + (p_{max} - p_{min}) \times l(t) \tag{3.3.1}$$

La figure 3.4 présente le centre de calcul de notre réseau intelligent. Il possède un ou plusieurs serveurs, permettant l'exécution de VMs.

Définition 3.1. *Les demandeurs de services*

Le centre de calcul peut recevoir des demandes de services d'autres centres de calcul ou directement par des utilisateurs. Ces demandes de services se traduisent par une ou plusieurs VMs.

Définition 3.2. *Les Machines Virtuelles (VMs)*

Une VM doit posséder un ensemble d'information : un identifiant unique, une durée d'exécution, et sa demande de ressources. Ces informations permettent par la suite de calculer la consommation sur les serveurs.

Définition 3.3. *Les machines physiques : Serveurs*

Les machines physiques ont des capacités de puissance (minimum et maximum), une durée d'exécution et des ressources vCPU. Ces ressources sont des unités CPU physiques qui sont affectées à des VMs. Un serveur, suivant ses ressources vCPU peut exécuter une ou plusieurs VMs à la fois.

Définition 3.4. *QoS*

Dans ce manuscrit, cet indicateur permet de juger la conformité du centre de calcul à allouer un nombre de ressources suffisant aux VMs.

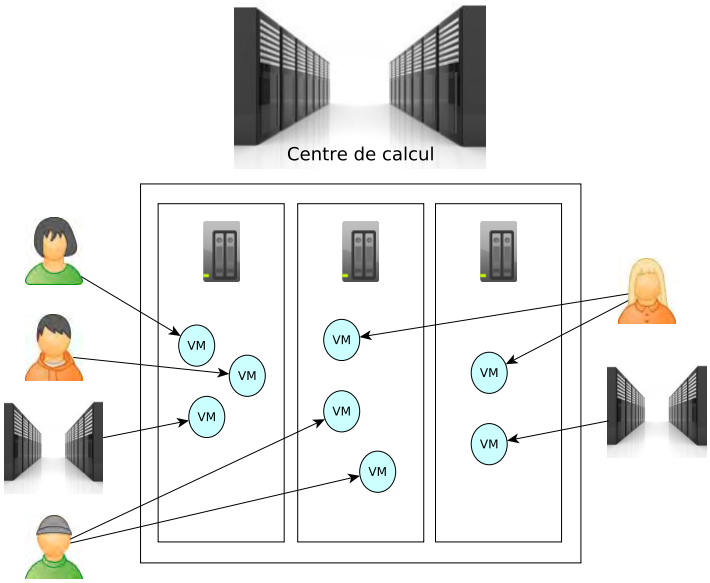


FIGURE 3.4 – Composition du centre de calcul du réseau intelligent

3.3.3 Les sources d'énergies

Une source d'énergie dans notre réseau intelligent peut être de trois types : non renouvelable, renouvelable, ou espace de stockage d'énergie.

Les énergies renouvelables et non renouvelables

Les énergies non renouvelables sont issues d'une production non respectueuse de l'environnement. Par exemple, elles proviennent du charbon, du pétrole, du gaz naturel ou du nucléaire. Les énergies renouvelables sont des énergies respectueuses de l'environnement, et dont les ressources se renouvellent naturellement et assez rapidement pour qu'elles soient non épuisables. Par

exemple, elles peuvent être solaires, éoliennes ou hydrauliques. Chaque dispositif de production possède des caractéristiques en terme de puissance. Par exemple, le réseau intelligent peut être constitué de panneaux solaires pour alimenter le centre de calcul. Un panneau solaire est un dispositif permettant de transformer l'énergie de la lumière en énergie électrique. Les contraintes des panneaux solaires sont les suivantes : ils produisent seulement durant la journée et seulement durant les jours ensoleillés, la production dépend du taux d'humidité et de leur chaleur. Par exemple, la puissance délivrée par les panneaux solaires est donnée par [78] dans l'équation 3.3.2 avec G_{eff} l'irradiance, μ_g le rendement du générateur et A_g la surface active du générateur.

$$P_{dc} = G_{eff}\mu_g A_g \tag{3.3.2}$$

Les énergies renouvelables peuvent être transportées vers le lieu où se situe le centre de calcul. Par exemple, cela peut permettre l'utilisation d'éoliennes dans un lieu où la nuisance sonore a un impact, comme un quartier résidentiel. Cependant, le transport d'électricité à un coût de fonctionnement et d'entretien

Définition 3.5. *Les énergies renouvelables*

Dans notre réseau intelligent, le lieu de production d'énergie renouvelable est situé au même endroit géographique que les autres éléments du réseau car le centre de calcul a pour vocation d'être auto-suffisant et de ne dépendre d'aucunes infrastructures physiques indépendantes. Ce sont des énergies qui doivent être respectueuses de l'environnement, et disposant elle même d'un modèle de rendement qui leur est propre. Le réseau intelligent doit permettre à l'énergie renouvelable d'utiliser son propre modèle de rendement, et/ou de production. Cependant, le réseau doit connaître impérativement les puissances supportées par de telles sources.

Définition 3.6. *Les énergies non renouvelables*

Dans ce réseau intelligent, le réseau d'électricité standard est représenté par le fournisseur d'électricité, car aucune information, ne nous permet pas de savoir à un instant donné d'où provient l'énergie. Cependant, le réseau intelligent doit connaître les coûts d'achats et de ventes, et les puissances du réseau d'électricité supportés.

Les espaces de stockage d'énergie

Les espaces de stockage représentent la sauvegarde d'énergie dans un élément. Cela permet d'utiliser cette énergie à un autre instant. Ces espaces de stockage peuvent être par exemple des batteries ou des super-condensateurs. Ils possèdent des caractéristiques en terme de puissance ou de temps de charge différentes. Par exemple, dans le réseau intelligent, il peut être utilisé une batterie. Son efficacité dépend de la façon dont elle est stockée. Elle a besoin d'une température

optimale, chaque modèle de batterie possède sa température efficiente. Les débits de charge et de décharge de la batterie sont limités. Par exemple, la dynamique d'une batterie suit le modèle suivant présentée par [79] dans l'équation 3.3.3 suivante avec $E_B(t)(Wh)$ la quantité d'électricité consommée dans la batterie à un instant t, $P_B(t)(W)$ le taux de charge/décharge.

$$\frac{dE_B(t)}{dt} = P_B(t) \tag{3.3.3}$$

Définition 3.7. *Espaces de stockage d'énergie*

Dans le réseau intelligent, le lieu où se situe l'espace de stockage est situé sur le même site géographique que les autres éléments du réseau. Un espace de stockage possède une capacité de stockage (en joules) et de puissance électrique maximale et minimale.

3.3.4 Définitions des différents flux circulant dans un réseau intelligent

Il y a deux types de flux qui circulent dans le réseau intelligent : les flux dit « énergétiques » et les flux « informatiques »

Définition 3.8. *Les flux « énergétiques »*

Les flux énergétiques sont la quantité d'énergie en fonction du temps circulant dans le réseau intelligent. Notre réseau doit, par exemple, prendre en compte les taux de charge et de décharge des éléments de stockage d'énergie, les coûts d'achats et de vente d'électricité, les pertes générées d'énergie d'un élément à un autre dans le centre de calcul.

Définition 3.9. *Les flux « informatiques »*

Les flux informatiques sont les informations relatives à la réalisation des différents VMs dans le centre de calcul. L'allocation des VMs dépend de leurs caractéristiques (priorité, préemptive ou non) mais aussi de la quantité totale d'énergie disponible.

3.3.5 Bilan de l'architecture matérielle du réseau intelligent pour la création d'un simulateur de co-simulation

Dans cette partie, nous avons décrit l'architecture d'un réseau intelligent polymorphe. Cette analyse permet la création d'un simulateur de co-simulation possédant un centre de calcul, des espaces des stockages d'énergie, et des ressources d'énergies renouvelables et non renouvelables. Cela nous permet, en partie, de répondre aux verrous édifiés par les fonctionnalités attendues décrites dans la partie 3.2.3.

Via ce type de réseau intelligent, notre simulateur peut :

- être de type greffon en ajoutant d'autres classes au module de contrôle (cf. figure 3.5). Cela n'impactera pas le fonctionnement des autres éléments ;
- être extensible. Le taux d'abstraction étant élevé, nous pouvons utiliser n'importe quelle source renouvelable ou différents types d'espaces de stockage d'énergie ;
- offrir, via cette représentation, la possibilité de créer un historique des données.

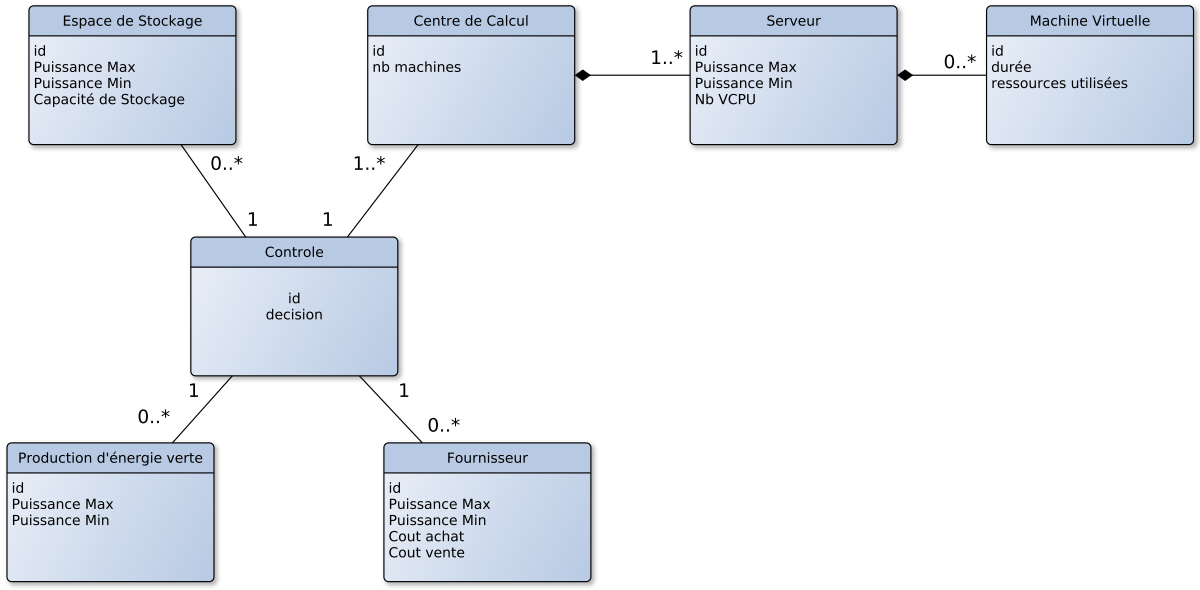


FIGURE 3.5 – Diagramme des classes UML des différents éléments du réseau intelligent

À travers cette première phase d'analyse, la figure 3.5 dresse le diagramme UML (Langage de modélisation graphique à base de pictogrammes) d'un tel simulateur. Cette approche permet de garder une abstraction des différents éléments du réseau afin de composer des simulations avec différents modèles ou simulateurs. De plus, cela permet d'ajouter des informations à un ensemble de classes sans impacter le reste du réseau et surtout le module de contrôle.

Les espaces de stockage possèdent un ensemble de caractéristiques différentes suivant la technologie utilisée. Par exemple, les batteries à plomb ou à lithium se chargeront plus ou moins rapidement. Pour cela le réseau intelligent doit pouvoir utiliser n'importe quelles technologies de stockage et permettre son utilisation avec les autres éléments du réseau. C'est pour cela que l'on trouve un module de conversion, *Convertisseurs*, afin d'uniformiser les différents éléments en terme de tension ou, de puissance sur le réseau intelligent.

Le centre de calcul est décrit à haut niveau pour permettre la modularité des éléments composants le réseau intelligent. Cela permet d'utiliser un ensemble de centres de calcul ayant des caractéristiques et des fonctionnalités différentes.

Les énergies renouvelables utilisées possèdent des rendements et des productions différents. Par exemple, les éoliennes produiront autant d'énergie le jour comme la nuit si le vent reste constant. Cependant, les panneaux solaires ne produiront pas la nuit mais le jour, si seulement si, les conditions météorologiques le permettent.
Les fournisseurs d'électricité ont un fonctionnement non modifiable par le système. Cependant, leurs caractéristiques doivent être connues pour la prise de décision.

Le module de contrôle doit permettre de gérer l'ensemble de ses données indépendamment de leur nature. Pour cela, le niveau d'abstraction de modélisation dans le réseau intelligent est haut, et permet au module de contrôle de récupérer toutes les informations du réseau afin de prendre une décision adéquate par rapport aux différentes caractéristiques des éléments qui le composent.

3.4 Le module de contrôle : la prise de décision

Cette section présente le processus de prise de décision dans le module de contrôle qui permet de gérer les deux types de flux : les flux informatiques et les flux énergétiques.

3.4.1 Le contrôle des flux informatiques : un ordonnanceur

Le contrôle des flux informatiques se fait via l'ordonnanceur. Ce dernier peut être de type de « en ligne » ou « hors ligne ». Le réseau intelligent accepte n'importe quel ordonnanceur. Ce dernier peut :

- choisir l'ordre d'exécution des différents services (VM) et de les allouer aux différentes ressources (Serveur) ;
- avoir accès aux informations suivantes : production solaire, charge des serveurs, et toutes les informations relatives au réseau intelligent (espace de stockage, fournisseur ...).

L'ordonnanceur peut donc prendre des décisions en fonctions des flux informatiques et énergétiques. Ceci fera l'objet de la contribution du chapitre 5.

3.4.2 Le contrôle des flux énergétiques : le contrôle du bus

Les différentes sources d'énergies et la demande du centre de calcul peuvent être divisées en deux catégories : les entrées pilotables et les entrées non pilotables.

Définition 3.10. *Les entrées non pilotables*

Ces entrées sont les sources d'énergies renouvelables et la charge de travail demandée par le centre de calcul. Par exemple, la production solaire ne dépend pas du système. La production des panneaux solaires dépend de la météo et de l'heure de la journée. De plus, le centre de calcul doit répondre à une demande qui ne dépend pas du réseau intelligent.

Définition 3.11. *Les entrées pilotables*

Ces entrées sont les espaces de stockage et le fournisseur d'électricité. Par exemple, le prix du fournisseur dépend du temps : pendant la nuit l'électricité est moins chère que pendant la journée.

Dans un premier temps, pour prendre une décision menant à une politique énergétique « verte », il faut donc gérer les différentes entrées pilotables ou spécifier le comportement intrinsèque de chaque bloc, décrits dans la partie 3.3.1 pour éviter des conflits d'utilisation entre chaque sources d'énergies. Pour prendre une décision plus rapide basée sur les entrées pilotables, chaque élément du réseau intelligent (cf. 3.2) est décrit par un modèle [80]. Le mode de caractérisation des éléments permet de définir quatre types d'éléments dans le réseau intelligent :

- Une **source** permet d'alimenter une **destination**.
- Une **destination** est alimentée par une **source**.
- Une **sauvegarde** permet d'alimenter la **destination** lorsque la **source** est insuffisante.
- Un **trop plein** permet de stocker la production de la **source** non nécessaire à la **destination**.

Afin de ne pas avoir de conflit entre plusieurs éléments de type **sauvegarde** et **trop plein**, ces derniers sont caractérisés par une priorité. La priorité égale à 1 est la plus élevée. À travers cette modélisation, il apparaît deux cas : la sous-production et la sur-production.

La sous-production

Lorsque la production des éléments de type **source**, est inférieure à la demande des **destinations**. Cette situation est présentée par l'équation 3.4.1, avec $\sum S_t$ est la somme de toutes les **sources** disponibles à l'instant t, et $\sum D_t$ est la somme de toutes les **destinations** disponibles à l'instant t.

$$\sum S_t < \sum D_t \tag{3.4.1}$$

L'élément de type **sauvegarde** va permettre d'équilibrer l'équation 3.4.1, et de permettre d'offrir aux éléments de **destinations** l'énergie désirée. Cela est représentée via l'équation 3.4.2, avec $\sum sauv_t$ la somme de toutes les **sauvegardes** à un instant t.

$$\sum S_t + \sum sauv_t = \sum D_t \tag{3.4.2}$$

Algorithme 1 Sous-production : $\sum S_t < \sum D_t$

```
for i=0;i;len(liste_element_sauvegarde);i++ do
  if liste_element_sauvegarde[i]_vide==Vrai then
    continue ;
  end if
  if liste_element_sauvegarde[i]_assez==Vrai then
    update_liste_element_sauvegarde
    break ;
  end if
  if liste_element_sauvegarde[i]_pas_assez==Vrai then
    update_liste_element_sauvegarde
    continue ;
  end if
end for
```

L'algorithme 1, représente le fonctionnement des éléments de types **sauvegarde**, lorsqu'il y a **sous-production**. Il y a trois comportements :

- l'élément **sauvegarde** est vide ;
- l'élément **sauvegarde** peut pallier entièrement à la sous-production ;
- l'élément **sauvegarde** ne peut pallier qu'à une partie de la sous-production ;

La sur-production

Lorsque la production des éléments de type **source**, est supérieure à la demande des **destinations**. Cette situation est présentée par l'équation 3.4.3, avec $\sum S_t$ est la somme de toutes les **sources** disponibles à l'instant t, et $\sum D_t$ est la somme de tous les **destinations** disponibles à l'instant t.

$$\sum S_t > \sum D_t \tag{3.4.3}$$

L'élément de type **trop plein** va permettre d'équilibrer l'équation 3.4.3, et de permettre de stocker le surplus d'énergie. Cela est représentée via l'équation 3.4.4, avec $\sum trop\_plein_t$ la somme de toutes les **trop pleins** à un instant t.

$$\sum S_t = \sum D_t + \sum trop\_plein_t \tag{3.4.4}$$

Algorithme 2 Sur-production : $\sum S_t > \sum D_t$

```
for i=0;i;len(liste_element_trop_plein);i++ do
  if liste_element_trop_plein[i]_plein==Vrai then
    continue ;
  end if
  if liste_element_trop_plein[i]_assez==Vrai then
    updateliste_element_trop_plein
    break ;
  end if
  if liste_element_trop_plein[i]_pas_assez==Vrai then
    updateliste_element_trop_plein
    continue ;
  end if
end for
```

L'algorithme 2, représente le fonctionnement des éléments de types **trop pleins**, lorsqu'il y a **sur-production**. Il y a trois comportements :

- l'élément **trop plein** stocke entièrement la sur production ;
- l'élément **trop plein** a atteint sa limite de stockage ;
- l'élément **trop plein** ne peut stocker qu'une partie de la sur production ;

Les éléments **destinations** et **sources** suivent l'équation 3.4.5. Pour garantir l'équation 3.4.5 en cas de sur-et sous-productions, l'équation 3.4.6 ajoute $\sum sauv_t$, la somme de toutes les ressources de type **sauvegardes** disponibles à l'instant t (cf. 3.4.2), et $\sum trop\_plein_t$ la somme des éléments de type **trop pleins** disponibles (cf. 3.4.4).

$$\sum S_t = \sum D_t \tag{3.4.5}$$

$$\sum S_t + \sum sauv_t = \sum D_t + \sum trop\_plein_t \tag{3.4.6}$$

Modélisation

La représentation de ce modèle peut se faire à travers une matrice. Les éléments du réseau sont représentés dans la colonne et leur type dans la ligne. La matrice 3.1 est le masque d'une tel représentation. Il peut y avoir n éléments, de stockages (*espaces de stockage 1 et 2*), d'énergies renouvelables (*énergies renouvelables 1 et 2*), ou de fournisseurs d'électricité (*fournisseur*), mais qu'un seul centre de calcul (*centre de calcul*).

Tableau 3.1 – Représentation en matrice des ensembles des valeurs possibles pour chaque élément du réseau intelligent

types / éléments	**Source**	**Destination**	**Sauvegarde**	**Trop-plein**
Énergie renouvelable 1	$\{\times, 1\}$	$\times$	$\{\times, 1, 2, ...\}$	$\times$
Énergie renouvelable 2	$\{\times, 1\}$	$\times$	$\{\times, 1, 2, ...\}$	$\times$
Centre de calcul	$\times$	$\{\times, 1\}$	$\times$	$\times$
Espace de stockage 1	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
Espace de stockage 2	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
Fournisseur	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
...	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
élément n	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$

Dans ce modèle de représentation des flux, les énergies renouvelables ne peuvent être destinataires, elles sont de type **source** ou de type **Sauvegarde**. Pour le centre de calcul, il ne peut être que de type **destination** car, il ne peut produire de l'énergie ou stocker de l'énergie. Les autres éléments, les espaces de stockage 1 et 2 et fournisseur, peuvent être de n'importe quels types, car ce sont des **entrées pilotables**. Cependant, il faut réussir à organiser les différentes priorités pour éviter les conflits entres eux. Il n'y a pas de priorités pour les **sources** et les **destinations**, car c'est la somme de celles-ci qui définit les cas de sur-production et sous-production.

Exemple de modélisation

Dans cette partie est présenté un exemple d'utilisation du modèle avec un réseau intelligent composé de panneaux solaires, d'une batterie, d'un fournisseur d'électricité et d'un centre de calcul.

Tableau 3.2 – Un exemple de modélisation - Politique de minimisation des coûts - matrice de jour

	Source	**Destination**	**Sauvegarde**	**Trop plein**
Panneaux solaires	1	0	0	0
Centre de calcul	0	1	0	0
Batterie	0	0	1	0
Fournisseur	0	0	2	1

TABLEAU 3.3 – Un exemple de modélisation - Politique de minimisation des coûts - matrice de nuit

	Source	**Destination**	**Sauvegarde**	**Trop plein**
Panneaux solaires	1	0	0	0
Centre de calcul	0	1	0	0
Batterie	0	1	0	0
Fournisseur	1	0	0	0

La modélisation dans les tableaux 3.2 et 3.3 minimise les coûts liés à l'électricité. **Pendant la journée**, le modèle est le tableau 3.2 : le réseau intelligent utilise les panneaux solaires pour alimenter le centre de calcul. S'il y a une sur-production d'énergie, l'excédant est vendu au fournisseur. S'il y a une sous-production, le manque est pris de la sauvegarde où la priorité est égale 1. Dans ce cas, c'est la batterie (décharge). Si la sauvegarde de priorité 1 est vide, alors le défaut est pris à la sauvegarde de priorité 2 (achat). Dans ce cas, c'est le fournisseur. **Pendant la nuit**, il n'y a plus d'élément de type sauvegarde et de trop plein (tableau 3.2). Le fournisseur est de type source avec les panneaux solaires afin de remplir l'espace de stockage qui est de type destination. Cela permet de remplir les éléments de stockage durant la nuit lorsque l'électricité est moins chère. Durant la journée, l'utilisation de la batterie sera favorisée au fournisseur d'électricité car celui-ci possède des tarifs d'achat plus élevés.

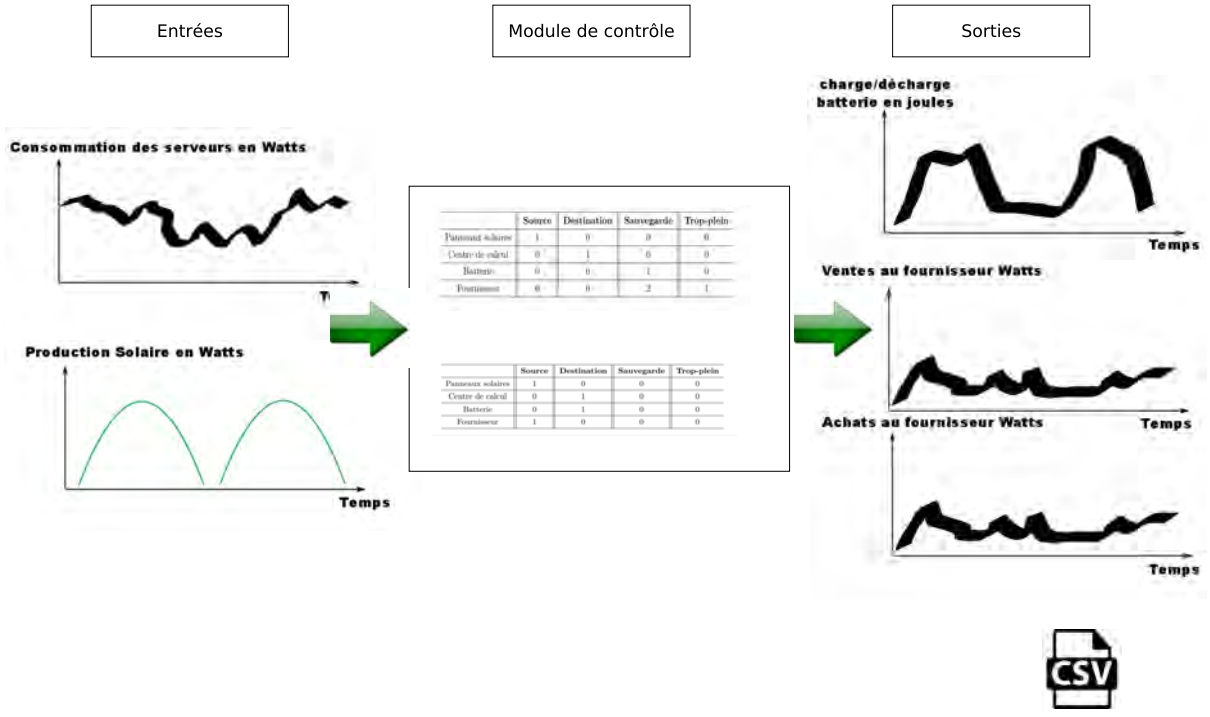


FIGURE 3.6 – Exemple de représentation du modèle de données en terme d'entrées et de sorties

La figure 3.6 représente les entrées et les sorties du modèle de décision à l'intérieur du module de contrôle :

- **Les entrées** sont la charge de travail du centre de calcul et la production solaire.
- **Les sorties** sont les décisions réalisées et leurs représentations graphiques et/ou textuelles (CSV). Cela permet de représenter et d'analyser le comportement des différents éléments du réseau en fonction d'un modèle défini. Dans la figure 3.6, les sorties représentées sont la vente et l'achat au fournisseur d'électricité en fonction du temps, puis, la charge et la décharge de la batterie.

3.4.3 Bilan de la prise de décision

Les deux types de flux informatiques et énergétiques, en plus des informations diverses délivrées par les capteurs, sont liés pour prendre une décision (figure 3.3). La demande du centre de calcul va correspondre à une charge d'électricité différentes en fonction de l'organisation en interne des VMs sur les différents serveurs composant le centre de calcul. **Le modèle** 3.1 autorise la représentation de chaque élément d'un réseau intelligent et améliore le module de contrôle. Le modèle, présenté dans la partie 3.4.2, permet une architecture polymorphe qui vise à prendre des décisions intelligentes et des politiques pour l'utilisation de sources d'énergies hétérogènes. Il aide à la redirection en étiquetant et priorisant les différentes sources d'énergie. Le module de contrôle peut prendre une décision dans un but et une politique sur une durée afin par exemple, d'utiliser plus d'énergie renouvelable, ou de diminuer la facture d'électricité. Ce modèle généralise tout le processus où la charge d'alimentation peut être déplacée. **L'ordonnanceur** anticipe la production d'électricité sur une ou plusieurs journées. Sachant que la production solaire dépend en partie de la notion temporelle, cette anticipation aide à placer une charge plus importante durant la journée. L'avantage est la maximisation de l'utilisation de la source énergétique verte.

3.5 Conclusion et limites

Ce chapitre présente la modélisation du réseau intelligent doté d'un module de contrôle. Il nous permet de répondre aux objectifs suivants :

- avoir un module de contrôle pour la communication entre les flux informatiques et énergétique ;
- offrir un niveau d'abstraction élevée pour autoriser la co-simulation ;
- obtenir un environnement d'expérimentation ;
- modéliser un réseau intelligent polymorphe.

La partie 3.3 décrit l'architecture matérielle. Cette architecture est présenté à haut niveau pour autoriser le branchement de plusieurs éléments de différents types. Il permet donc une architecture polymorphe offrant la possibilité d'établir de la co-simulation. Les définitions de l'architecture sont contenues dans cette partie de manière à exprimer comment chaque composant est représenté dans les chapitres suivants.

La partie 3.4 présente la prise de décision à travers un module de contrôle composé d'un ordonnanceur et d'un modèle de flux d'énergie. Cela autorise la communication entre les flux informatiques et énergétiques. Le modèle de flux présenté dans la section 3.4.2 prend en compte l'architecture polymorphe afin de prendre une décision intelligente pour l'utilisation de sources hétérogènes d'énergie.

À l'issue de ce chapitre, nous avons créé RenewSim qui est un simulateur d'environnement d'expérimentation et de co-simulation qui suit les spécifications énoncées précédemment. Le simulateur est codé en python et possède 6824 lignes de code. À travers RenewSim, il est possible de contrôler et de gérer la production d'énergie verte, afin d'alimenter un centre de calcul. Ce dernier reçoit des demandes de services dans le temps. Ce simulateur est un environnement d'expérimentation. Celui-ci peut modéliser graphiquement ou textuellement les données, afin de comparer un ensemble d'expérimentations entres elles. Le simulateur, RenewSim a permis d'utiliser des traces synthétiques du chapitre 4, de réaliser des simulations via des algorithmes adaptés à nos besoins (chapitre 5), de caractériser l'environnement, et de créer un module de contrôle permettant la gestion des différents flux évoluant dans un réseau intelligent.

Les limites de cette architecture sont les suivantes : elle ne prend pas en compte de base l'aspect sécurité des échanges de données entre les différents éléments du réseau, la climatisation des serveurs, et de même la climatisation des espaces de stockage et plus particulièrement des batteries. L'abstraction des éléments du réseaux intelligent permet de prendre en compte ces limites, étant donné que le réseau peut intégrer ses éléments à travers l'utilisation de modèles ou simulateurs du domaine dans ce système. Par la suite, le module de contrôle possède les informations nécessaires, il ne reste plus qu'à l'intégrer à son système de décision.

Cependant, pour réaliser de telles simulations, nous avons besoin de données réelles à grande échelle de la charge d'un centre de calcul. C'est pourquoi, le chapitre 4 suivant est une étude de charge de travail afin de générer des traces synthétiques directement utilisable dans le simulateur.

Chapitre 4

Analyse de charges de travail de serveurs à grande échelle

Sommaire

Dans le chapitre précédent, nous avons présenté la conception d'un réseau intelligent contrôlé par un module central, permettant de prendre des décisions pour satisfaire la demande de charge de travail des centres de calcul.

Ce chapitre 4 va permettre de caractériser une charge de travail d'un centre de calcul à grande échelle, son comportement dans le temps et dans l'espace (ordonnancement). Afin de valider les

différentes contributions proposées dans les chapitre 5 et 3, il convient de connaître :

- les besoins fonctionnels pour obtenir des caractéristiques de traces réelles puis synthétiques ;
- les besoins en terme d'infrastructure (caractéristiques des serveurs).

Pour cela, nous avons analysé des traces déjà existantes, afin de proposer un modèle de traces pour mettre en place les algorithmes d'ordonnancements proposés dans le chapitre 5. Dans ce chapitre, il est proposé :

- une analyse pour générer de telles données, à travers les traces mises à disposition par Google. Cette analyse porte sur les particularités statistiques des traces Google, mais aussi sur leur contexte d'utilisation.
- un modèle de traces synthétiques qui peut être introduit dans le simulateur RenewSim (chapitre 3) afin d'implémenter un système de décisions cohérent.

Dans une première partie sont décrites les caractéristiques essentielles de traces à vocation de validation. Dans une deuxième partie est présenté le choix des traces Google. Dans une troisième partie, la présence des traces Google dans la littérature est exposée. Dans une quatrième partie est introduit la composition de ces traces. Dans une cinquième partie, une analyse des traces est proposée. Dans une sixième partie, les traces synthétiques sont détaillées. Dans une septième partie sont exposés le bilan et les limites des traces synthétiques.

4.1 Notre approche et nos motivations de l'utilisation de traces à grande échelle

Dans le cadre de neOCampus, un centre de calcul pour un campus doit être capable de réaliser des calculs importants. Les traces Google permettent donc de viser une grande charge. Dans ces travaux de thèse, l'utilisation de traces réelles a une vocation de validation et simuler un environnement réel. Elles ont pour rôle :

- de garantir l'utilisation des ressources ;
- de mesurer et respecter la QoS ;
- d'être utilisées dans un contexte d'infrastructure réel de centre de calcul qui lui-même est composé de plusieurs serveurs ayant des caractéristiques hétérogènes ;

Cependant, les services (tâches, jobs, VMs) qui la composent influent sur l'utilisation des ressources. De part leurs placements sur des machines physiques, la QoS des utilisateurs sera influencée. Le placement des tâches, et leurs distinctions ainsi que les caractéristiques des ressources vont impacter directement la consommation d'énergie. Par exemple, une tâche longue ou courte utilisant n unités physiques consommera plus ou moins si elle est placée sur un serveur possédant un microprocesseur d'une puissance plus ou moins grande. Afin de réaliser les validations des

différents algorithmes présentés dans le chapitre 5 et l'environnement de simulation présenté dans le chapitre 3, il faut des traces existantes avec des caractéristiques réelles. Pour la réalisation des expérimentations (chapitre 6), nous allons utiliser des traces synthétiques. L'objectif est d'analyser des traces existantes à grande échelle pour obtenir un panel caractéristique pour établir de nouvelles traces.

Il existe plusieurs traces à grande échelle disponibles. « Parallel Workload Archive » [47] est un ensemble de traces sur des données d'utilisation de jobs pour des clusters ou des supercalculateurs. Historiquement, elles étaient pour un usage de grilles de calculs et maintenant, elles se sont étendues pour des traces de charges de travail. Ces données émanent de centres de calcul séparés géographiquement ainsi que des modèles. Il y a des données provenant de processus réels et des données modélisées basées sur du réel. De plus, certaines de ces données ont été mesurées, il y a plus d'une dizaine d'années.
Afin de réaliser des traces synthétiques, le choix s'est porté vers les traces proposées par Google [8]. Celles-ci sont composées de priorités liées à la QoS, ou liées à des contraintes temporelles. Ce sont des données réelles provenant d'un cluster de Google. Les traces Google sont composées de 25 millions de tâches, de plus de 12 000 serveurs et ont été récoltées sur 29 jours. L'étude de ces traces sont très présentes dans la littérature et sont représentatives des données à grande échelle.

4.2 Les traces Google dans la littérature

Plusieurs études se sont précédemment portées sur une meilleure compréhension de l'ensemble des traces Google et de leurs caractéristiques. Cette littérature permet de connaître :

1. l'hétérogénéité de l'infrastructure des ressources physiques des centres de calcul ;
2. les ressources demandées par les tâches ;
3. la classification des tâches qui composent la charge de travail.

Reiss *et al.* [81] ont décrit l'hétérogénéité des différentes ressources physiques (« hardware ») présentes dans les traces Google. Ils ont classifié les priorités en cinq catégories : « Infrastructure (11) », « Monitoring (10) », « Normal production (9) », « other (2-8) » et « Gratis (0-1) ». Ils ont aussi identifié les « *boulders* », en français, « Rochers » et les « *sand* », en français, « Sable » de cette charge de travail : une majorité de petites tâches (« *sand* ») et un nombre considérable de tâches de tailles importantes (« *boulders* »). Dans cette analyse, ils ont montré que 2% des tâches représentées 80% du CPU, de la mémoire, et que 92% sont des tâches longues avec une priorité de type « *Free* ». De plus, Reiss *et al.* [82] ont analysé les performances des centres de calcul. Selon cet article, de nombreuses tâches longues ont une utilisation stable des ressources, ce qui aide les ordonnanceurs de ressources. Ils ont conclu que la configuration de la machine et la composition de la charge de travail sont hétérogènes. Cette trace analysée nous

permet de comprendre l'ensemble des données proposées par Google, l'importance d'un centre de calcul hétérogène pour adapter, par exemple, les ressources à la demande. Cependant, ils n'ont pas fourni d'information sur la façon d'utiliser cette charge de travail dans un autre contexte.

L'article [83] évalue l'écart entre les ressources demandées et celles consommées par les tâches. La charge moyenne demandée par un processeur est de 10% sur les centres de calcul Google. Cette analyse de traces Google montre que les processeurs sont globalement sous utilisés ce qui conduit à une augmentation de l'énergie consommée (90 % de la puissance de calcul du processeur disponible n'est pas utilisée). De plus, la majeure partie de la charge de travail a une faible priorité et n'est pas sensible à la latence (note de capacité d'adaptation à attendre une ressource). Cette analyse montre qu'il n'y a que très peu de tâches sensibles égale à 2 ou 3. Dans cet article, les auteurs se concentrent sur le manque de considération d'énergie dans les traces de Google.

Liu *et al.* [84] ont analysé comment les serveurs sont gérés dans le cluster et comment la charge de travail se comporte. Selon eux, il y a plus de 870 événements machines en moyenne chaque jour. Dans cette étude, ils ont groupé les différentes machines par même CPU et mémoire (15 groupes). Contrairement à [82], [84] montrant que les machines sont presque homogènes, 93 % machines ont les mêmes capacités CPU et 86 % des machines ont les mêmes capacités de mémoires. Les auteurs ont également exploré la durée de vie de la charge de travail. Ils montrent que beaucoup de tâches sont tuées, car 40,52 % des tâches ordonnancées sont tuées au moins une fois.

Alam *et al.* [85] ont fourni une analyse statistique des traces pour faire émerger des profils de travail de référence. Ils ont fondé cette approche sur l'utilisation des ressources, le regroupement des modèles de charge de travail et la classification des tâches avec des regroupements. Ils ont démontré que les tâches Google sont *tri-modales*. Ils peuvent être *Jobs longs*, *Jobs courts* et *Jobs moyens*. Chaque type de job peut également être sous-catégorisé comme *Utilisation faible des ressources*, *Utilisation moyenne des ressources* et *Ressources affamées*. Ils ont groupé des jobs avec un k-mean, avec k égal à 5 (nombre de classes). Les jobs peuvent également être classés en *courts*, *presque moyens*, *moyens*, *presque longs* et *longs*. Ce clustering vise à utiliser cette charge de travail dans un simulateur. Le regroupement des profils de tâches est également le résultat principal de [86].

Ces études montrent que les données de la charge de travail sont caractérisées (priorité, sensibilité à la latence). La partie suivante présente la composition des traces Google et l'articulation de toutes ces données entre elles.

4.3 Composition des traces Google : des données de tailles importantes

Dans cette partie, l'objectif est de comprendre l'organisation des traces Google dans le but de créer des traces synthétiques. Dans un premier temps, la structure est présentée puis, dans un deuxième temps le fonctionnement des tâches.

4.3.1 Structure des traces Google

En 2011, Google a rendu public des traces sur l'utilisation de certains de leurs serveurs [8]. Ces traces incluent les demandes d'ordonnanceurs et leurs actions. Les données relatives aux différentes tâches et de leurs utilisations de ressources au cours du temps sont aussi connues. De plus, elles permettent de connaître la disponibilité des différents serveurs au cours du temps. Certaines données ont été normalisées entre 0 et 1, afin de cacher la configuration exacte. Pour la sécurité de Google, l'utilisation du CPU, de la mémoire, et de l'utilisation du disque pour un job sont donc normalisées. La taille des traces est de 43 GB. La figure 4.1 présente la structure de celles-ci. Le répertoire des traces contient un ensemble de fichiers CSV. Les données collectées dans l'ensemble de ces fichiers sont mesurées au niveau de l'ordonnanceur de Google et au niveau des différentes machines. Chaque répertoire est décrit dans le tableau 4.1.

FIGURE 4.1 – Architecture des données de Google

Google dispose de jobs et de tâches qu'il place sur des machines. Un job est composé d'une ou plusieurs tâches, les tâches sont accompagnées d'un ensemble d'exigences en terme de ressources pour pouvoir être ordonnancées sur des machines. Chaque tâche représente un programme Linux pouvant être exécuté sur une seule machine. Les capacités des machines physiques ne sont également connues. Il est donc impossible d'évaluer la consommation d'un tel cluster. Il existe aussi une complexité au niveau des liens entre les différents dossiers de données. Par exemple, **task_events** est liées aux contraintes des machines (**machine_attributes**) et des ressources disponibles (**machine_events**) et à ses propres contraintes définies dans **task_constraints**. Il y a une description à plusieurs niveau des jobs (**job_events**) et de leur composition en tâches (**task_events**). Cela aide à connaître le cycle de vie d'un job et d'une tâche durant les 29 jours de mesures.

Tableau 4.1 – Description de l'architecture des données Google

Nom répertoire	Description
job_events	Ce répertoire décrit les jobs et leur cycle de vie. Il contient 500 fichiers de type CSV (315 MB). Un job possède une sensibilité à la latence (≪ Scheduling Class ≫). C'est-à-dire que le temps d'ordonnancement et d'exécution du Job doit correspondre à certaines contraintes de temps afin de satisfaire l'utilisateur. Un job au cours de son cycle de vie passe par différents états qui sont décrits par l'attribut ≪ Event Type ≫.
machine_attributes	Ce répertoire contient un seul fichier CSV (1.1 GB). Ce fichier décrit toutes les capacités d'une machine. Les données ont été normalisées à l'aide d'un entier afin de spécifier les contraintes pour les différentes tâches (voir description ≪ task_constraints ≫).
machine_events	Ce répertoire contient un seul fichier CSV (3 MB). Il décrit les différentes machines au cours du temps, suivant trois événements : - ≪ ADD ≫ : La machine est disponible dans le cluster. - ≪ REMOVE ≫ : La machine ne fait plus partie du cluster. - ≪ UPDATE ≫ : La machine a ses ressources disponibles mises à jours.
task_constraints	Ce répertoire contient 500 fichiers CSV (2.8 GB). Une tâche peut avoir aucune ou plusieurs contraintes de placement. Ces fichiers permettent de décrire les machines idéales pour une tâche.
task_events	Ce répertoire contient 500 fichiers CSV (15.4 GB). Comme ≪ job_events ≫, l'ensemble de ces fichiers décrit le cycle de vie d'une tâche. Une tâche au cours de son cycle de vie passe par différents états qui sont décrits par l'attribut ≪ Event Type ≫.
task_usage	Ce répertoire contient 500 fichiers CSV (158 GB). Google a enregistré toutes les valeurs d'utilisation d'une tâche pour chaque période de mesure.

Pour générer des traces synthétiques, il manque les aspects d'énergie et de consommation qui ne sont pas représentés. Cependant, l'avantage est qu'elles offrent plusieurs niveaux de composition de charge de travail. Un job est composé de une ou plusieurs tâches. Pour générer une charge de travail d'un centre de calcul, il convient de s'intéresser au dossier **task_events** dans un premier temps car la composition des jobs (**job_events**) découlent directement du fichier **task_events**. C'est ce dossier que nous allons étudier dans la partie suivante.

4.3.2 Cycle de vie des tâches

Les tâches et les jobs passent d'un état à l'autre via des événements. Les tâches en état *unsubmited* attendent pour avoir une ressource qui correspond à ses contraintes. Elle va être ordonnancée et assignée au processeur approprié pour l'exécution. Elle peut être annulée ou tuée avant d'être ordonnancée. Cependant, les tâches tuées peuvent être derechef ordonnancées dans un autre job. L'ordonnanceur de Google utilise les propriétés des tâches pour réaliser une décision. Si elles ont les mêmes priorités, alors les tâches sont choisies par ordre d'arrivée. Ces données,

concernant les différentes tâches, sont accessibles via trois fichiers : *task events*, *task constraints* et *task ressource usage*. Les jobs et les tâches sont planifiés sur les machines en fonction du cycle de vie décrit dans la figure 4.2.

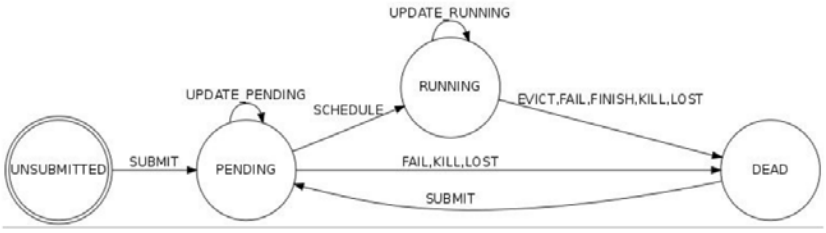


FIGURE 4.2 – États des tâches et les événements au cours de leur cycle de vie (Documentation Google [8])

Une tâche est décrite avec les informations présentées dans le tableau 4.2. Pour passer d'un état à un autre, une tâche utilise des événements. Les différents *event type* sont les suivants :

- 0 pour **submit** : la tâche devient éligible pour être ordonnancée.
- 1 pour **schedule** : la tâche a été ordonnancée sur une machine.
- 2 pour **evict** : la tâche a été dé-ordonnancée car soit une priorité plus élevée est arrivée, soit l'ordonnanceur est surchargé et la demande actuelle est supérieure aux capacités de la machine, soit une machine sur laquelle elle est exécutée est devenue inutilisable, et soit le disque a perdu la tâche.
- 3 pour **fail** : la tâche a été dé-ordonnancée à cause d'un problème lié à la tâche.
- 4 pour **finish** : la tâche a été accomplie normalement
- 5 pour **kill** : la tâche a été annulée par l'utilisateur ou le driver car une tâche dépendante de celle-ci a été tuée.
- 6 pour **lost** : la tâche était supposée terminer mais un enregistrement indique que la terminaison manque dans les sources de données.
- 7 pour **update_pending** : la tâche a été mise à jour pendant son attente pour être ordonnancée.
- 8 pour **update_running** : la tâche a été mise à jour pendant qu'elle été ordonnancée.

À travers la description des tâches ci-dessus, il existe beaucoup d'événements pour des tâches qui ne se finissent pas (**kill**, **evict**, **fail**), et pour des pour des tâches ayant des problèmes d'enregistrement ou de mise à jour. La normalisation de la valeur de l'attribut **ressource request for CPU cores** utilisé, ne nous permet pas de calculer la consommation d'une tâche. Cet attribut est un pourcentage d'utilisation (noté entre 0 et 1) de la machine physique sur laquelle la tâche se trouve. Or, les capacités des machines physiques Google sont elles aussi cachées. Cette normalisation apparaît aussi pour les attributs suivants : **ressource request for RAM** et **ressource**

request for local disk space. Cependant, la description des tâches possède les événements **submit** et **schedule**. La séparation de l'événement d'arrivée et d'exécution aide à calculer le temps d'attente d'une tâche, et ainsi connaître la QoS. Pour la génération de traces synthétiques, l'attribut **missing info** peut être omis car, il découle de l'ordonnancement prévue par les centres de calcul de Google. Pour les événements de fin, ils peuvent être séparés en deux catégories les tâches qui finissent bien (**finish**) et les autres qui se terminent de manière anormale (**evict**, **fail**, **kill** et **lost**). Tout les événements ne sont pas à prendre en compte dans les futures traces synthétiques. La section suivante, entre autre, va permettre d'analyser les différents événements de fin et de connaître leurs poids dans les traces.

Tableau 4.2 – Les attributs d'une tâche par Google

Attribut	**Description**
timestamp	Le temps de la mesure (normalisé par Google) en millisecondes.
missing info	Les enregistrements Google proviennent de capture de données à un instant. Cela permet de suivre l'évolution d'une tâche. Cependant, lorsqu'il manque un événement d'enregistrement, il réalise un remplacement de cette information. Il y a trois types de remplacement (0 : SNAPSHOT_BUT_NO_TRANSITION, 1 : NO_SNAPSHOT_OR_TRANSITION, 2 : EXISTS_BUT_NO_CREATION). Les enregistrements sans informations manquantes n'ont pas ce champs rempli.
job ID	L'identifiant du job est caché par Google, il a un nom opaque en Base 64. Il peut être testé pour des égalités.
task index -within the job	L'identifiant d'une tâche. Si le ≪ task index ≫ est égal au ≪ job ID ≫, la tâche peut être stoppée et relancée sans assigner un nouvel ≪ job ID ≫ et un nouvel ≪ index ≫.
machine ID	L'identifiant de la machine est caché par Google, il a un nom opaque en Base 64. Il peut être testé pour des égalités.
event type	Cet attribut fait référence à la figure 4.2. Il correspond au type d'événement pour passer d'un état à l'autre.
user name	L'identifiant de l'utilisateur est caché par Google, il a un nom opaque en Base 64. Il peut être testé pour des égalités.
scheduling class	Cela permet de connaître la sensibilité à la latence d'une tâche. C'est-à-dire le temps d'ordonnancement et d'exécution de la tâche doit correspondre à certaines contraintes de temps afin de satisfaire l'utilisateur. Le ≪ scheduling latency ≫ est classé de 0 à 3 sachant que 0 est le niveau le plus faible, et 3 le niveau le plus élevé. Ce n'est pas une priorité.
priority	Il y a trois types de priorité : ≪ free ≫(0 -1) c'est le niveau de priorité le plus bas, ensuite il y a ≪ monitoring ≫ (2-8) et puis vient ≪ production ≫(9-11) le niveau de priorité le plus élevé.
ressource request for CPU cores	La demande CPU de la tâche. L'unité est normalisée elle va de 0 à 1 (pourcentage d'utilisation).
ressource request for RAM	La demande RAM de la tâche. L'unité est normalisée elle va de 0 à 1 (pourcentage d'utilisation).
ressource request for local disk space	La demande disque de la tâche. L'unité est normalisée elle va de 0 à 1 (pourcentage d'utilisation).
different-machine constraint	Si cet attribut existe et qu'il est à ≪ vrai ≫, cela indique qu'une tâche doit être programmée pour être exécutée sur une machine différente.

4.4 Notre analyse des traces Google

Afin de générer des traces synthétiques pour valider les différents algorithmes et modèles proposés dans les chapitres 3 et 5, il convient d'analyser les tâches proposées par Google, pour obtenir une charge réelle de centre de calcul. Cela permettra dans le chapitre 6 d'obtenir un environnement favorable à la validation des expérimentations. Pour les statistiques suivantes nous avons considéré les 500 fichiers CSV contenu dans le dossier « task_events ».
La structure de cette analyse est la suivante :

- Dans un premier temps, le nombre d'événements de tâches est analysé pour comprendre son cycle de vie.
- Dans un deuxième temps, pour connaître l'hétérogénéité des tâches, il est énuméré la répartition des tâches en fonction de leurs attributs de priorité et de sensibilité à la latence (« scheduling class »).
- Dans un troisième temps, une analyse permet de comprendre le type de priorité et de sensibilité à la latence associés aux événements de fin d'une tâche.
- Dans un quatrième temps, la durée d'exécution des tâches est présentée.
- Dans un cinquième temps, une étude du pourcentage de CPU utilisé est inspectée.
- Dans une sixième partie, la QoS des tâches est sondée.
- Dans une septième partie, le bilan des différentes hypothèses soulevées est exposé.

4.4.1 Le nombre d'événements par tâche

Dans le tableau 4.3, 95 % des tâches possèdent trois événements : « Submit », « Schedule » et un événement de fin (« Fail », « Kill », « Evict », « Finish »). Cependant, certaines tâches (4.5 %) n'ont pas d'événement de fin. Dans les analyses suivantes, nous nous sommes seulement intéressés aux tâches possédant ces 3 événements, car cela représente la majeure partie des traces. Dans [87], le tableau 4.4 met en avant les événements de fin pour les différentes tâches à 3 événements. 73% des tâches finissent correctement et 26% finissent par être tuées. Presque toutes les tâches qui ne finissent pas normalement sont les événements « Kill ». Dans la description des traces par Google [8], aucunes informations renseignent la raison de ces fins anormales.

Tableau 4.3 – Distribution du nombre d'événements par tâche [87]

Nombre d'événement	**1**	**2**	**3**	**4**	**5**
Nombre de tâches	248	1,093,992	24,258,907	19,378	17,215
Nombre d'événement	**6**	**7**	**8**	**9+**	**total**
Nombre de tâches	8,510	5,093	3,028	18,360	25,424,731

TABLEAU 4.4 – Distribution des événements de fin pour les tâches à trois événements [87]

Type de l'événement	**Finish**	**Kill**	**Fail**	**Other**
Nombre de tâches	17,775,284	6,381,906	86,348	15,369
Pourcentage	73.31 %	26.31 %	0.35 %	0.03 %

Hypothèses 1

95 % des tâches possèdent trois événements : ajout à la liste d'attente (**submit**), exécution de la tâche (**schedule**) et fin de la tâche (**finish**, **kill**, **fail**, et **other**). Pour les analyses suivantes, les études se porterons uniquement sur les tâches ayants ces trois événements. De plus, beaucoup d'événements sont peu représentés (**evict**, **lost**, **update_pending** et **update_running**). Il convient de s'intéresser aux événements suivants : **finish**, **kill**, et **fail**. Pour les analyses suivantes nous avons inclus les tâches ayant l'événement de fin **evict** car par exemple une tâche peut être arrêtée en cours d'exécution à cause d'une tâche ayant une priorité plus élevée afin de respecter la QoS à un instant t.

4.4.2 Le nombre de tâches à trois événements

Dans [87], la classification des tâches est répartie par rapport à leurs priorités en quatre catégories :

- « monitoring » : les priorités 11 et 10 (peu présentent dans les tâches à trois événements) ;
- « production » : la priorité égale à 9 ;
- « normal » : les priorités de 2 à 8 ;
- « free » : les priorités 0 et 1

Les illustrations suivantes présentent la répartition des tâches en fonction de leur priorité et sensibilité à la latence.

Dans la figure 4.3 presque 60 % des tâches ont une priorité égale à 4 (moyenne), tandis que plus de 80 % des tâches ont une sensibilité à la latence égale à 0 (figure 4.4). Dans la figure 4.5, par conséquent le couple « 4_0 » représente presque 50 % des tâches présentes dans la traces de Google, alors que le couple « 0_0 », le plus faible (priorité et sensibilité à la latence), ne représente que 20 % des tâches. Les tâches à haute priorité sont peu représentées. Il n'y a aucune tâche de priorité 7 et 11.

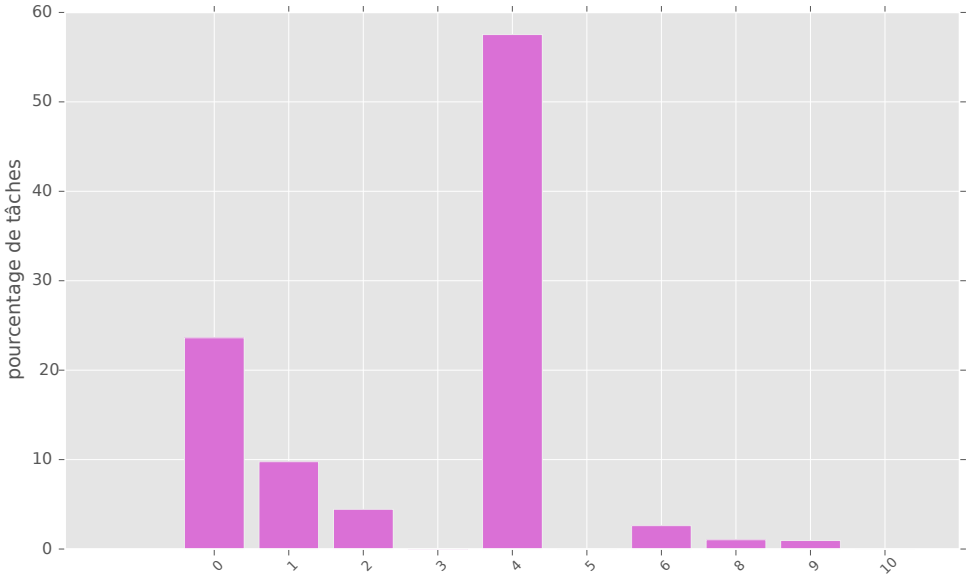


FIGURE 4.3 – Répartition des tâches par rapport à la priorité

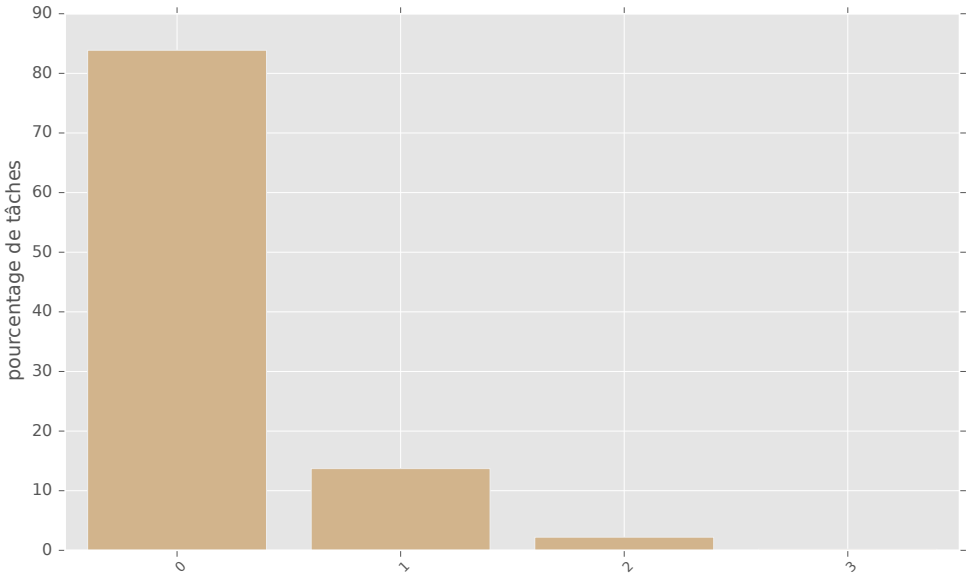


FIGURE 4.4 – Répartition des tâches par rapport à la latence

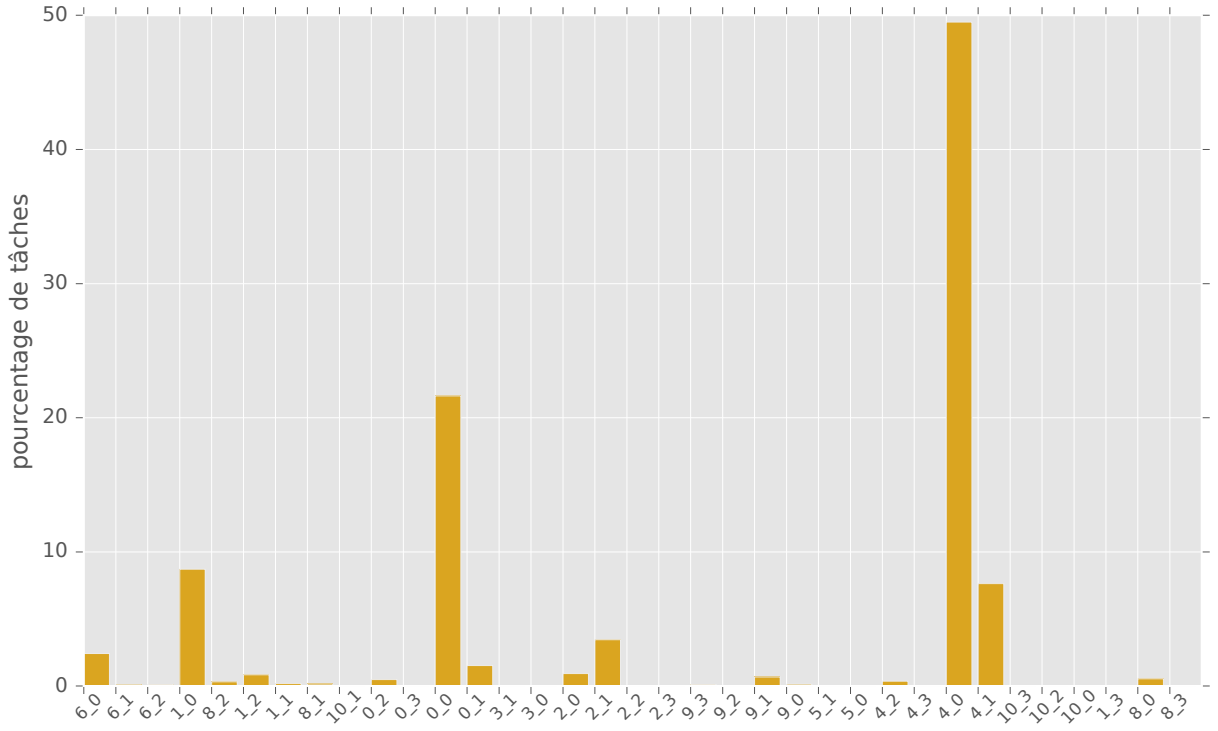


FIGURE 4.5 – Répartition des tâches par rapport aux couples (priorité - latence)

Hypothèses 2

Les tâches de type « monitoring » correspondent à moins de 0.002 % des traces, ce qui est très insignifiant. Les tâches à haute priorité (« production ») ont une sensibilité à la latente la plus élevée tandis que les tâches « normal » ont une sensibilité à la latence faible. Google n'apporte aucune information sur le type de tâches présentées et leurs domaines d'exécution. Cependant, ces tâches peuvent être de nature applicative et demandent un temps de réponse très faible afin d'être exécutées le plus rapidement possible. Les tâches ayant une haute sensibilité et une haute priorité peuvent être des tâches de type service. Si ces tâches ont ces caractéristiques, il convient de connaître leur événement de fin.

4.4.3 Les événements de fin pour les tâches

La figure 4.6 montre que plus de 73.31 % des tâches finissent correctement contre environ 26.31 % par un événement de type « Kill ». La figure 4.7 met en avant les événements de fin pour chaque priorité. Nous remarquons que les tâches à faible priorité ont plus de chance de finir correctement (0,1) alors que les tâches à haute priorité finissent par un « Kill ». La figure 4.8 met en avant les événements de fin par rapport à leur sensibilité à la latence (« scheduling class »).

Nous remarquons que la répartition des tâches est à peu près similaire sur les quatre niveaux de sensibilité à la latence.

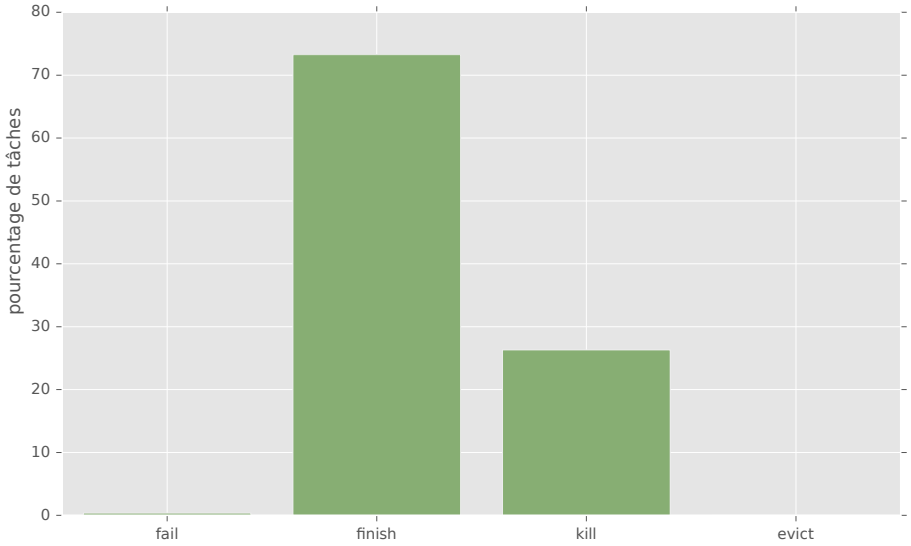


FIGURE 4.6 – Répartition des tâches par rapport à leur événement de fin

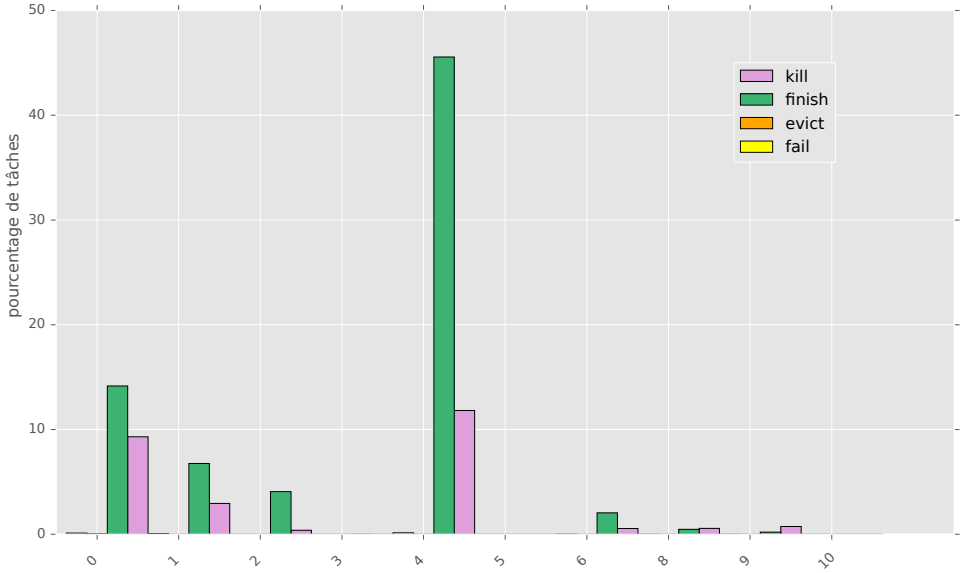


FIGURE 4.7 – Répartition des tâches par rapport à leur événement de fin et de la priorité

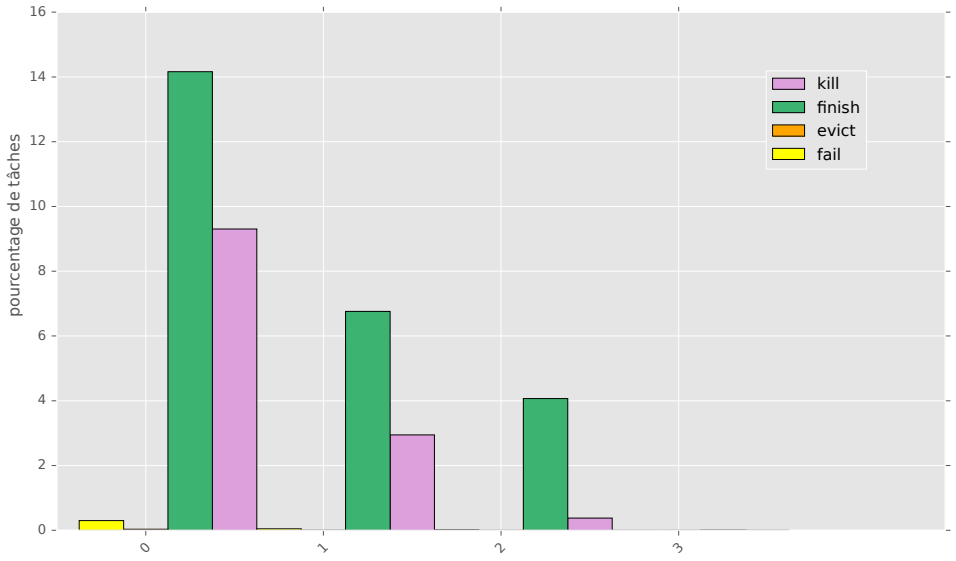


FIGURE 4.8 – Répartition des tâches par rapport à leur événement de fin et de la latence

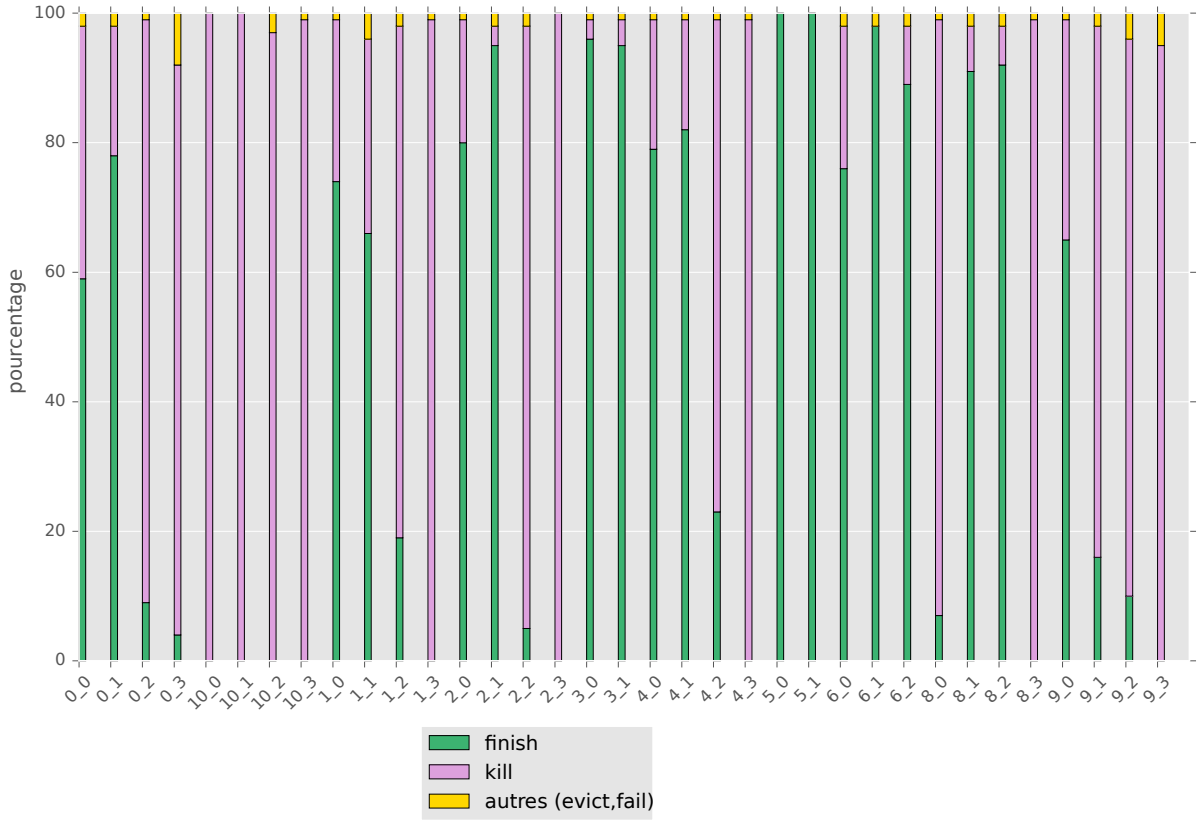


FIGURE 4.9 – Pourcentage d'événements de fin pour chaque couple (priorité - latence)

La figure 4.9 est un bilan des figures 4.7 et 4.8. Les tâches sont réparties par couple (priorité et sensibilité à la latence). Pour chaque couple, il y a le pourcentage de tâches pour chaque type d'événements de fin. Plus la priorité et la sensibilité à la latence sont élevées plus les événements de fins sont des « Kill ». Les tâches qui ont plus de chances de finir normalement (« Finish ») sont les couples : 2_1 - 3_0 - 3_1 - 4_0 - 4_1 - 5_0 - 6_0 - 6_1 - 6_2 - 8_1 - 8_2

Dans la section 4.4.2, les tâches « 4_0 » sont les tâches les plus présentes. Ici, elles ont en moyenne beaucoup de chances de finir bien (presque 80 % de « Finish »).

Hypothèses 3

Les tâches ayant une haute priorité et sensibilité à la latence ont plus d'événements de fin de type « Kill ». Cela peut s'expliquer par le type de tâche à exécuter. Effectivement, cela peut correspondre à des tâches de type service. Par exemple, le lancement d'une application ou d'une requête web. Si le service n'est plus demandé par l'utilisateur alors la tâche est tuée. Il existe un lien entre les caractéristiques d'une tâche et son événement de fin. Si la tâche a une priorité haute, elle se termine souvent anormalement. Cependant, il existe des tâches dans la catégorie « normal »(section 4.4.2) qui se termine par un « Finish », alors que la latence ou la priorité est élevée.

4.4.4 Les temps d'exécution des tâches

La figure 4.10 présente le temps moyen d'exécution d'une tâche par rapport à son événement de fin. Les tâches finissant par un « Kill » durent beaucoup plus longtemps que les autres événements. Les tâches finissant par un « Finish », ont un temps moyen d'exécution supérieur aux tâches ayant une fin anormalement (« Evict » et « Fail »).

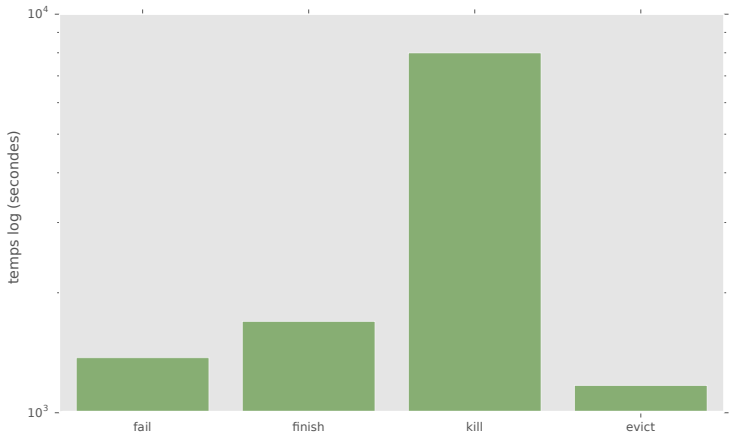


FIGURE 4.10 – Durée moyenne d'exécution par rapport à l'événement de fin d'une tâche

Les figures 4.11 et 4.12 représentent le temps moyen d'exécution des tâches par rapport à la priorité et à la sensibilité à la latence. Les tâches ayant une priorité forte (9 ou 10) ont une durée de vie plus importante. Cependant, les tâches ayant une priorité égale à 3, ont tendance à durer plus longtemps que les autres priorités (hors 9 et 10). Les tâches ayant une sensibilité à la latence élevée durent plus longtemps que les autres (2 et 3). Cependant, les « scheduling class » qui sont à 2, durent plus de temps en moyenne que celles à 3.

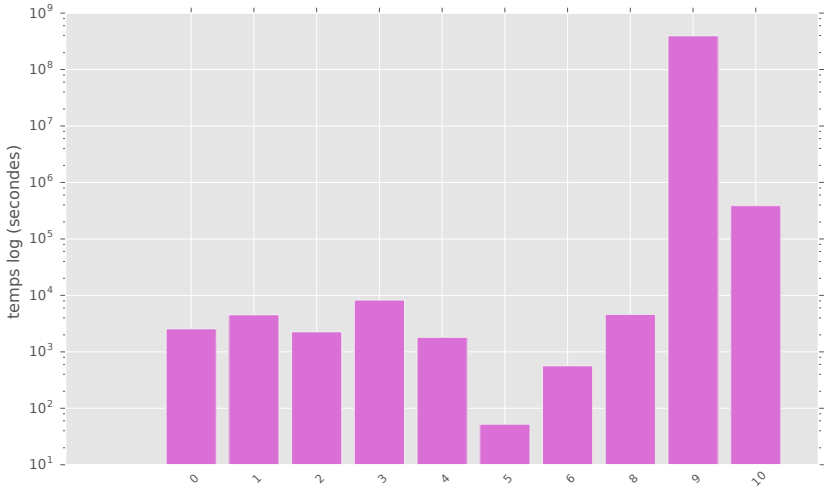


FIGURE 4.11 – Temps moyen d'exécution des tâches par rapport à la priorité

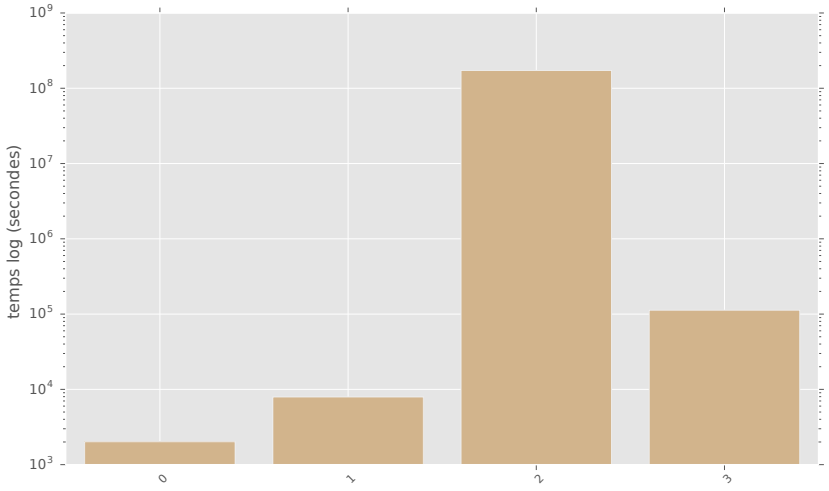


FIGURE 4.12 – Temps moyen d'exécution des tâches par rapport à la latence

La figure 4.13 représente le temps moyen d'exécution par rapport au couple (priorité et sensibilité à la latence). Le couple ayant le temps moyen le plus long est 9_2. Le couple 1_1 possède un temps d'exécution assez élevé. Cependant, la priorité et la « scheduling class » ne correspondent

pas aux types de tâches ayant des temps d'exécution long (cf. figures 4.11 et 4.12).

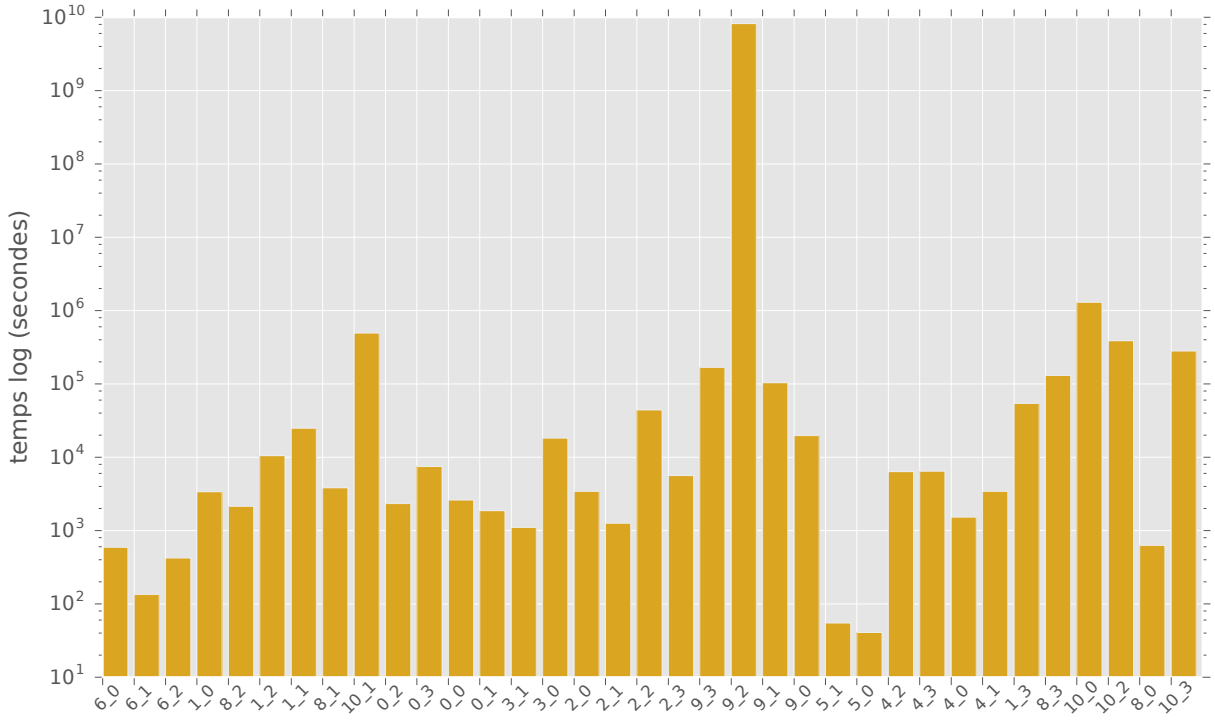


FIGURE 4.13 – Durée moyenne d'exécution par rapport au couple (priorité - latence)

Hypothèses 4

La priorité et la sensibilité à la latence influent sur la durée d'exécution d'une tâche. Si la priorité est égale à 9 ou 10 et que la « scheduling class » est égale à 2, alors les tâches dureront longtemps plus longtemps. Un coefficient peut être alloué en face de chaque couple pour déterminer le temps d'exécution d'une tâche. Cependant, des tâches ont des temps moyens d'exécution élevés alors que ni la priorité et ni la « scheduling class » sont élevées. De telles tâches peuvent être associées à des services de types applicatifs. Les tâches applicatives sont généralement gourmandes en consommation.

4.4.5 Le CPU utilisé pour les différentes tâches

La figure 4.14 présente la moyenne de CPU utilisée pour les différentes priorités. Les tâches dont le CPU utilisé est le plus élevé, concernent les priorités 9, 3, 10, 8 et 4. La plus faible consommation de CPU correspond aux tâches dont la priorité est égale à 2. La figure 4.15 met en avant la moyenne de CPU utilisée par rapport à la sensibilité à la latence, plus la « scheduling class » est élevée plus la consommation de CPU est importante.

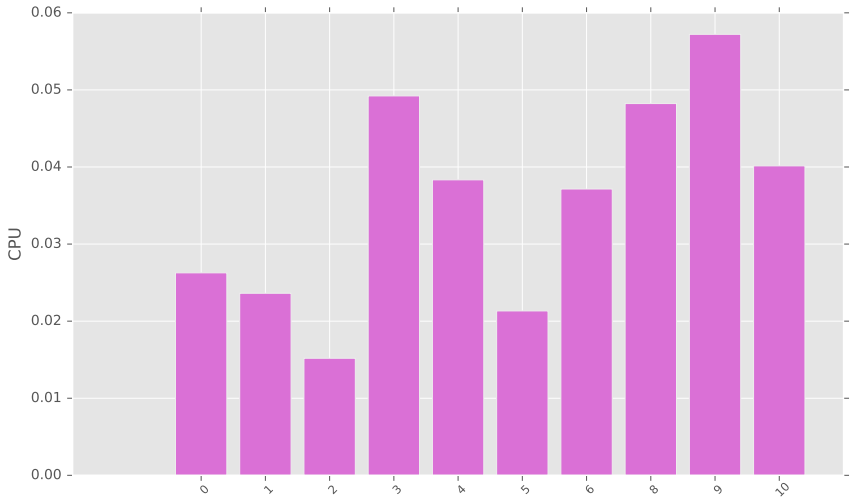


FIGURE 4.14 – CPU moyen des tâches par rapport à la priorité

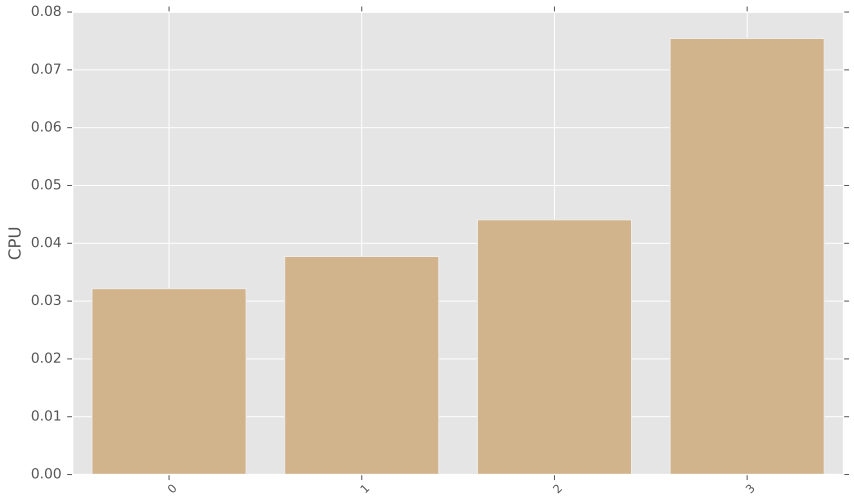


FIGURE 4.15 – CPU moyen des tâches par rapport à la latence

Dans la figure 4.16, les tâches ayant un événement de fin « Kill » utilisent environ en moyenne le double de CPU que les tâches finissant normalement (« Finish »). Dans la figure 4.17, la moyenne de CPU est la plus grande pour les couples « 9_3 » « 9_2 » et pour le couple « 6_2 ». Le couple « 6_2 » possède une majeure partie d'événements de fin de type « Finish ». Cependant, le couple « 6_2 » est peu présent dans les traces (figure 4.5 dans la partie 4.4.2).

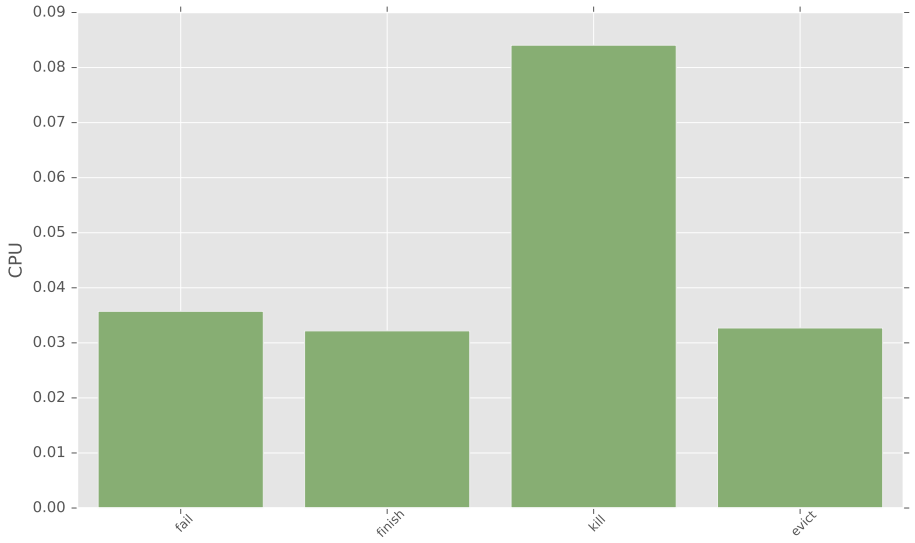


FIGURE 4.16 – Moyenne de CPU utilisée pour les différentes tâches par rapport à son événement de fin

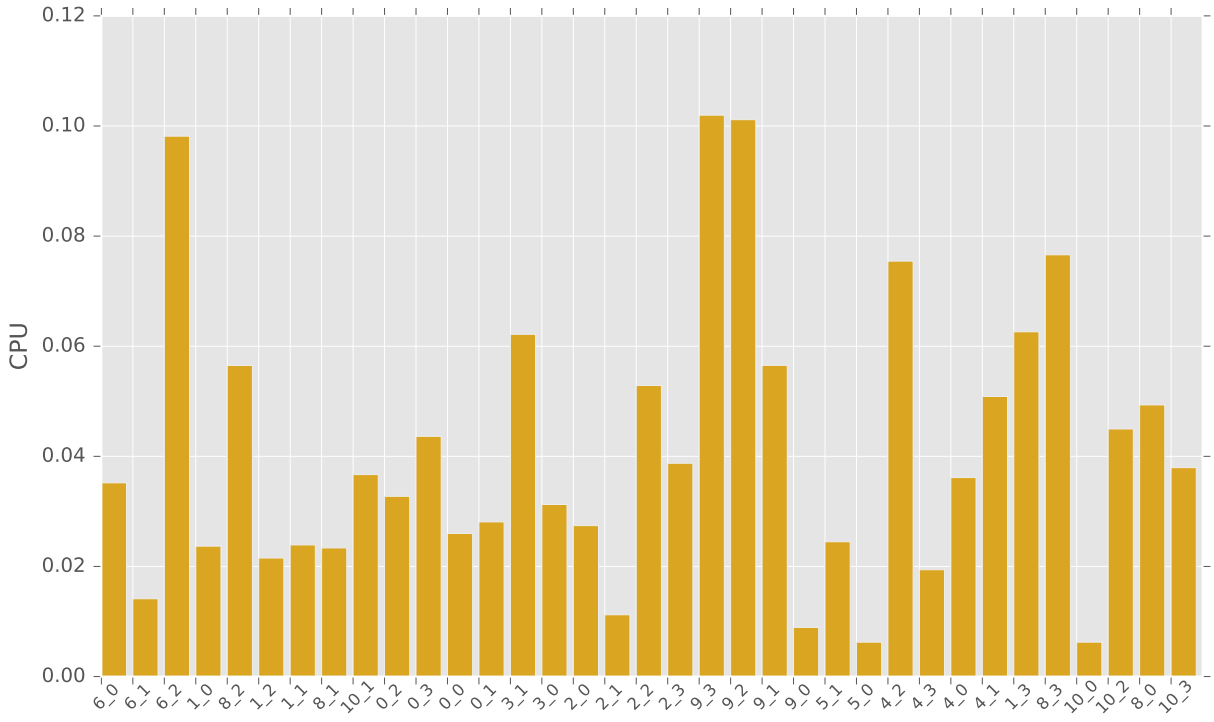


FIGURE 4.17 – Moyenne de CPU utilisée pour les différentes tâches par rapport aux couples (priorité - scheduling class)

Hypothèses 5

Dans les différentes figures nous remarquons deux hypothèses. La première (figure 4.16), si les tâches finissent anormalement (« Fail » , « Kill », « Evict »), en moyenne, les tâches consomment plus de CPU. La deuxième hypothèse met en avant la priorité et la sensibilité à la latence : plus ces deux paramètres sont élevés, plus la tâche consomme de CPU. Les tâches ayant une fin anormale nous permet d'appuyer l'hypothèse de différents types de services : application et batch ou log.

4.4.6 La QoS des différentes tâches

La figure 4.18 présente le temps moyen pour passer d'un événement à l'autre ainsi que le temps moyen d'exécution d'une tâche par rapport à sa priorité. Les tâches ayant une priorité élevée (9 et 10) ont un cycle de vie plus grand et un temps très faible pour passer de l'événement « submit to schedule ». La figure 4.19 représente le temps moyen de changement d'événements par rapport à la sensibilité à la latence (« scheduling class »). Les tâches ayant la sensibilité à la latence la plus élevée sont les « scheduling class » de type 2. La figure 4.20 présente les temps moyens par rapport aux couples. Les tâches ayant un « scheduling class » de 2 ont tendance à s'exécuter plus longtemps ainsi que les tâches ayant une priorité élevée. Le couple « 9_2 » possède les deux caractéristiques.

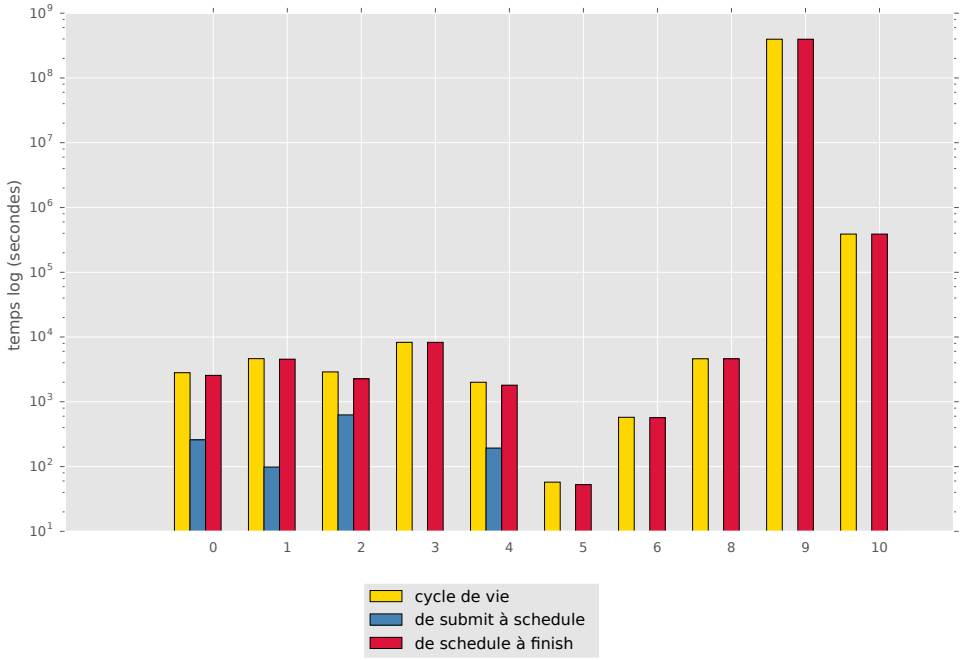


FIGURE 4.18 – Temps moyen d'ordonnancement d'une tâche par rapport à sa priorité

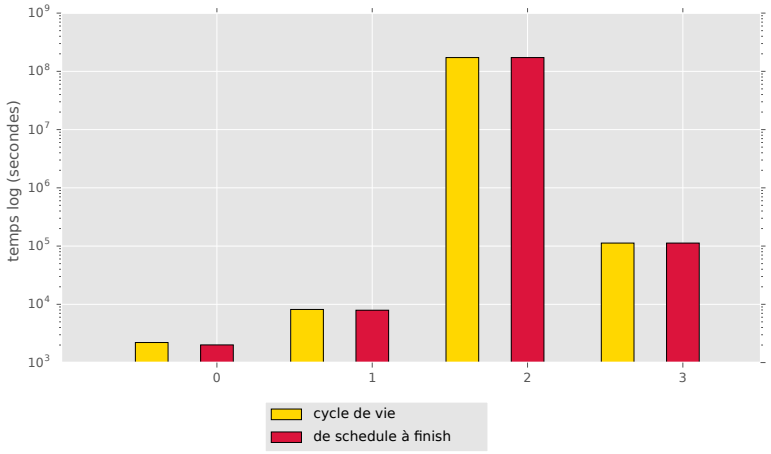


FIGURE 4.19 – Temps moyen d'ordonnancement d'une tâche par rapport à sa latence

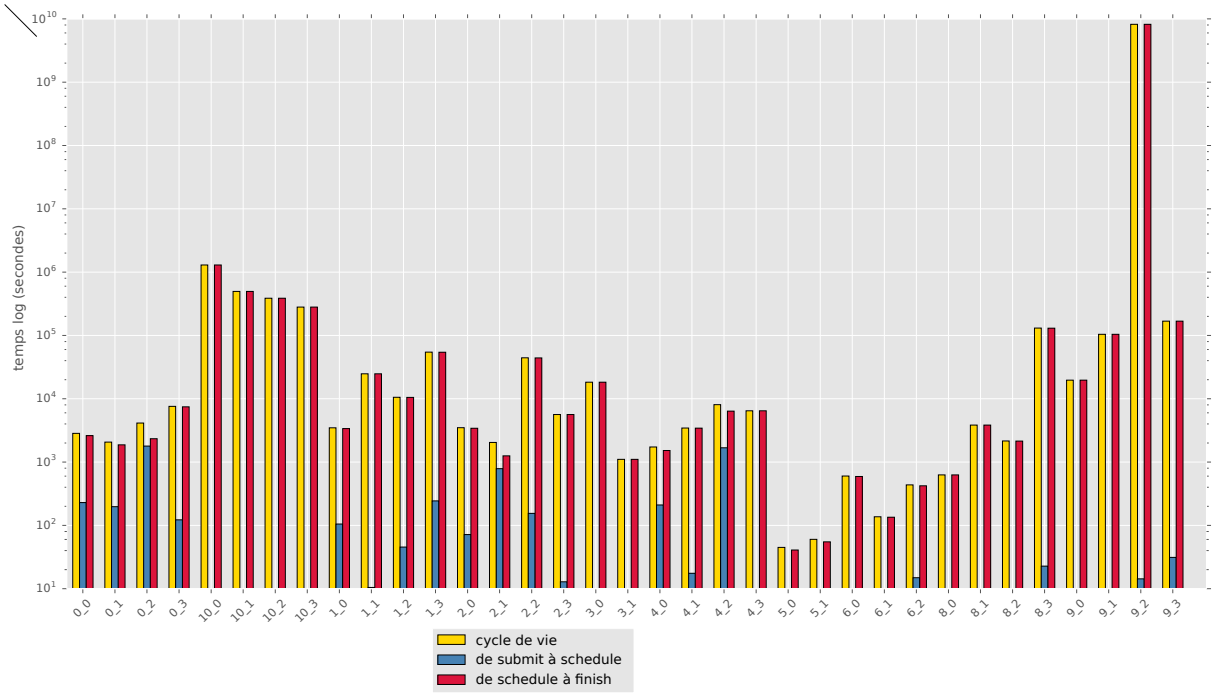


FIGURE 4.20 – Temps moyen d'ordonnancement d'une tâche par rapport aux couples (priorité - latence)

Hypothèses 6

Les tâches ayant une priorité élevée et un « scheduling class » égale à 2, possèdent un temps d'attente très faible pour passer de l'état « Submit » à « Schedule ». Elles ont donc une QoS très élevée. Les tâches de types application sont souvent demandées par des utilisateurs qui désirent obtenir un retour rapide. On peut donc poser l'hypothèse que les tâches ayant une sensibilité à la latence élevée doivent être rapidement exécutées afin de satisfaire un certain taux de QoS.

4.4.7 Bilan

Dans cette partie nous nous sommes intéressés aux différentes tâches des traces Google. Le tableau 4.5 est un récapitulatif des différentes hypothèses et des décisions prises pour l'élaboration de traces synthétiques.

Tableau 4.5 – tableau bilan des différentes hypothèses et décisions

Paramètres	**Analyses**	**Hypothèses**
Le nombre d'événements par tâches (paragraphe 4.4.1)	95% des tâches ont trois événements : « Submit », « Schedule » et un événement de fin (« Finish », « Kill », « Evict », « Fail »).	Ces tâches représentent l'ensemble des caractéristiques des traces Google.
Le nombre de tâches (paragraphe 4.4.2)	Les tâches de type « monitoring » correspondent à moins de 0.002 % des traces. Cependant, les tâches à haute priorité (« production ») ont une sensibilité à la latente la plus élevée tandis que les tâches « normal » ont une sensibilité à la latence faible.	Les tâches qui ont une priorité élevée sont peu nombreuses et ont une sensibilité à la latence plus élevée.
Les événements de fin de tâches (paragraphe 4.4.3)	Les tâches ayant une haute priorité et sensibilité à la latence ont plus d'événements de fin de type « Kill ».	Ils existent deux types de tâches : applicatives, et de traitements (batch,log).
Le temps d'exécution des tâches (paragraphe 4.4.4)	Si la priorité est égale à 9 ou 10 et que la « scheduling class » est égale à 2, alors les tâches dureront longtemps.	La priorité et la sensibilité à la latence influent sur la durée d'une tâche.
Le pourcentage CPU utilisé (paragraphe 4.4.5)	La moyenne de CPU est la plus grande pour les couples « 9_3 » « 9_2 » et pour le couple « 6_2 ». Les tâches qui terminent par un « Kill », consomment en moyenne le double de CPU.	Si les tâches finissent anormalement (« Kill », « Evict », « Fail »), alors en moyenne, les tâches consomment plus de CPU. Si les priorités et la sensibilité à la latence sont élevées, alors la tâche consomme plus de CPU.
La QoS (paragraphe 4.4.6)	Les tâches ayant une priorité élevée et un « scheduling class » égale à 2, possède un temps d'attente très faible pour passer de l'état « Submit » à « Schedule ».	De manière générale, les traces Google ont une QoS très élevée.

Le tableau 4.5 met en avant l'hétérogénéité de la charge de travail parce qu'elle propose un ensemble de tâches ayant à la fois des temps d'exécution long et court, le manque d'indicateurs pour la consommation d'énergie, et de surcroît, l'aspect énergie verte et l'existence de deux types de services : les services de types application (longs et gourmands en terme de CPU) et les services de types Batch et log. Par contre Google, ne met pas en avant les tâches de type sauvegarde.

Dans les traces synthétiques il convient de :

- garder l'hétérogénéité des tâches (tâches longues, tâches courtes) ;
- avoir plus de tâches courtes que de tâches longues ;
- ajouter un indicateur pour calculer la consommation des tâches. Les données Google étant cachée, il faut donc rajouter un poids aux différentes tâches afin calculer la consommation.

La partie suivante présente la modélisation des traces synthétiques à partir du modèle de Google.

4.5 Modélisation des données Google pour leur implémentation dans RenewSim : conception des traces synthétiques

Afin d'utiliser les traces proposées par Google dans le simulateur, une épuration s'est imposée. Nous nous sommes intéressés aux données stockées dans le répertoire « task_events ». Il contient 500 fichiers csv. Afin de rejouer les données Google dans notre simulateur, nous avons utilisé certaines caractéristiques des tâches Google. Le simulateur place des VMs sur des machines physiques. Une VM est définie comme suit :

Dans le tableau 4.6, les attributs *schedule_time* et *ending_time* nous permettent de calculer le « makespan » (le temps d'exécution). L'attribut *ide* est un identifiant unique obtenue par *job ID*et *task index -within the job* (section 4.6) des caractéristiques Google. Ensuite, les attributs suivants ont été calculés en fonction du cycle de vie de la tâche à trois événements présenté en section 4.6) :

- *tps_soumission* : le *timestamp* de l'événement « SUBMIT » ;
- *tps_debut_exe* : le *timestamp* de l'événement « SCHEDULE » ;
- *tps_fin_exe* : le *timestamp* d'un événement de type fin ;
- *événement_de_fin* : correspond à « Finish »,« Kill » et « Fail » (*event type*)

Les attributs de priorité et de latence sont respectivement obtenus par les attributs déjà présents dans les traces initiales, *priority* et *scheduling class*.

Tableau 4.6 – Description d'une VM dans RenewSim

Nom	**Description**
ide	un identifiant unique qui nous permet de suivre l'évolution d'une VM
tps_soumission	le temps d'arrivée de la VM dans la pile de VM à exécuter, ce temps nous permet de savoir quand la VM est éligible pour être ordonnancée
tps_debut_exe	le temps où la VM est ordonnancée, c'est-à-dire le moment où la VM s'exécute sur le serveur
tps_fin_exe	le temps où la VM est terminée
événement_de_fin	événement de fin. Il peut être de type : « Evict »,« Fail », « Finish », « Kill »
priorité	sa priorité
latence	sa sensibilité à la latence
VCPU	demande VCPU de la VM

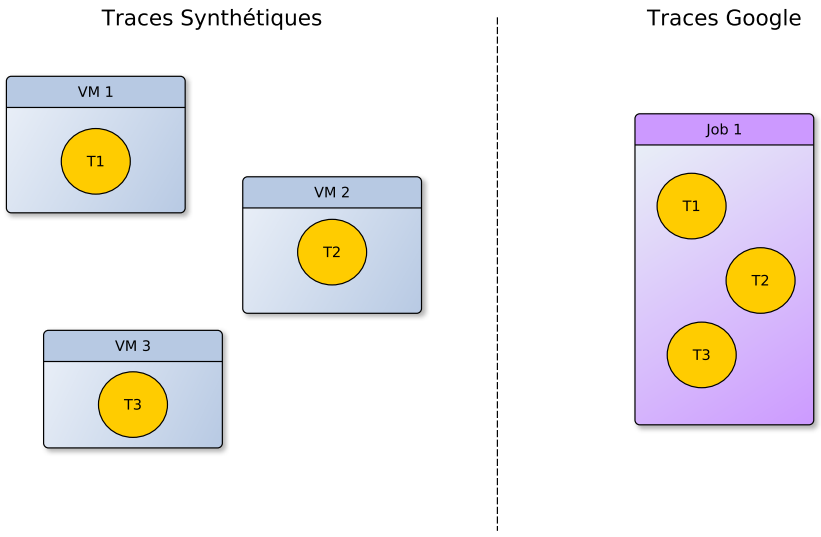


Figure 4.21 – Différence entre les VMs des traces synthétiques et les Jobs et Tâches présente dans les traces Google

La figure 4.21 montre la différence entre une VM et les tâches et les jobs Google. Une tâche est déliée de son job et est associée à une VM. Cela nous permet de calculer la consommation via l'attribut VCPU et d'estimer la consommation d'une seule tâche afin d'être précis dans les algorithmes d'ordonnancement liés à la consommation d'énergie verte. Cela permet de dissocier chaque exécution pour contrôler au mieux la consommation d'énergie.

La signification des événements proposée par Google est liée à l'ordonnanceur. Google ayant caché pour des raisons de sécurité le fonctionnement, on n'a pas assez d'information pour les types « fail/kill/evict ». Elles sont conservées dans les traces synthétiques car dans le futur ces tâches peuvent réapparaitre sous un autre identifiant et cela permet de garder une classification des types de tâches (services applicatifs, ou services batch ou log).

Renewsim est un simulateur polymorphe donc utilisable avec des algorithmes d'ordonnancement ou de placement différents. Les attributs liés à la latence et la priorité peuvent être utilisés afin de déterminer quand et où placer une tâche, par rapport aux ressources disponibles. De plus, d'après l'analyse statistique, le poids des priorités influent par exemple sur la durée d'une tâche. Cela permet donc de tester des jeux de données ciblés pour connaître le comportement de l'ordonnanceur. De plus, cela peut permettre à l'ordonnanceur de prendre certaines décisions. Par exemple, définir des ressources pour des tâches avec une priorité égale à « 4 » ou bien, gérer les conflits entre plusieurs tâches arrivant dans la file d'attente. La sensibilité à la latence est conservée dans les traces synthétiques car cette sensibilité permet comme au priorité d'avoir un rôle de levier dans la prise de décision par l'ordonnanceur. Elle apporte en plus la notion de QoS.

Dans ces traces synthétiques la notion de VCPU a été introduite afin de gérer les différentes ressources physiques et de calculer la consommation. Une VM doit être placée sur un serveur dont la capacité VCPU est disponible durant tout le temps où la VM s'exécute.

4.6 Conclusion et limites

Ce chapitre a permis d'analyser des traces réelles à grande échelle de créer des traces synthétiques qui ont vocation de validation dans un environnement expérimental. Les traces générées ont atteint les objectifs suivants :

- de la capacité d'adaptation à une infrastructure hétérogène ;
- de mesurer la consommation d'un serveur ;
- de quantification de la QoS à travers les attributs des VMs ;
- d'implémentation de ces traces dans l'infrastructure matérielle de RenewSim (chapitre 3).

Dans la partie 4.4, l'analyse des traces Google met en avant l'abondance des d'événements pour les tâches, et la normalisation des données (CPU, RAM et Disque). Cependant, la description des tâches permet de connaître les temps d'arrivée et d'exécution d'une tâche qui

sert à calculer le temps d'attente d'une tâche, et ainsi connaître la QoS. Cette analyse permet de mettre en avant l'hétérogénéité de la charge de travail parce qu'elle propose un ensemble de tâches ayant à la fois des temps d'exécution long et court. Cependant, cette charge de travail ne ne prends pas en compte la consommation d'énergie et par surcroît l'aspect énergie verte.
Dans la partie 4.5, les traces synthétiques sont proposées. Elles permettent de garder l'hétérogénéité observée dans les traces réelles. Afin de palier aux données normalisées concernant l'utilisation des ressources, la notion de VCPU a été rajoutée. De plus, à travers l'étude de ces traces, nous pouvons mettre en avant le contexte applicatif de ces traces et générer des charges de travail plus génériques et indépendantes de la notion de consommation [87].

Les limites de cette génération sont les pertes de notions pour les services composés permettant de calculer la consommation d'une charge à un instant donné. Cependant, la VM encapsule l'idée de composition pour de futures traces.

Dans cette partie, l'analyse des traces Google permet de les utiliser dans le simulateur RenewSim. Les expérimentations réalisées dans le chapitre 6 seront donc réalisées en implémentant ces traces synthétiques dans RenewSim. Le chapitre 5 suivant présente un ensemble d'algorithmes d'ordonnancement. Ils ordonnancent des VMs, possédant les caractéristiques décrites dans ce présent chapitre.

Chapitre 5

Ordonnancement sous contraintes : maximiser l'utilisation de l'énergie renouvelable

Sommaire

De nos jours, les systèmes de nuages informatiques consomment de plus en plus d'énergie. L'économie d'énergie est un objectif visé par une grande partie des entreprises pour réduire l'empreinte carbone grâce à l'énergie verte via des panneaux solaires ou éoliennes. Contrairement à l'approvisionnement classique en énergie sur le réseau électrique, la production d'énergie verte est instable et dépend de la nature (les conditions météorologiques). Cela introduit de nouveaux défis en tant que charge de travail adaptable dans le temps et l'espace pour réduire la consommation d'un serveur.

Dans le chapitre précédent, nous avons analysé des traces réelles à grande échelle et créé des traces synthétiques qui ont permis de caractériser une charge de travail pour un centre de calcul. Dans ce chapitre, nous concevons un ordonnancement de charge de travail qui utilise une prédiction de production d'énergie solaire pour prendre une décision. Cela nous permet d'ordonnancer des machines virtuelles dans un centre de calcul en raison de différents algorithmes qui consomment le moins d'énergie non renouvelable possible. L'ordonnanceur vise à utiliser toute l'énergie verte et à réduire la consommation du centre de données en minimisant l'impact sur la QoS. Nous avons utilisé des algorithmes d'optimisation sous contraintes et nous les avons adapté à nos besoins. Ils permettent d'ordonnancer une charge de travail sur un centre de calcul en fonction de la production solaire. Afin d'évaluer ces algorithmes un ensemble de métriques est présenté.

Tout d'abord, l'approche d'ordonnancement sous contraintes et les objectifs sont présentés. Ensuite sont énumérées les spécifications d'un tel ordonnanceur. Puis sont introduits notre approche et les différents algorithmes sous contraintes modélisés. Afin d'évaluer la QoS et l'impact sur l'environnement de ces algorithmes plusieurs métriques existantes sont analysées. Enfin, dans une dernière partie sont exposés le bilan et les limites de ces algorithmes.

5.1 Approche par ordonnancement sous contraintes

5.1.1 La création d'une charge de travail agile

Notre ordonnanceur vise à modeler une charge de travail pour la rendre adaptable dans le temps et dans l'espace. Nous la qualifions d'agile. La figure 5.1 présente un exemple de charge de travail agile. La corrélation entre la charge de travail et la production de panneaux solaires vise à réduire l'achat chez le fournisseur d'électricité et à avoir un centre de calcul presque autosuffisant.

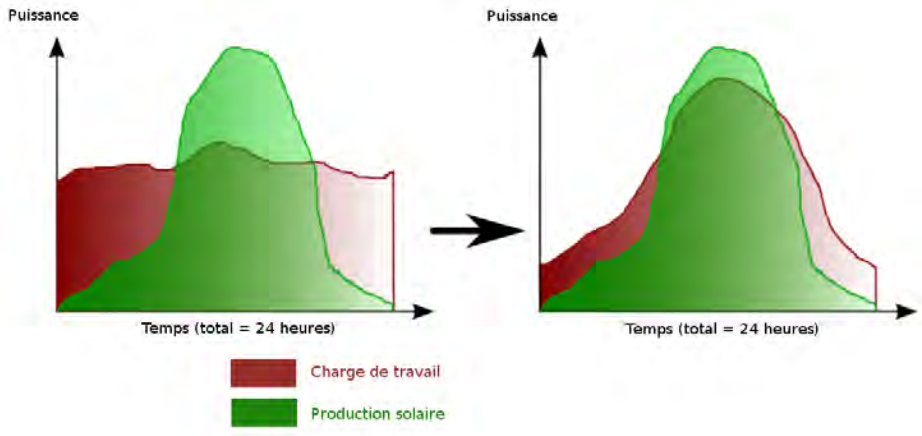


FIGURE 5.1 – un exemple d'une charge de travail agile qui suit la courbe de production solaire

Le verrou principal avec les énergies renouvelables est la production d'énergie non constante car celle-ci dépend de la météo. Les défis sont donc de respecter les contraintes rencontrées pour :

- les exigences d'ordonnancement et de demande d'énergie pour les VMs ;
- satisfaire la demande des utilisateurs ;
- optimiser et rendre les serveurs plus rentables en fonction du temps, de l'énergie disponible et de leurs exigences.

A travers la sauvegarde d'énergie dans des espaces de stockage (batteries), il est possible : d'utiliser d'une meilleure façon l'énergie renouvelable, de rentabiliser l'énergie produite (ventes) ou de réduire la consommation, les coûts d'achat via des algorithmes sous contraintes. Dans ce chapitre est présentée une approche d'ordonnancement pour rapprocher faire coïncider la charge de travail à la production solaire (figure 5.1). Cela permettra réduire l'achat d'électricité chez le fournisseur tout en respectant un bonne QoS et tendre vers un centre de calcul auto-suffisant.

5.1.2 Utilisation d'algorithmes gloutons

Dans ce chapitre est décrit un ensemble d'algorithmes appartenant à la famille des gloutons [41, 43]. Leurs fonctionnements diffèrent des algorithmes génétiques ou des programmes linéaires car ils n'intègrent pas l'optimisation directe d'objectifs à réaliser. Les avantages des algorithmes gloutons sont la création de schémas heuristiques simples et rapides. Les décisions réalisées par l'algorithme sont prises pas à pas, sans jamais revenir sur la décision précédente. Dans les algorithmes, la prise de décision se fait rapidement. Le choix d'utilisation s'est porté sur les algorithmes gloutons car une décision doit émerger de manière rapide, afin de satisfaire le plus rapidement possible la demande de l'utilisateur. L'utilisation d'énergie verte n'est pas pérenne dans le temps. Nous insistons sur le fait que l'avantage des algorithmes gloutons est qu'ils prennent une décision pas à pas sans jamais revenir sur celle-ci. Les décisions prisent par un algorithme génétique sont prises en fonction des autres, or, si les conditions météorologiques divergent et que la production solaire est affaiblie, alors les décisions prises sur cette fenêtre temporelle ne seront plus optimales et n'auront plus de sens.

5.2 Spécifications des algorithmes

5.2.1 Contexte

Les algorithmes présentés dans ce chapitre sont issus de la littérature. Ils ont été adaptés à nos besoins et ils ont pour objectifs d'être implémentés dans le simulateur RenewSim et plus particulièrement, en tant qu'ordonnanceur dans le module de contrôle. Cet ordonnanceur doit avoir accès aux différentes informations et flux qui circulent dans le réseau intelligent. Il ne prend pas seulement une décision en fonction des activités informatiques liées au centre de calcul mais

aussi aux disponibilités énergétiques disponibles. C'est-à-dire, il prend en compte la production d'énergie renouvelable, et la quantité disponible dans les espaces de stockage.

5.2.2 Objectifs

Notre objectif principal est de fournir de manière plus efficace un centre de calcul avec de l'énergie verte à travers un ordonnanceur se basant sur une prédiction de production. Ce centre de calcul est alimenté simultanément par des sources d'énergie renouvelables et non renouvelables. Le problème d'ordonnancement pour alimenter un centre de calcul est l'utilisation d'énergie verte via des panneaux solaires, dont la production dépend de la nature, variable en fonction de la météorologie.
La principale source provient de la production de panneaux solaires et la principale destination est le centre de calcul. Ce réseau intelligent est supervisé par un module de contrôle. Le rôle est de trouver la meilleure configuration possible pour la gestion du flux d'énergie (alimentation du centre de calcul) et la charge de travail demandée par le centre de calcul. Cela consiste à gérer l'énergie vendue des panneaux solaires aux fournisseurs, l'énergie achetée au fournisseur en fonction de son coût ainsi que la charge et décharge de la batterie. Le but est donc de doter le module de contrôle d'un ordonnanceur afin de placer dans le temps la charge de travail. Cette charge de travail est constituée d'un ensemble de VMs (chapitre 4). L'attribution des VMs dépend de leurs caractéristiques (CPU et capacité de mémoire, priorité, préemptive ou non) et de la quantité totale d'énergie disponible.
Les algorithmes d'ordonnancement doivent permettre :

- d'être le moins dépendant possible du fournisseur d'électricité ;
- de diminuer l'empreinte écologique ;
- de réduire la facture d'électricité.

5.2.3 Les fonctionnalités attendues des algorithmes

Les algorithmes doivent respecter plusieurs éléments de méthodes ou contraintes. Dans un premier temps, il faut respecter la capacité VCPU disponible des différents serveurs composant le centre de calcul. Les serveurs ne peuvent héberger une certaine limite de VMs à la fois. Dans un deuxième temps, il faut respecter la prédiction de production solaire. Une VM est placée sur un serveur que quand elle peut être alimentée via de l'énergie solaire. Dans un troisième temps, ces algorithmes doivent pouvoir proposer des résultats et des simulations à l'aide de RenewSim (chapitre 3) qui repose sur une architecture qui mélange les deux domaines complexes, les flux d'énergie et les flux de type informatique. Dans un quatrième temps, les différents algorithmes doivent être mesurables via des métriques évaluant la QoS et l'empreinte écologique afin de les comparer et de les évaluer.

5.3 Notre approche d'ordonnancement

5.3.1 Stratégie d'ordonnancement

Le tableau 5.1 présente toutes les possibilités de stratégie. Les lignes du centre de calcul et des panneaux solaires n'ont aucune valeur, car certaines situations ne peuvent pas se produire. Par exemple, le panneau solaire ne peut pas recevoir d'énergie, donc il ne peut pas stocker d'énergie et être de type *Sauvegarde*, *Trop-plein* et *Destination*. Le centre de calcul ne peut pas générer d'énergie (*Source*) et stocker de l'énergie (*Sauvegarde* et *Trop-plein*), il ne peut consommer d'énergie (*Destination*).

TABLEAU 5.1 – Représentation des éléments et l'ensemble des valeurs possibles pour chaque élément du réseau intelligent, via le modèle de flux d'énergie présenté dans le chapitre 3

types / **éléments**	**Source**	**Destination**	**Sauvegarde**	**Trop-plein**
Énergie renouvelable 1	$\{\times, 1\}$	$\times$	$\{\times, 1, 2, ...\}$	$\times$
Énergie renouvelable 2	$\{\times, 1\}$	$\times$	$\{\times, 1, 2, ...\}$	$\times$
Centre de calcul	$\times$	$\{\times, 1\}$	$\times$	$\times$
Espace de stockage 1	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
Espace de stockage 2	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
Fournisseur	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
...	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$
élément n	$\{\times, 1\}$	$\{\times, 1\}$	$\{\times, 1, 2, ...\}$	$\{\times, 1, 2, ...\}$

Notre stratégie d'ordonnancement utilise le tableau 5.2 comme modèle de données. Ce tableau 5.2 met en évidence l'ordre d'utilisation des différentes sources d'énergie. Les panneaux solaires fournissent le centre de calcul en tant que sources. En cas de sous-production, la batterie (Sauvegarde = 1) alimente le centre de calcul. Si la batterie est vide, le fournisseur électrique alimente le centre de calcul (Sauvegarde = 2). En cas de sur-production, le surplus d'énergie est stocké dans la batterie (Trop-plein = 1). Si la batterie est pleine, l'énergie est vendue au fournisseur d'électricité (trop-plein = 2). Ce modèle vise à acheter le moins d'énergie possible auprès du fournisseur. Notre stratégie peut être représentée par la figure 5.2.

TABLEAU 5.2 – Modèle énergétique pour la stratégie d'ordonnancement

	Source	**Destination**	**Sauvegarde**	**Trop-plein**
Panneaux solaires	1	×	0	×
Centre de calcul	×	1	×	×
Batterie	0	0	1	1
Fournisseur	0	0	2	2

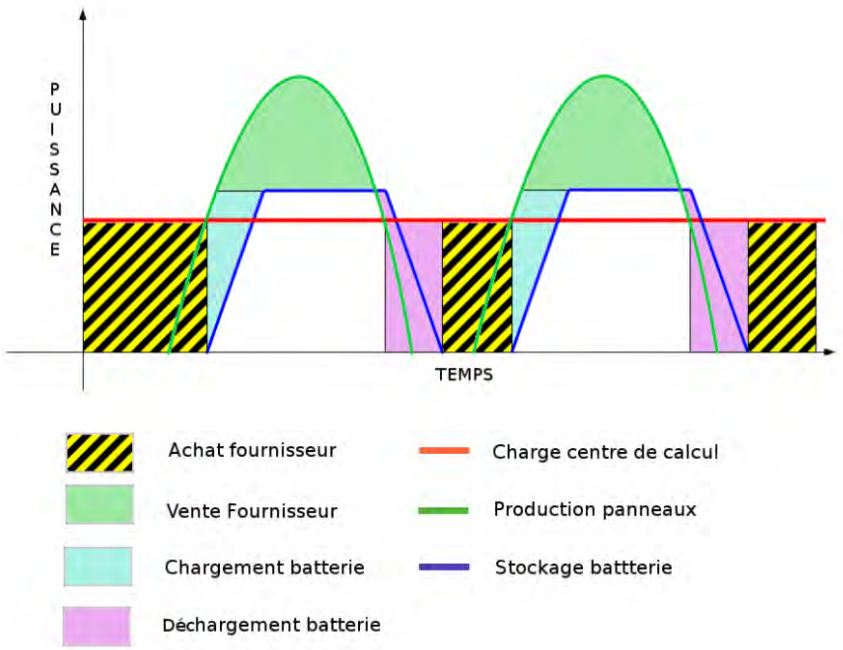


FIGURE 5.2 – L'utilisation des différentes sources d'énergies pendant 48 heures

La figure 5.3 est un exemple dans une situation presque parfaite. Les VMs sont programmées en fonction de l'heure, de l'énergie disponible dans la production solaire et de l'espace disponible dans les serveurs. Le stockage (batterie) n'est pas une contrainte d'ordonnancement des algorithmes mais elle fait partie de la stratégie d'ordonnancement de la gestion des flux d'énergie. Il s'agit d'une solution de sauvegarde, seulement si la production des panneaux solaires n'est pas suffisante pour alimenter le centre de calcul.

Les contraintes d'ordonnancement des VMs sont :

- l'utilisation de serveurs hétérogènes en termes de VCPU ;
- le placement de VMs hétérogènes en termes de temps d'exécution ;
- le nombre de VCPU utilisés par VM ;
- l'énergie disponible issue de la production des panneaux solaires.

L'avantage de cette méthode est l'utilisation maximale de la production des panneaux solaires et la réduction de l'achat d'énergie chez le fournisseur d'électricité. La VM n'est ordonnancée que s'il y a suffisamment d'énergie produite par les panneaux solaires. Cependant, les VMs continuent de fonctionner même si l'énergie solaire est nulle. Par exemple, dans la figure 5.3, la *Vm 10* commence à s'exécuter lorsqu'il y a assez d'énergie solaire sur le serveur *S3*. Cependant, cette VM continue de s'exécuter lorsque la production solaire diminue.

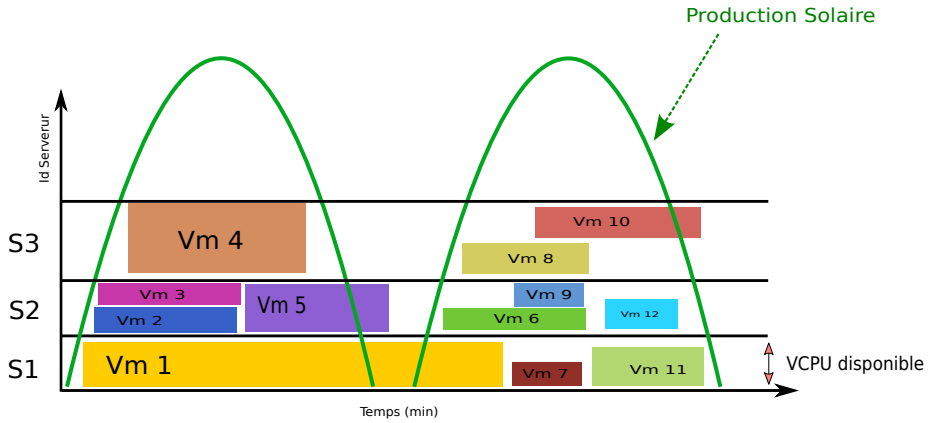


FIGURE 5.3 – Un exemple d'ordonnancement de VMs en fonction de la production solaire

5.3.2 Proposition d'adaptation d'algorithmes existants

Dans cette partie sont présentées et analysées plusieurs solutions d'ordonnancement. Pour chacun des algorithmes suivants, les contraintes de ressources de différents serveurs et l'énergie solaire disponibles au fil du temps sont toujours prises en compte. Pour chacun des algorithmes suivants est ordonnancé un ensemble de VMs indépendantes, stockées dans une liste d'attente.

La figure 5.4 présente les différentes entrées et sorties des algorithmes décrits dans cette partie.

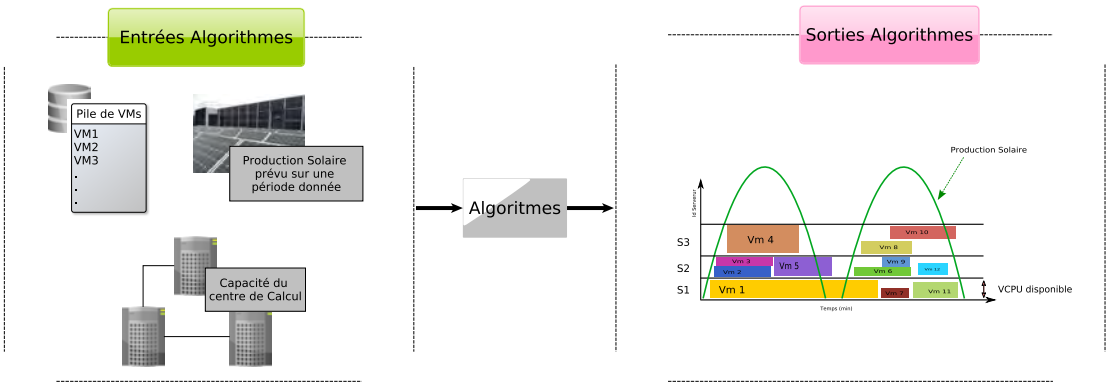


FIGURE 5.4 – Entrées et sorties des différents algorithmes d'ordonnancement

Round Robin [41] (RR)

L'algorithme RR permet de placer une VM sur un serveur puis une fois la VM placée, la VM suivante sera placée sur le serveur suivant et ainsi de suite. Dans ce cas, si une VM ne peut être placée sur un serveur, le suivant est sélectionné. Si elle ne peut être placée sur l'un des serveurs du centre de calcul, elle est éjectée et la VM suivante est sélectionnée. La charge des serveurs est plus ou moins équilibrée dans le temps. L'inconvénient est qu'il n'y a pas d'optimisation du parc informatique en termes de serveurs allumés, il y a une sous-utilisation des ressources. Cet algorithme a pour but de servir de référence.

Algorithme 3 RR

while VM < nbVmsOrdonnancees **do**
  for `tps=0;tps<fenetreOrdonnancement;tps++` **do**
    if $energieSolaire[tps] == vrai$ and $serveurs[s][tps] == vrai$ **then**
      $VM \Rightarrow serveur[s][tps]$
      $serveurs[s+1]$
    end if
  end for
end while

First Fit (FF)

L'algorithme FF consiste à placer les différentes VMs sur un serveur. Une fois ce dernier plein, l'opération se répète sur le serveur suivant. L'inconvénient est la charge des différents serveurs. Les premiers serveurs auront une charge plus importante que les derniers serveurs car l'énergie solaire disponible aura fortement diminuée par les placements déjà effectués.

Algorithme 4 FF

```
while VM < nbVmsOrdonnancees do
  for tps=0;tps<fenetreOrdonnancement;tps++ do
    if energieSolaire[tps] == vrai and serveurs[s][tps] == vrai then
      VM ⇒ serveur[s][tps]
      if serveurPlein == vrai then
        serveurs[s + 1]
      end if
    end if
  end for
end while
```

Round Robin [41] avec des classes de serveurs (RR_CLASSES)

L'algorithme RR_CLASSES reprend l'algorithme (RR) sauf qu'il contient des classes de serveurs. Un ensemble de serveurs est dédié à l'ordonnancement de VMs courtes, moyennes ou longues. Par exemple, 10% des serveurs ordonnancent les VMs longues, 40% les VMs moyennes et 50% les VMs courtes. Cette division en classes, permet de répartir un ensemble de VMs, et d'ordonnancer plus de VMs dans le temps. Cela permet aux VMs longues d'obtenir plus facilement des ressources VCPU qui peuvent être utilisées par des VMs de durée moins longues et donc empêcher l'ordonnancement de certaines VMs plus gourmandes. L'inconvénient de cet algorithme est l'optimisation des différentes ressources du parc informatique. Par exemple, une VM longue, qui ne peut plus être placée sur un des serveurs appartenant à la classe VMs longues, sera éjectée, même s'il reste de la place sur un serveur permettant d'ordonnancer des VMs courtes ou moyennes. Les classes serveurs doivent être implémentées en fonction du nombre de différentes VMs à exécuter, pour trouver la meilleure répartition de classes serveurs.

Algorithme 5 RR_CLASSES

```
while VM < nbVmsOrdonnancees do
  for tps=0;tps<fenetreOrdonnancement;tps++ do
    if energieSolaire[tps] == vrai and serveurs[s][classe][tps] == vrai then
      VM ⇒ serveur[s][tps]
      serveurs[classe][s + 1]
    end if
  end for
end while
```

Round Robin et Longest Processing Time (RR_LPT)

Cet algorithme RR_LPT consiste à ordonnancer une VM sur un serveur, puis une fois que la VM est placée, la prochaine VM sera placée sur le serveur suivant et ainsi de suite. Cet algorithme

diffère de l'algorithme RR car les VMs sont ordonnancées en fonction de leur taille. Les VMs les plus longues sont ordonnancées en premier.

Algorithme 6 RR_LPT

Require: liste de VMs triées par temps d'exécution (des plus longues au plus courtes)
  while VM < nbVmsOrdonnancees **do**
    for `tps=0;tps<fenetreOrdonnancement;tps++` **do**
      if $energieSolaire[tps] == vrai$ and $serveurs[s][tps] == vrai$ **then**
        $VM \Rightarrow serveur[s][tps]$
        $serveurs[s+1]$
      end if
    end for
  end while

First Fit et Longest Processing Time (FF_LPT)

L'algorithme FF_LPT consiste à placer les différentes VMs sur un serveur. Une fois ce dernier plein, on répète l'opération sur le serveur suivant. Cet algorithme diffère de l'algorithme FF car les VMs sont placées en fonction de leur taille. Les VMs les plus longues sont placées en premier.

Algorithme 7 FF_LPT

Require: liste de VMs triées par temps d'exécution (des plus longues au plus courtes)
  while VM < nbVmsOrdonnancees **do**
    for `tps=0;tps<fenetreOrdonnancement;tps++` **do**
      if $energieSolaire[tps] == vrai$ and $serveurs[s][tps] == vrai$ **then**
        $VM \Rightarrow serveur[s][tps]$
        if $serveurPlein == vrai$ **then**
          $serveurs[s+1]$
        end if
      end if
    end for
  end while

5.4 Métriques pour l'évaluation des algorithmes

Afin d'évaluer les différents algorithmes et les comparer entre eux, cette section présente un ensemble de métriques. Dans une première partie sont introduites des métriques pour évaluer la consommation dite verte. Dans une deuxième partie sont présentées des métriques pour évaluer la QoS de l'ordonnancement.

5.4.1 Métriques vertes

Il existe plusieurs mesures vertes, pour mesurer à quel point un centre de calcul est vert. L'efficacité d'utilisation de la puissance (PUE) [44] est une mesure pour déterminer l'efficacité énergétique d'un centre de calcul. C'est la méthode la plus populaire pour calculer l'efficacité énergétique. Par exemple, Google l'utilise pour évaluer l'efficacité de ses centres de calcul, en 2017, depuis 2013, il est à environ 1.12 [88]. Cela consiste à évaluer la quantité d'énergie totale consommée par un site sur un an, par rapport à la quantité d'énergie requise pour l'exploitation du matériel informatique. Cependant, les travaux présentés dans cette thèse ne portent pas sur les technologies de refroidissement dans un centre de calcul. Le coefficient d'énergie vert (GEC) [89] permet de quantifier la part d'énergie renouvelable consommée par un centre de calcul. Comme le PUE, il est introduit par l'organisation "Green Grid" en novembre 2012 [90], afin de palier aux inconvénients du PUE, car ce dernier ne prend pas en compte certains éléments tels que la géolocalisation du centre de calcul, ou le taux de charge des différents composants du centre de calcul. Pour le calculer, il faut diviser la quantité d'électricité consommée provenant des sources renouvelables par la consommation totale du centre de calcul. L'équation 5.4.1 propose de calculer le GEC dans notre cas, où $Prod_{solaire}$ toute l'énergie verte générée dans le centre de calcul, $Prod_{vend}$ toute l'énergie verte non utilisée et vendue au fournisseur d'électricité, et $Fournisseur_{achat}$ toute l'énergie achetée chez le fournisseur.

Proposition 1 : Le calcul du GEC s'écrit :

$$GEC = 0 \leqslant \frac{Prod_{solaire} - Prod_{vend}}{Fournisseur_{achat} + (Prod_{solaire} - Prod_{vend})} \leqslant 1 \tag{5.4.1}$$

Nous choisissons de retirer la vente de surplus de l'énergie verte ($Prod_{vend}$) car elle ne fournit jamais le centre de calcul. Cela ne fait pas partie de la quantité d'énergie requise pour le fonctionnement du matériel informatique. Si le résultat du GEC est plus proche de 1, cela signifie une plus grande utilisation de l'énergie verte et plus proche de 0, l'architecture utilise plus d'énergie non renouvelable que d'énergie verte.

5.4.2 Métriques de QoS

Plusieurs travaux ont abordé la définition de la QoS dans un environnement distribué [91] ainsi que, la QoS liée à l'énergie [39, 92]. Les centres de calcul doivent être capables de délivrer un niveau de performance pour les utilisateurs. Un ensemble de métriques peut être utilisé pour évaluer la qualité dans un environnement de type nuage. Certaines métriques, comme celles présentées dans [93], permettent d'évaluer l'élasticité d'un nuage, à travers les ressources utilisées, les coûts d'approvisionnements et de sous approvisionnement. Rossi *et al.* [94] présentent *Orchestrator (e-eco)*, permettant d'améliorer les économies d'énergies et les performances des applications dans

un nuage. Cet indicateur se base, aussi, sur un ensemble de métriques permettant d'évaluer les ressources, les consolidations, ou encore la virtualisation des VMs. Par exemple, à travers leur simulation, les indicateurs SLA violation/saturation sont très inférieurs ($\simeq$ 30 pts) par rapport à d'autres méthodes telles que Alvarruiz *et al.* [95].

Pour évaluer la QoS des différents algorithmes présentés, il a été choisi de comparer le taux de VMs exclues et ordonnancées. Cela vise à montrer la capacité du centre de calcul à répondre à la demande de l'utilisateur et la capacité du centre de calcul à s'adapter à la charge de travail. Ces valeurs sont fortement liées à la qualité ressentie par les clients et par conséquent, liées aux bénéfices que peut réaliser l'opérateur du centre de calcul. De plus, pour comparer les algorithmes, nous avons introduit un indicateur de temps d'achèvement (Équation 5.4.2). Il correspond à la somme du temps d'exécution (de son ordonnancement à sa fin d'exécution) pour chaque VM ordonnancée et chaque VM exclue.

Proposition 2 : La quantité horaire de VMs éjectées/ordonnancées s'écrit :

$$\sum_{i=1}^{n}(comp_t^{vm_i}) \tag{5.4.2}$$

où n le nombre de VMs exclues ou ordonnancées et $comp_t$ le temps de fin de la VM vm_i. Cet indicateur aide à comprendre les VMs exclues, pour savoir s'il y a beaucoup de VMs longues ou courtes et son impact sur l'ordonnanceur.

5.4.3 Exemples d'utilisation des métriques

L'équation 5.4.3 est un exemple d'utilisation du GEC lorsqu'il obtient un bon taux. Si les panneaux solaires produisent 100 joules en 48 heures et la vente au fournisseur est égale à 10 joules et l'achat est égal à 20 joules, le GEC sera égal à 0,82.

$$\frac{100-10}{20+(100-10)} = \frac{90}{110} = 0.82 \tag{5.4.3}$$

L'équation 5.4.4 est un autre exemple d'utilisation du GEC, lorsqu'il obtient un mauvais taux. Si les panneaux solaires produisent 80 joules en 48 heures et la vente au fournisseur est égale à 40 et l'achat est égal à 40 joules, le GEC sera égal à 0.50. Dans ce cas là, il y a eu plus de vente au fournisseur que dans l'exemple précédent 5.4.3. Cependant, le GEC obtient un moins bon taux car l'achat est plus important. Cela peut s'expliquer par la période d'utilisation de l'énergie. Si les VMs sont ordonnancées lorsque la production solaire décline, elles vont continuer à s'exécuter lorsqu'il n'y aura plus d'énergie et donc le déstockage de la batterie va commencer et une fois celle-ci déchargée, l'achat au fournisseur sera réalisé. Donc, le but des algorithmes n'est pas de vendre au fournisseur et de stocker au maximum dans la batterie mais d'utiliser au mieux

l'énergie renouvelable pour alimenter le centre de calcul.

$$\frac{80-40}{40+(80-40)} = \frac{40}{80} = 0.50 \tag{5.4.4}$$

5.5 Conclusion et limites

Dans ce chapitre, nous avons adapté un ensemble d'algorithmes pour répondre aux objectifs suivants :

- créer un ordonnancement basé sur de la prédiction d'énergie verte ;
- être le moins dépendant possible du fournisseur d'électricité ;
- pouvoir évaluer l'ordonnancement via des métriques.

Nous avons choisis la famille des gloutons parce que l'utilisation d'énergie verte n'est pas pérenne dans le temps et cela nous permet d'obtenir une prise de décision de manière rapide. Ces différents algorithmes possèdent les mêmes types d'entrées afin de les rendre comparables et mesurables entre eux. Nous avons définis nos indicateurs de QoS qui jugent de la conformité du centre de calcul à allouer un nombre suffisant de ressources aux VMs. Nous avons adapté un indicateur pour évaluer le pourcentage d'énergie verte utilisé dans le centre de calcul. Les contraintes des algorithmes sont les mêmes, et correspondent à la place disponible en termes de VCPU pour chaque serveur et l'énergie produite disponible. Ces algorithmes nous permettent d'obtenir une charge de travail agile et modulable.

Cependant, ces algorithmes ont besoin de prédictions météorologiques afin de réaliser un ordonnancement. Actuellement, l'ordonnancement ne s'adapte pas en fonction de la production réelle. Il est uniquement basé sur les prédictions de production solaire réalisée. De plus, les VMs continuent de s'exécuter même s'il n'y a plus de production solaire à cet instant t. Une amélioration possible serait de suspendre le travail d'une VM et attendre que l'énergie revienne. Une deuxième solution est la migration en temps réel [54] vers un autre serveur afin de diminuer la consommation totale du centre de calcul et de surcroît permettre un gain d'énergie. De plus, dans ces algorithmes, il n'y a pas de prédiction de la charge de la batterie au cours du temps pour optimiser leur utilisation. Cela pourrait permettre de l'utiliser plus et non de l'utiliser au minimum, afin d'acheter moins d'électricité provenant du fournisseur.
Le chapitre 6 suivant présente les différentes expérimentations des contributions de cette thèse. Elles sont réalisées à l'aide du simulateur présenté dans le chapitre 3.

Chapitre 6

Expérimentations : évaluations et comparaisons

Sommaire

Ce chapitre détaille les différentes expérimentations sur l'évaluation des différents modèles et algorithmes. Le simulateur RenewSim présenté dans le chapitre 3 a permis d'obtenir les différents résultats. À travers son module de contrôle, il est exécute un certain nombre de politiques de décisions afin de gérer les réserves énergétiques et diminuer les coûts de consommation. Les traces synthétiques présentées dans le chapitre 4 sont une entrée de ce simulateur. Elles sont utilisées afin de pouvoir réaliser la simulation d'ordonnancement des algorithmes présentés dans le chapitre 5. Cette phase d'évaluation par simulation expérimente l'ensemble des travaux présentés dans les chapitres précédents. L'évaluation par simulation amène également la possibilité d'étudier la corrélation entre la taille de la batterie et la production solaire. Cela permet de permet de connaître la corrélation entre les deux sources énergétiques dites vertes (batterie, panneaux solaires) pour obtenir un centre de calcul, le moins consommateur d'énergies dites non renouvelables.

Ce chapitre présente les résultats expérimentaux et se divise en cinq parties. La première partie 6.1 spécifie le réseau intelligent, en terme de modèles et des différents éléments qui le compose. La seconde partie 6.2 évalue différentes politiques de décisions et l'implémentation du modèle de décisions présenté dans le chapitre 3. La troisième partie 6.3 évalue les différents algorithmes présentés dans le chapitre 5 à travers un ensemble de métriques présentées dans la partie 5.4. La quatrième partie 6.4 est une étude sur le dimensionnement des éléments du réseau intelligent, afin d'établir une corrélation entres eux et leurs influences sur le rendement énergétique vert. Enfin, la partie 6.5 dresse un bilan des différentes expérimentations réalisées.

6.1 Spécifications du simulateur pour les différentes expérimentations réalisées

Dans cette section 6.1 est présenté l'environnement de simulation des différentes expérimentations. Dans un premier temps est spécifiée la structure du centre de calcul puis les caractéristiques des différents éléments qui le compose.

6.1.1 Composition du réseau intelligent

Le réseau intelligent proposé pour les différentes simulations et expérimentations suivantes est composé d'un parc photovoltaïque, d'une liaison vers le fournisseur d'électricité, d'une batterie et d'un centre de calcul.

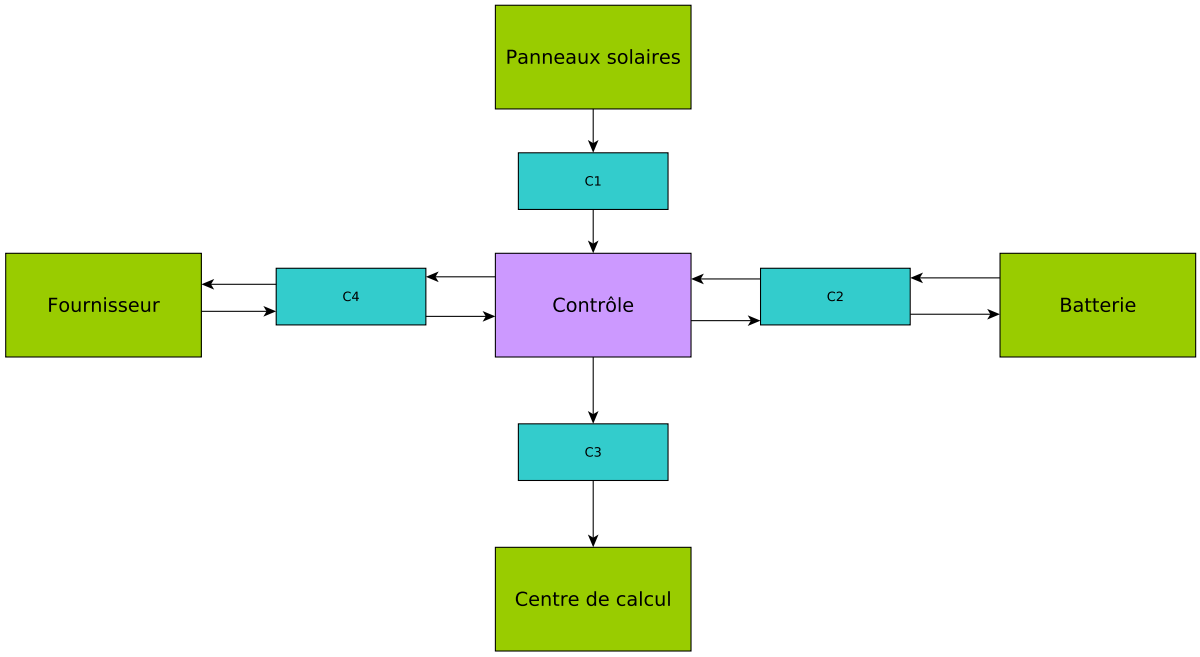


FIGURE 6.1 – Architecture du réseau intelligent pour les différentes expérimentations

La figure 6.1 présente les différents éléments du réseau intelligent reliés via un bus. Les convertisseurs C1 à C4 possèdent des pertes respectives de 15 %, 17 %, 13 % et 14 %. Ce choix de grandeur s'est orienté vers une caractérisation nominative car dans nos modèles la tension et le courant des différents composants ne sont pas représentés. Donc, nous avons décidé d'attribuer une valeur de perte car cette perte doit être considérée dans les modèles de décisions. Chaque élément du réseau est situé sur le même lieu géographique. C'est-à-dire, que la production solaire réalisée est directement acheminée vers le centre de calcul qui se situe dans le même bâtiment.

6.1.2 Les caractéristiques des composants du réseau intelligent

Les panneaux solaires

La production des panneaux solaires provient du bâtiment ADREAM [1]. Pour les différentes expérimentations, le choix s'est porté d'utiliser une production solaire sur 48h pour évaluer de manière efficace les différents résultats obtenus et avoir un cycle jour/nuit. La figure 6.2 présente un exemple de production solaire des panneaux du bâtiment ADREAM. Ce dernier est constitué de plus de 800 m^2 de panneaux. Pour certaines des expérimentations, la production a été adaptée pour obtenir une production relative à un centre de calcul inférieur à une dizaine de serveurs (cf. figure 6.3). Pendant 48 heures, la production solaire de ADREAM fournit 1648 mesures, une

1. https://www.laas.fr/public/fr/adream

unité de mesure équivaut à $\frac{48\times*60}{1648} \simeq 1,75$ mesures soit $\simeq$ 1 min et 45 secondes. La figure 6.2 présente donc deux jours consécutifs de production avec quelques pics de déclin de la production. Les pics de déclin peuvent être expliqués par le passage de nuages et/ou des intempéries. Une forme de « cloche » sur la courbe représente un jour.

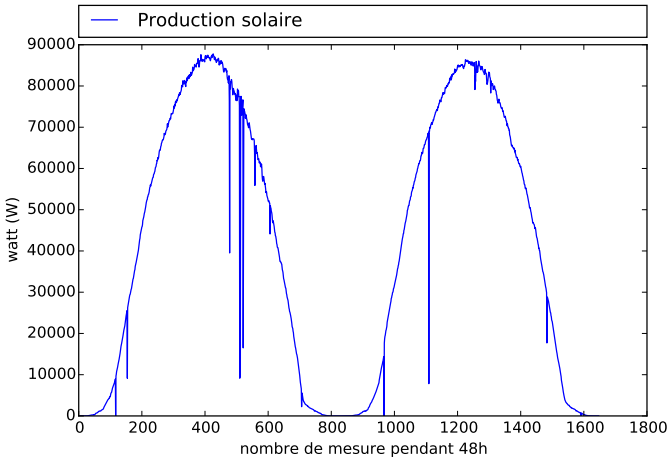


FIGURE 6.2 – Production solaire du bâtiment ADREAM sur 48 heures (Octobre 2015)

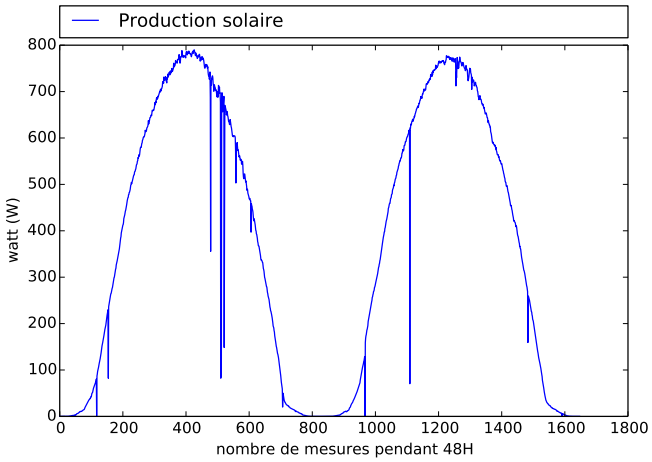


FIGURE 6.3 – Production solaire du bâtiment ADREAM sur 48 heures (Octobre 2015) adaptée à notre cas d'étude

La batterie

La puissance maximum de la batterie est de 200 W. Ce choix s'est orienté vers une caractérisation nominative de la puissance car dans nos modèles de charge et de décharge, la tension et le courant ne sont pas représentés. Certaines batteries possèdent environ cette ordre de grandeur en terme de puissance [96].

$$E_b(t) = E_b(t-1) + P_{bb} \times \Delta \tag{6.1.1}$$

$$E_b(t) = E_b(t-1) - P_{bb} \times \Delta \tag{6.1.2}$$

La charge et la décharge de la batterie sont respectivement modélisées par les équations 6.1.1 et 6.1.2, avec t = l'heure, Δ correspond à une capacité de temps de charge/décharge égale à 3 unités de temps, P_{bb} l'alimentation de la batterie après le module convertisseur (CM2) et E_b la l'énergie disponible dans la batterie (joules).

Le fournisseur d'électricité

L'achat et la vente au fournisseur sont respectivement modélisés par les équations 6.1.3 et 6.1.4 avec A la puissance achetée, V la puissance vendue, P_{ee} l'alimentation du fournisseur avant le module convertisseur et la perte du convertisseur l (CM4). Dans les expériences suivantes, les coûts d'énergie du fournisseur sont : 0,2366€ par kWh de 21h à 8h du matin (nuit) et 0.6173 € par kWh de 8h à 21h (jour) [2].

$$A(t) = P_{ee}(t)/(1-l) \tag{6.1.3}$$

$$V(t) = V(t-1) + P_{ee}(t) \times (1-l) \tag{6.1.4}$$

Le centre de calcul

Pour nos différentes expérimentations, les serveurs et la charge de travail ont des caractéristiques différents et seront définis au début de chaque expérimentation.

2. ENGIE - 2017 https://particuliers.engie.fr/electricite/conseils-electricite/prix-electricite/heures-pleines-heures-creuses.html

6.2 Politiques de décisions à travers un modèle de description des données

Dans cette partie 6.2 sont analysés les résultats du modèle de données présenté dans le chapitre 3 partie 3.4.2

6.2.1 Motivations des différentes expérimentations sur le modèle de description

Les expérimentations suivantes sont deux modèles de décisions possédant chacun un objectif différent :

1. **modèle environnemental** : utilise au mieux l'énergie renouvelable et consomme le moins possible d'énergie non renouvelable ;
2. **modèle lucratif** : minimise les flux d'énergie provenant du fournisseur d'électricité et maximise les ventes d'énergie des panneaux solaires ;

Afin de réaliser, les expérimentations de cette partie, nous avons utilisé une charge de travail définie en entrée du système. Cette dernière est représentée par la figure 6.4. Nous avons opté pour l'utiliser d'une charge de travail, proche de la production photovoltaïque car l'objectif est la création de charge de travail agile. La charge de travail est toujours consommatrice de service même la nuit, il peut y avoir des demandes utilisateurs ou applicatifs. Cependant, la journée, elle est plus importante, c'est là où la demande des utilisateurs est plus présente. Pour ces expériences, le nombre de serveurs n'est pas connu, nous connaissons simplement sa demande de consommation. Cette charge est adaptée à la production totale représentée par la figure 6.5 des panneaux solaires de ADREAM. Dans cette figure 6.5, la « cloche » est inférieure est la production solaire après le convertisseur (perte de 15 %). La courbe en pointillée est la charge de travail du centre de calcul. Cette dernière est nettement inférieure à la production solaire. Cependant, durant la nuit, le parc photovoltaïque ne produit aucune énergie mais la demande du centre de calcul reste stable à environ 7000 W.

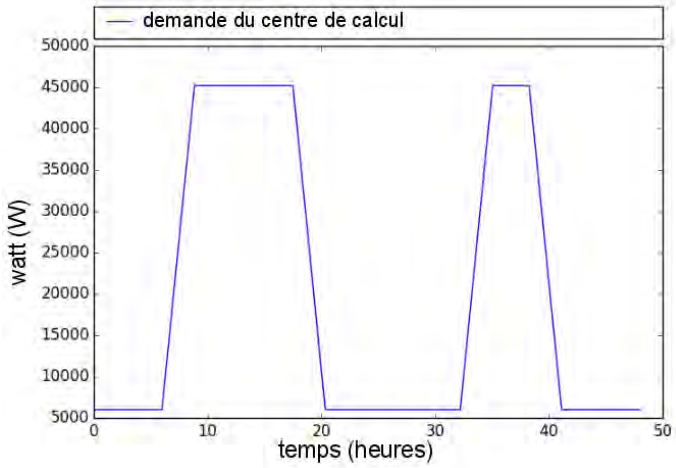


FIGURE 6.4 – Demande du centre de calcul pour les expérimentations de la partie 6.2

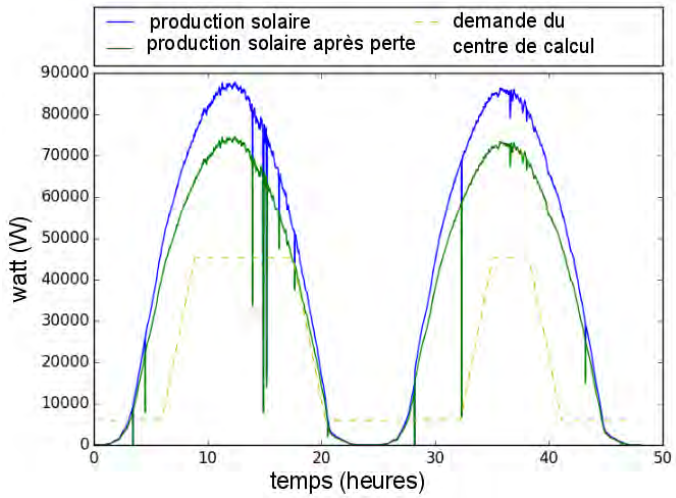


FIGURE 6.5 – Production solaire avant et après pertes générées par le module de conversion ainsi que la demande du centre de calcul pour les expérimentations de la partie 6.2

Les deux sous-parties suivantes présentent les simulations réalisées utilisant toutes les configurations introduites ci-dessus. Les deux expérimentations utilisent le modèle environnemental et une deuxième expérimentation utilise **modèle lucratif**.

6.2.2 Expérimentation 1 : un modèle pour une gestion énergétique portée vers le renouvelable

Conditions initiales 1

Pour cette expérimentation, nous avons utilisé la stratégie suivante :

TABLEAU 6.1 – Modélisation pour une stratégie moins consommatrice d'énergie fossile

	Source	**Destination**	**Sauvegarde**	**Trop-plein**
Panneaux Solaires	1	0	0	0
Centre de calcul	0	1	0	0
Batterie	0	0	1	1
Fournisseur	0	0	2	2

Dans ce cas, RenewSim ne pourra vendre de l'énergie au fournisseur que lorsque la somme de toutes les destinations est supérieure à la somme des sources (équation 6.2.1) et de la la batterie (élément de sauvegarde) (algorithme 2).

$$\sum S_t < \sum D_t \tag{6.2.1}$$

Analyse 1

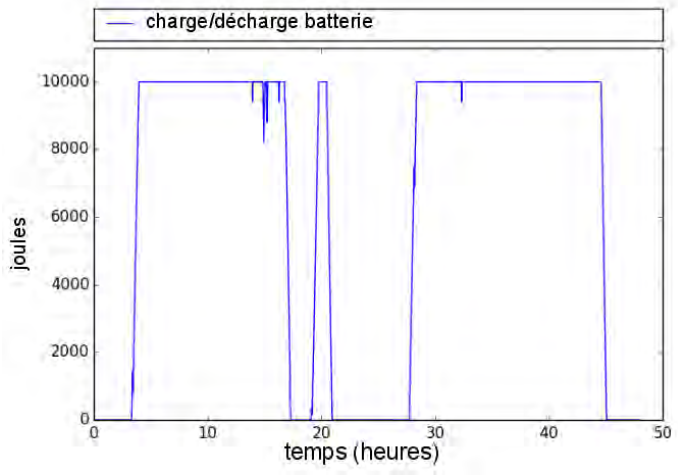


FIGURE 6.6 – Exp 1 : Charge de la batterie pendant deux jours

La charge de la batterie est représentée par la figure 6.6. La batterie peut fournir le centre de calcul lorsque la production solaire ralentit. À environ $t = 15\ h\ 17\ h$, il y a une baisse de

production solaire (cf. figure 6.2). Cette baisse de production par intermittence peut s'expliquer par une mauvaise météo pendant deux heures. Dans cette période horaire, la figure 6.6 montre la charge/décharge de la batterie pour donc palier à ces manques de production solaire.

Les figures 6.7et 6.8 sont respectivement les courbes d'achat et de vente au fournisseur. La vente a lieu lorsque la batterie est pleine. L'achat au près du fournisseur a lieu dans deux cas :

1. Lorsque la production solaire et la batterie ne permettent pas de répondre à la demande du centre de calcul ;
2. Lorsque la puissance maximale de la batterie est dépassée pour fournir à temps le centre de calcul.

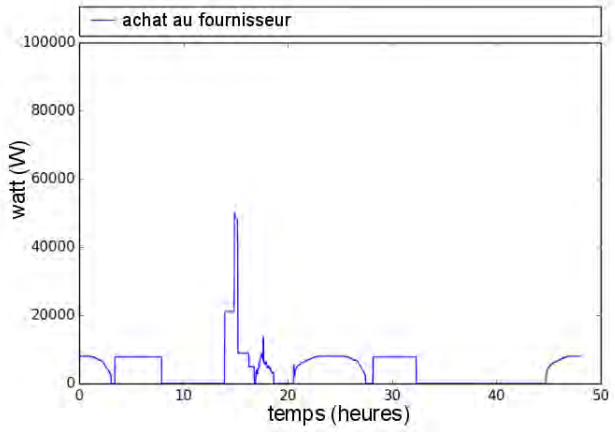


FIGURE 6.7 – Exp 1 : Achat au fournisseur pendant deux jours

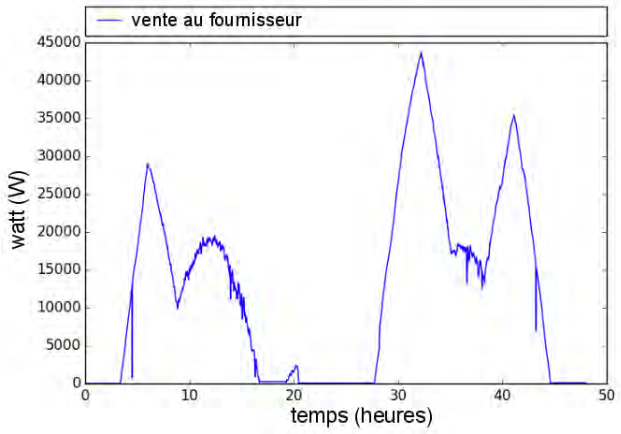


FIGURE 6.8 – Exp 1 : Vente au fournisseur pendant deux jours

Dans cette expérience à environ 18 heures (le premier jour) nous nous retrouvons dans le cas n^o2. Pendant la nuit, la batterie est déchargée (figure 6.6), puis l'achat au fournisseur commence à environ 20h (figure 6.7). Cependant, suite à la stratégie énergétique effectuée (tableau 6.1), la vente d'énergie dans la nuit ne se réalise jamais car le réseau intelligent n'est pas en sur-production et la batterie est vide.

6.2.3 Expérimentation 2 : un modèle économique porté sur la réduction de la facture d'électricité

Conditions initiales 2

Pour cette expérience, deux modèles de stratégie de flux énergétiques ont été utilisés, différenciant le jour (de 8h à 21h) et la nuit de (21h à 8h). Ils ont été introduits à titre d'exemple dans le chapitre 3 dans la sous-partie 3.4.2.

Tableau 6.2 – Exp 2 : Modélisation pour une stratégie moins couteuse (jour)

	Source	**Destination**	**Sauvegarde**	**Trop-plein**
Panneaux solaires	1	0	0	0
Centre de calcul	0	1	0	0
Batterie	0	0	1	0
Fournisseur	0	0	2	1

Tableau 6.3 – Exp 2 : Modélisation pour une stratégie moins couteuse (nuit)

	Source	**Destination**	**Sauvegarde**	**Trop-plein**
Panneaux solaires	1	0	0	0
Centre de calcul	0	1	0	0
Batterie	0	1	0	0
Fournisseur	1	0	0	0

La modélisation dans les tableaux 6.2 et 6.3 favorisent le coût énergétique plutôt que l'énergie renouvelable. Pendant la journée, le modèle représenté par le tableau 6.2 est le suivant :

- la production des panneaux solaires alimente le centre de calcul.
- s'il y a une sur-production d'énergie (demande centre de calcul $\leq$ production de panneaux solaires), le surplus est vendu aux fournisseurs.
- s'il y a une sous-production (demande centre de calcul $\geq$ production de panneaux solaires), le manque est retiré de la sauvegarde 1. Dans ce cas, la batterie sera déchargée.

- si la sauvegarde est vide, le besoin d'énergie manquant est retiré de la sauvegarde 2. Dans ce cas, le fournisseur sera prestataire d'énergie (achat).

Pendant la nuit, le modèle représenté par le tableau 6.3 est le suivant :

- la production des panneaux solaires alimente le centre de calcul et charge la batterie.
- le fournisseur alimente le centre de calcul et charge la batterie.

Le modèle de nuit permet au centre de calcul de ne jamais se retrouver en sous-production, car le fournisseur est utilisé aussi comme source principale pour alimenter à la fois le centre de calcul et la batterie. Même s'ils sont représentés dans le modèle, les panneaux solaires ne produisent pas d'énergie pendant la nuit. Cependant, pendant une fenêtre de 1h (entre 21h et 22h), ils peuvent produire de l'énergie si la nuit n'est pas tombée, par exemple, durant l'été. Cette stratégie énergétique divisée en deux parties (jour/nuit) permet de réduire la facture d'électricité. Effectivement, le fournisseur d'électricité possède deux catégories de prix une pour les heures pleines (la journée) et une pour les heures creuses (la nuit) [3]. Le tarif d'électricité pendant la nuit est moins coûteux. Dans ce cas là, la stratégie est d'utiliser l'énergie renouvelable pendant la journée au maximum (batterie, et panneaux solaires) puis, durant la nuit de ne pas utiliser la batterie et de la charger via le fournisseur pour l'utiliser lorsque le tarif d'électricité est plus élevé (la journée).

Analyse 2

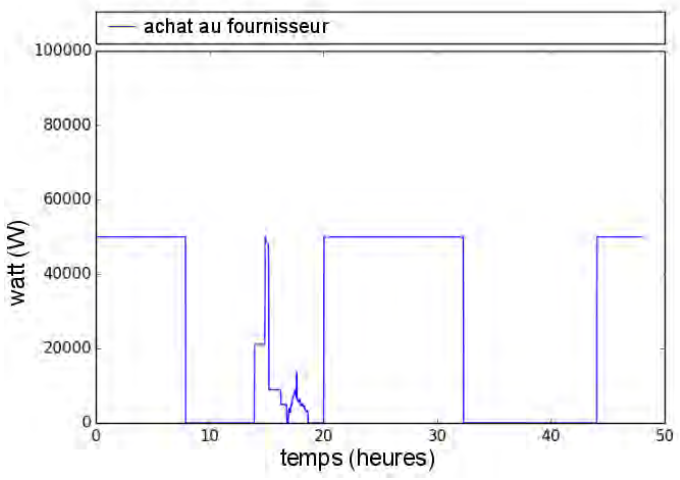


FIGURE 6.9 – Exp 2 : Achat au fournisseur pendant deux jours

3. ENGIE - 2017 https://particuliers.engie.fr/electricite/conseils-electricite/prix-electricite/heures-pleines-heures-creuses.html

La figure 6.10 présente la charge et la décharge de la batterie. Le premier jour la batterie diminue lorsque la production solaire commence à s'arrêter. Cependant, la nuit la batterie maximise son chargement se qui permet de ne pas acheter au fournisseur le second jour pendant la journée (figure 6.9).

Les deux modèles fonctionnent ensemble et doivent fonctionner dans une fenêtre temporelle où il y a au moins deux cycles jour/nuit.

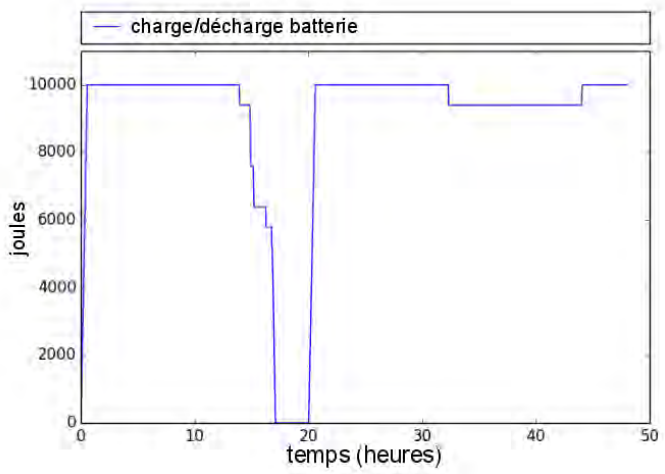


FIGURE 6.10 – Exp 2 : Charge de la batterie pendant deux jours

6.2.4 Bilan des modèles de décisions

L'objectif de ces expériences était de présenter un modèle pour représenter chaque élément d'un réseau intelligent et permettre au module de contrôle de RenewSim d'obtenir une stratégie pour le flux énergétique. Le modèle permet une architecture polymorphe qui vise à prendre une décision intelligente pour l'utilisation de sources hétérogènes. Dans nos expérimentations, il est mis en avant un modèle de données qui permet la redirection d'alimentation en identifiant et diffusant les différentes sources d'énergie. Grâce à deux scénarios distincts, le module de contrôle peut prendre une décision dans le but : utiliser plus d'énergie renouvelable ou acheter à faible coût. Ce modèle généralise tout le processus où nous pouvons déplacer la charge d'alimentation.

À l'avenir, cette représentation peut permettre d'inclure

- des modèles de prévisions et/ou des modèles d'apprentissages [97, 98] ;
- des études d'optimisation multi-objectifs via des heuristiques [4, 48] ;
- de la co-simulation en intégrant des simulateurs dédiés à chaque composant.

Dans la partie suivante, nous allons faire des simulations des algorithmes présentés dans le chapitre 5 à l'aide de RenewSim.

6.3 Évaluation et comparaison des différents algorithmes de décisions

Dans ces expériences est évaluée la consommation du centre de calcul, l'utilisation des énergies renouvelables et le comportement des VMs avec les différents algorithmes décrits dans le chapitre 5. Les indicateurs utilisés sont présentés dans le chapitre 5 (partie 5.4). Dans ces expériences, est mis en avant la corrélation les flux énergétiques, proposés dans le modèle présenté dans le chapitre 3 et expérimenté dans la section précédente 6.2, et les flux d'énergies.

6.3.1 Motivations des différentes expérimentations sur les algorithmes de placements

Les expériences suivantes sont la mise en place des différents algorithmes présentés dans le chapitre 5. Ils sont introduits dans le module de contrôle de RenewSim afin d'obtenir des résultats sur l'efficacité énergétique et sur la QoS via un ensemble d'indicateurs présenté dans la partie 5.4. Afin de mettre en corrélation les flux informatiques et énergétiques, ces expériences mettent en évidence une prise de décisions sur des objectifs hétérogènes. Ces algorithmes ordonnancent un ensemble de VMs sur des serveurs et dans le temps. Pour une expérience, un ensemble de VMs est donné en entrée du simulateur. Ces dernières correspondent à des traces synthétiques générées à partir des données Google [8]. Ces traces synthétiques suivent le modèle proposé dans le chapitre 4, dans la partie 4.5. Nous rappelons que la VM est décrite de la manière suivante :

- **ide** : un identifiant unique qui nous permet de suivre l'évolution d'une VM.
- **tps_soumission** : le temps d'arrivée de la VM dans la pile de VM à exécuter, ce temps nous permet de savoir quand la VM est éligible pour être ordonnancée. Pour les expériences suivantes le temps de soumission n'est pas pris en compte. Toutes les VMs arrivent en même temps, c'est-à-dire au début, de la simulation.
- **tps_debut_exe** : le temps où la VM est ordonnancée, c'est-à-dire le moment où la VM s'exécute sur le serveur. Pour les expériences suivantes, l'unité de mesure est donnée par l'échantillonnage de la production solaire de ADREAM (1648 mesures). Par exemple, si la VM est ordonnancée à partir de l'unité de mesure 44 de la production solaire, alors **tps_debut_exe** est égal à 44.
- **tps_fin_exe** le temps où la VM est terminée. Pour les expériences suivantes, l'unité de mesure est toujours celle de l'échantillonnage donné par la production solaire du bâtiment ADREAM. Cependant, le temps d'exécution d'une VM est toujours le même que les

données proposées par Google. Par conséquent, le temps de fin est normalisé via les données de mesure ADREAM. Par exemple, pour la durée d'une tâche Google égale à 103 unités de mesure (3 heures). Si celle-ci est ordonnancée dans RenewSim comme VM à l'unité de temps 44, alors la VM sera finie à $44 + 103 = 147$. Soit **tps_fin_exe** est égal 147.

- **évènement_de_fin** : l'évènement de fin. Il peut être de type : ≪ Fail ≫ , ≪ Kill ≫, ≪ Evict ≫, ≪ Finish ≫. Dans ces expériences, **évènement_de_fin** est une donnée, qui n'influence pas la décision de l'ordonnanceur.
- **priorité** : sa priorité. Dans ces expériences, **priorité** est une donnée, qui n'influence pas la décision de l'ordonnanceur.
- **latence** : sa sensibilité à la latence. Dans ces expériences, **latence** est une donnée, qui n'influence pas la décision de l'ordonnanceur.
- **vCPU** : demande vCPU de la VM. Dans ces expériences, le vCPU des différentes VMs est égal 2 pour être en adéquation avec le nombre de serveurs et leur capacité. Cela nous permet de calculer la consommation d'une seule VM sur un serveur.

Pour les expériences suivantes, il a été utilisé 1000 VMs (600 courtes, 200 moyennes et 200 longues) de différentes durées. Il y a plus de VMs courtes car Google a plus de tâches courtes que longues. Les tâches courtes ont une moyenne de temps de fin d'attente de 3 minutes, les tâches les plus longues au-dessus de 3 heures et les tâches moyennes de 45 minutes à 2 heures. Cela nous permet de garder une hétérogénéité dans la charge de travail. Le choix du nombre de VMs s'est porté à 1000 car cette charge de travail devait être en adéquation avec les capacités des serveurs disponibles. Le centre de calcul est composé d'un ensemble de serveurs possédant des caractéristiques hétérogènes.

Le centre de calcul est composé de cinq serveurs (tableau 6.4) avec des caractéristiques différentes. Le serveur s4 possède beaucoup plus de vCPU (20 vCPU) que les autres car nous avions besoin d'un serveur ayant des ressources plus importantes pour les VMs longues par exemple. Nous avons choisi un centre de calcul de petite taille et hétérogène, pour comprendre les différents flux informatiques et énergétiques. De plus, le nombre de cinq serveurs a été choisi, afin d'obtenir une corrélation entre le nombre de VMs et la production solaire pendant 48 heures. A l'état initial de l'expérimentation, tous les serveurs sont éteints. Ils s'allument pour la première fois dès qu'une VM est ordonnancée. Pour l'algorithme RR_CLASSES, nous avons utilisé les classes suivantes : VMs a durée d'exécution courte [s0], VMs a durée d'exécution moyenne [s1] et VMs d'exécution longue [s2, s3, s4]. Il y a plus de serveur pour les VMs longues car elles utilisent le vCPU des serveurs pendant une plus longue période. Donc, les VMs longues utilisent plus de vCPU dans le temps que les VMs moins longues (moyennes, courtes).

Tableau 6.4 – Nombre de vCPU par serveurs

s0	s1	s2	s3	s4
8 vCPU	8 vCPU	16 vCPU	16 vCPU	20 vCPU

La consommation est modélisée par l'équation 6.3.1. La consommation représente la consommation à l'instant t d'une machine virtuelle sur un serveur.

$$C_{ser}(t) = base + (Pmax_{ser} - Pmin_{ser}) \times (\frac{vcpu_{vm}}{vcpu\_total_{ser}}) \tag{6.3.1}$$

- *base* est la puissance minimale du serveur ($Pmin_{ser}$), si le serveur est déjà allumé, la *base* est égale à zéro sinon égale à la puissance minimale du serveur($Pmin_{ser}$) ;
- $vcpu_{vm}$ est le nombre de vCPU demandé par une VM ;
- $vcpu\_total_{ser}$ est le total de vCPU appartenant au serveur ;
- $Pmax_{ser}$ la puissance max du serveur

La puissance minimale de chaque serveur est de 80 W et la puissance maximale est de 169. Ces données sont extraites de données réelles, elles ont été mesurées sur des serveurs DELL R630 possédant deux CPU (XEON).

Les expériences suivantes aide à :

1. analyser et comprendre le résultat d'ordonnancement des algorithmes (partie 6.3.2) ;
2. connaître l'algorithme qui consomme le moins (partie 6.3.3) ;
3. identifier les algorithmes qui éjectent le plus de VMs (partie 6.3.4) ;
4. distinguer les algorithmes utilisant le moins d'énergie renouvelable via l'indicateur GEC (partie 6.3.5).

Toutes ces simulations sont liées les unes aux autres et nous ont permis de :

- comprendre le lien entre le nombre de VMs éjectées et la consommation d'énergie totale ;
- comparer la part d'énergie verte à la consommation d'énergie totale.

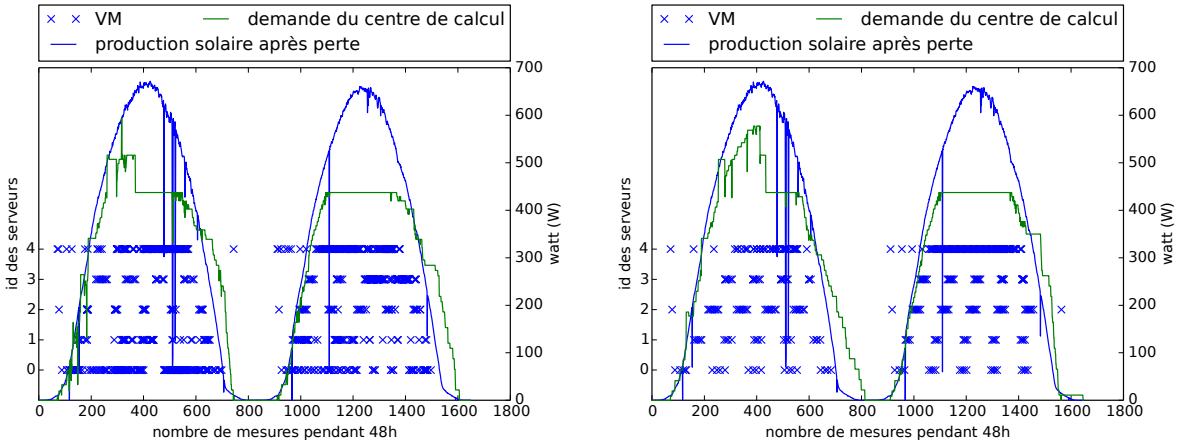
6.3.2 Les résultats de placement pour chaque algorithme

Dans cette partie sont présentés les résultats d'ordonnancement, en terme de placement sur les serveurs du centre de calcul, pour chaque algorithme.

Analyse 1

Les algorithmes de type First Fit, (figures 6.11a et 6.11b) présentent des résultats d'ordonnancement assez similaires. La différence est visible au niveau des serveurs. Les figures 6.12a et

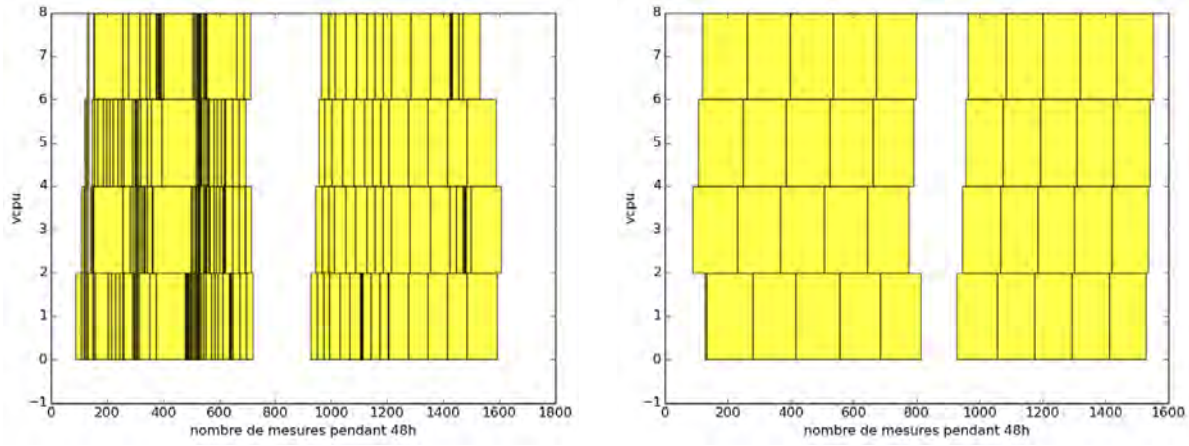
6.12b exposent le résultat d'ordonnancement pour le serveur *S0*. Chaque boite jaune est une VM. Dans l'algorithme FF_LPT, les VMs les plus longues sont placées en premiers sur le premier serveur, puis sur le second jusqu'au dernier le serveur (*S4*). L'écart d'ordonnancement des VMs est plus important dans les premiers serveurs puis diminue. Cependant, on observe un écart pour le serveur *S4*. Les figures 6.13a et 6.13b mettent en avant ce phénomène. Ce serveur *S4* est le dernier de l'algorithme où les VMs sont ordonnancées. L'énergie solaire disponible est donc moins importante car beaucoup d'autres VMs sont déjà en cours d'exécution sur les autres serveurs. De plus, cet écart d'ordonnancement entre les différentes VMs, se matérialise au moment où les panneaux solaires produisent le moins d'énergie, c'est-à-dire, en début de journée, ou en fin de journée. Le premier pic de consommation, sur chacun des résultats est dû à l'allumage des serveurs (équation 6.3.1).



(A) Distribution des VMs avec consommation de centre de calcul et production de panneaux solaires pour l'algorithme FF

(B) Distribution des VMs avec consommation de centre de calcul et production de panneaux solaires pour l'algorithme FF_LPT

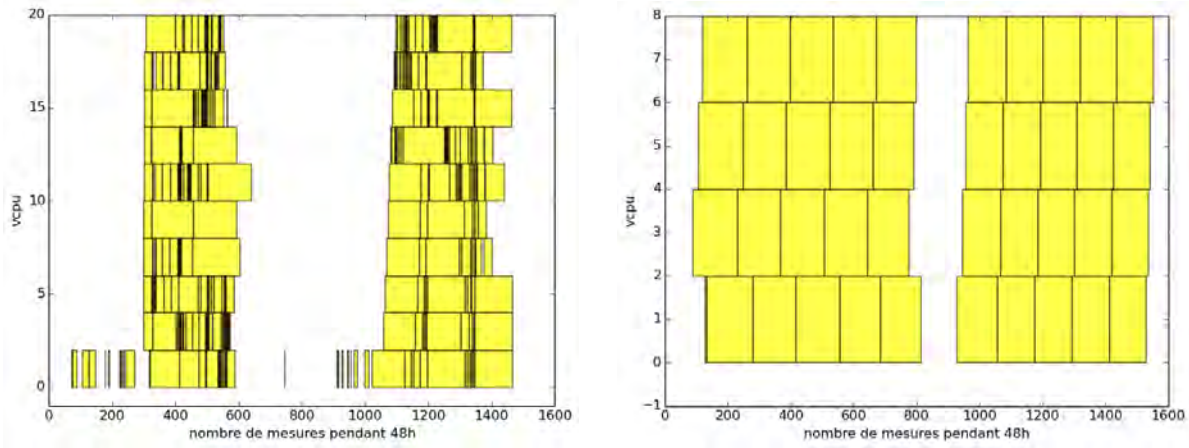
FIGURE 6.11 – Les algorithmes de type First Fit



(A) Ordonnancement des VMs sur le serveur *S0* pour l'algorithme FF

(B) Ordonnancement des VMs sur le serveur *S0* pour l'algorithme FF_LPT

FIGURE 6.12 – L'ordonnancement dans le serveur S0 pour les algorithmes de type First Fit



(A) Ordonnancement des VMs sur le serveur *S4* pour l'algorithme FF

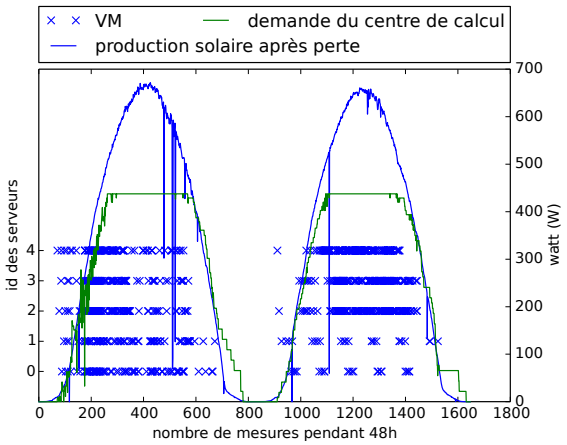
(B) Ordonnancement des VMs sur le serveur *S4* pour l'algorithme FF_LPT

FIGURE 6.13 – L'ordonnancement dans le serveur *S4* pour les algorithmes de types First Fit

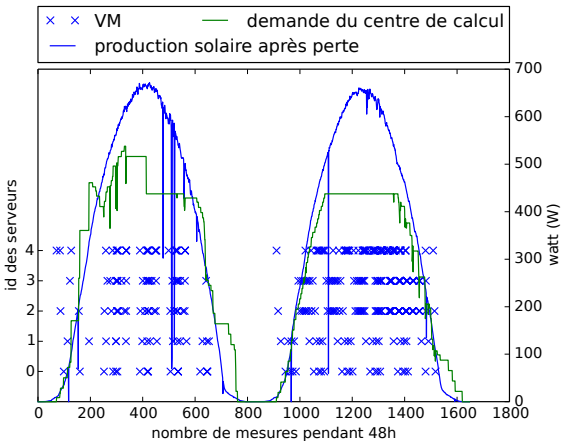
Analyse 2

Les algorithmes de type Round Robin, figures 6.14a, 6.14b, et 6.14c, présentent des résultats différents. Tout d'abord, sur l'algorithme RR 6.14a, le pic d'allumage des machines n'apparaît pas. Cela peut s'expliquer par le fait que les serveurs, au début de l'expérimentation, ne sont pas surchargés et le pic d'allumage est aussi grand que la consommation des serveurs lorsque ceux-ci sont déjà allumés en milieu de journée. Ensuite, lors de la première matinée (unité de temps $\simeq 150$), pour l'algorithme RR_LPT (figure 6.14b), la consommation des serveurs est supérieure à

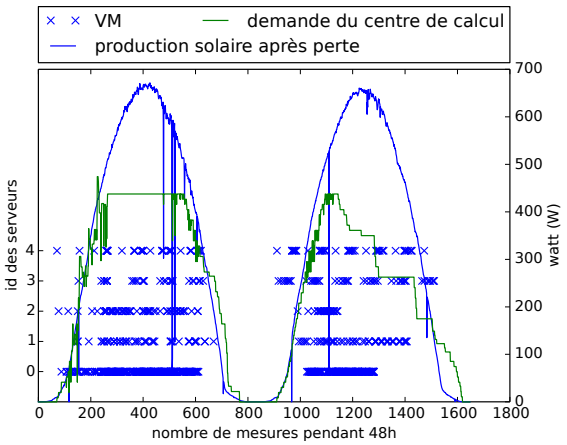
la production photovoltaïque. Ce résultat s'observe, aussi, lorsque la production photovoltaïque diminue. Chaque algorithme ordonne des VMs lorsqu'il y a assez d'énergie photovoltaïque à un instant t. Cependant, une VM continue à consommer tout au long de son cycle de vie. Même s'il n'y a pas assez d'énergie photovoltaïque à un instant t mais que des VMs ont commencé à s'exécuter à $t-1$, elles vont continuer à consommer, même après. Pour l'algorithme RR_CLASSES (figure 6.14c), les serveurs *S0*, *S1* et *S2*, ne sont pas occupés jusqu'à la fin du deuxième jour (unité de temps $\simeq 1300$), ces serveurs appartiennent à la classe des VMs courtes et des VMs longues.



(A) Distribution des VMs avec consommation de centre de calcul et production de panneaux solaires pour l'algorithme RR



(B) Distribution des VMs avec consommation de centre de calcul et production de panneaux solaires pour l'algorithme RR_LPT



(C) Distribution des VMs avec consommation de centre de calcul et production de panneaux solaires pour l'algorithme RR_CLASSES

FIGURE 6.14 – Les algorithmes de type Round Robin (RR)

Plusieurs hypothèses peuvent être émises pour cette répartition des VMs :

- soit toutes les VMs courtes et longues sont programmées ;

- soit il n'y a plus assez de vCPU disponible sur les serveurs sur une durée importante pour exécuter les VMs de types longues (Serveur *S2*).

Pour comprendre les différents résultats d'ordonnancement obtenus dans les algorithmes ci-dessus, les parties suivantes vont analyser le comportement énergétique, et les caractéristiques des VMs ordonnancées.

6.3.3 Consommation d'énergie pour les différents algorithmes

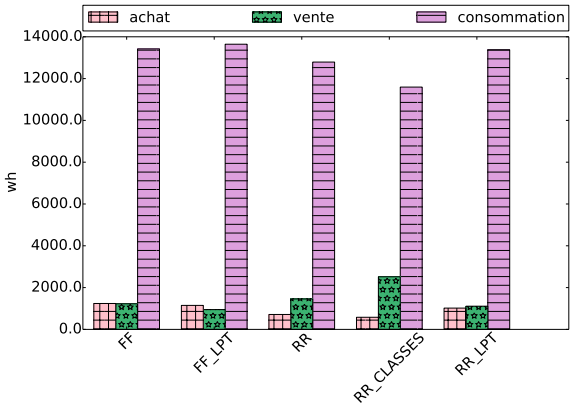
Analyse 3

La consommation Pour chaque algorithme, la consommation a été comparée aux VMs ordonnancées et éjectées. La figure 6.15a présente ces résultats. Pour chaque algorithme, la valeur de la consommation varie de 11 500 Wh à 13 000 Wh pendant 48 heures. L'algorithme qui consomme le moins est RR_CLASSES (11 500 Wh). Ce résultat s'explique par le regroupement de serveurs en classe. Chaque serveur exécute un type de VM : long, moyen ou court. Donc, chaque serveur est entièrement optimisé. Cela inclut qu'il a connaissance de la charge de travail avant le début de l'ordonnancement. Dans notre cas, il n'y a qu'un seul serveur pour les VMs les plus courtes et trois pour les VMs les plus longues car elles utilisent plus de ressources au fil du temps. Cet algorithme possède également le meilleur résultat sur la vente et l'achat. Il vend plus d'énergie que les autres algorithmes et achète moins d'énergie.

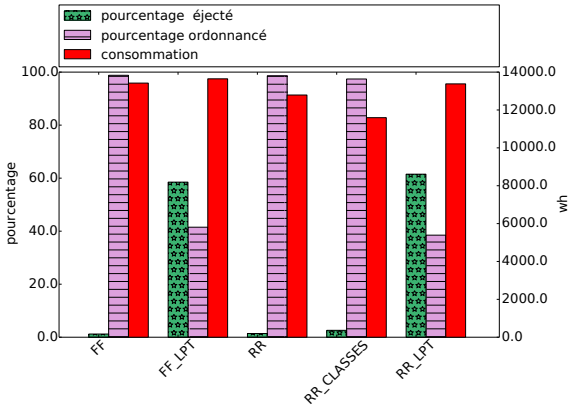
L'achat Pour chaque algorithme, la valeur de l'achat ne dépasse pas 2000 Wh pendant 48 heures. Les algorithmes FF et FF_LPT possèdent les consommations les plus élevées et les achats le plus élevés, car ils utilisent les ressources maximales de chaque serveur (au moins les premiers). Ces résultats s'expliquent par le fait que tous ces algorithmes ne tiennent pas compte de l'énergie renouvelable pendant l'exécution d'une VM. Ils vérifient seulement si la VM a suffisamment d'énergie renouvelable à l'instant t pour être programmée à l'instant t. S'il n'y a plus d'énergie au temps $t+1$, l'algorithme continue d'exécuter la VM et obtiendra l'énergie de la batterie (Sauvegarde=1) ou s'il n'y a plus d'énergie dans la batterie, l'énergie sera achetée chez le fournisseur d'électricité. Les serveurs doivent être alimentés en permanence dans le temps contrairement à un algorithme Round Robin. Un algorithme Round Robin recherche des ressources serveur par serveur.

La vente Pour chaque algorithme, la valeur de la vente varie de 1500 Wh à 3000 Wh pendant 48 heures. L'algorithme RR_CLASSES a toujours le meilleur résultat ($\approx$ 3000 Wh). Il vend plus d'énergie que les autres algorithmes car les classes de certains serveurs sont pleines et d'autres sous utilisées. Pour comprendre ces résultats, il est nécessaire de suivre les taux de VMs éjectées et ordonnancées.

VMs éjectées et ordonnancées La figure 6.15b présente la consommation par rapport aux VMs ordonnancées et éjectées. Les algorithmes FF, RR et RR_CLASSES possèdent les meilleurs résultats pour les VMs ordonnancées, le pourcentage de VM éjectées est très faible (entre 0,5 et 2%) mais l'algorithme RR_CLASSES consomme moins que les autres. Cela s'explique par la taille des VMs éjectées, car si elles sont plus longues, elles produiront une consommation maximale.



(A) La consommation du centre de calcul comparée à la vente et à l'achat d'énergie



(B) La consommation du centre de calcul comparée aux nombres de VMs ordonnancées et éjectées

FIGURE 6.15 – Impact de la consommation des différents algorithmes

6.3.4 L'impact du nombre de VMs ordonnancées et éjectées pour chaque algorithme

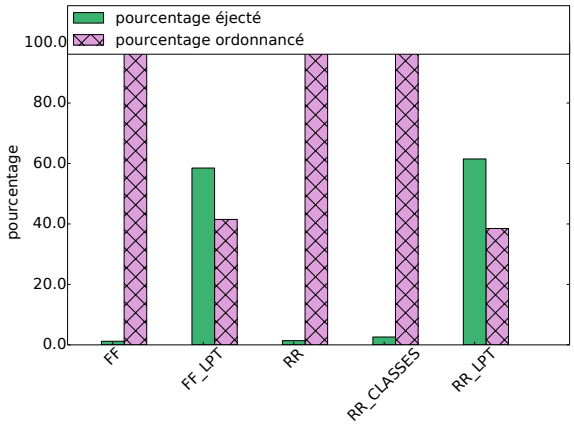
Nous allons étudier les caractéristiques des VMs de chaque algorithme, par rapport à leur quantité horaire de VMs éjectées/ordonnancées et leur temps d'exécution.

Analyse 4

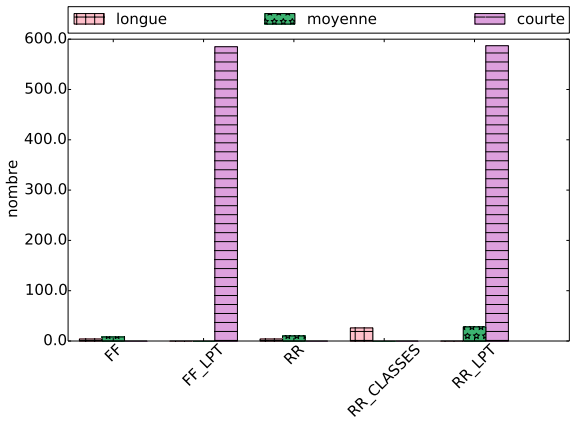
Résultats ordonnancement et caractéristiques des VMs éjectées/ordonnancées La figure 6.16a présente le pourcentage de VMs éjectées et ordonnancées. L'algorithme FF possède le moins de VMs éjectées avec les algorithmes RR_CLASSES et RR. Cependant, la figure 6.16b présente la taille des VMs éjectées et l'algorithme RR_CLASSES éjecte seulement des VMs longues. Ce type de VMs consomme plus d'énergie que les autres, ce qui explique les meilleurs résultats de consommation par rapport aux autres algorithmes (cf. analyse 6.3.3).
La figure 6.16b montre que l'algorithme FF_LPT possède beaucoup de VMs éjectées pour les plus courtes, ceci s'explique par l'ordonnancement prioritaire des VMs longues.
L'algorithme RR_LPT possède beaucoup de VMs éjectées et une énorme quantité horaire éjectées (92,5 heures) étant donné que les VMs courtes n'ont jamais accès aux ressources, ce sont les VMs

longues qui ont la priorité sur les VMs courtes, et moyennes (figure 6.16b). L'espace de ressources en terme de vCPU n'est pas disponible pour ordonnancer des VMs, et les intervalles vCPU disponibles sont trop courts pour accueillir des VMs de taille moyenne.

Quantité horaire des VMs ordonnancées/éjectées par algorithme La figure 6.17a présente la quantité horaire de VMs éjectées par rapport au pourcentage d'éjection de chaque algorithme. Cette illustration met en avant la plus haute quantité horaire de VMs éjectées (65 heures) détenue par l'algorithme RR_CLASSES. Cependant, cet algorithme possède un pourcentage de VMs éjectées relativement faible.
L'algorithme FF consomme plus d'énergie que les autres algorithmes (figure 6.15a), cette analyse est justifiée par la quantité horaire de VMs ordonnancées très élevée (figure 6.17b) et le nombre de VMs ordonnancées. L'algorithme FF possède de meilleurs résultats concernent la quantité horaire des VMs éjectées et le pourcentage d'expulsion. Sa quantité horaire de VMs éjectées est de 25 heures, étant donné que toutes les VMs éjectées sont des VMs moyennes ou courtes et qu'elles ne sont que 12.
Les algorithmes RR_LPT et FF_LPT ont ordonnancé plus de 70 % des VMs et n'ont pas la quantité horaire ordonnancée la plus faible ceci s'explique par la priorisation d'ordonnancement des VMs longues.

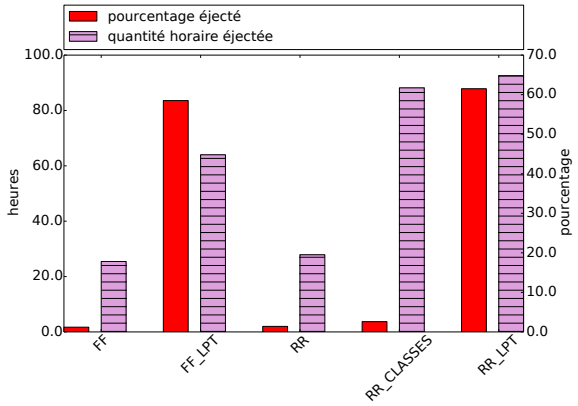


(A) Les résultats de l'ordonnancement par rapport au pourcentage de VMs programmées et de VMs éjectées pour chaque algorithme

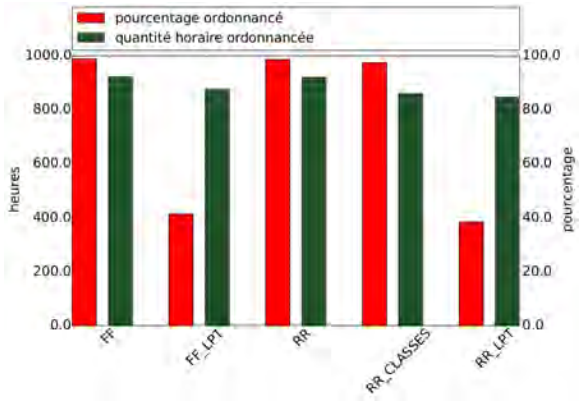


(B) Les résultats de l'ordonnancement par rapport à la taille des VMs éjectées pour chaque algorithme

FIGURE 6.16 – Impact du nombre de VMs ordonnancées et éjectées pour chaque algorithme



(A) Les résultats de l'ordonnancement par rapport à la quantité horaire des VMs éjectées pour chaque algorithme



(B) Les résultats de l'ordonnancement par rapport à la quantité horaire des VMs ordonnancées pour chaque algorithme

FIGURE 6.17 – Impact de la quantité horaire de VMs ordonnancées et éjectées pour chaque algorithme

6.3.5 L'impact énergétique pour chaque algorithme

Pour rappel, l'équation 5.4.1 propose de calculer le GEC dans notre cas. Cette équation nous permet de quantifier la part d'énergie renouvelable consommée par un centre de calcul, remarquons que plus la vente sera grande et l'achat faible plus le GEC sera élevé (meilleur).

$$GEC = 0 \leqslant \frac{Prod_{solaire} - Prod_{sell}}{Fournisseur_{achat} + (Prod_{solaire} - Prod_{vend})} \leqslant 1 \tag{6.3.2}$$

La figure 6.18 montre les résultats du GEC pour chaque algorithme. Pour confirmer ces résultats, le tableau 6.5 présente les résultats des différentes métriques utilisées pour chaque algorithme afin d'évaluer la stratégie d'ordonnancement.
L'algorithme FF_LPT consomme le plus, vend le moins et achète le plus d'énergie, c'est pour cela que son GEC est le plus faible. Cependant, sa quantité horaire éjectées est la meilleure, étant donné, qu'il ordonnance les VMs les plus longues d'abord.
Le meilleur GEC est détenu par les algorithmes ayant une quantité horaire de VMs éjectées très élevée (RR_CLASSES et RR_LPT) mais les deux algorithmes Round Robin n'optimisent pas les ressources des différents serveurs. C'est pourquoi, il y a beaucoup de VMs éjectées. Cependant, il y a moins de VMs éjectées pour le RR_CLASSES ceci s'explique par l'ordonnancement via des classes serveurs. Cet algorithme peut être optimisé en redéfinissant les classes serveurs. Dans notre cas, il n'y a pas assez de ressources pour exécuter certaines tailles de VMs. Selon la figure 6.16b, les VMs les plus éjectées appartiennent au type moyen (de 45 minutes à 2 heures).

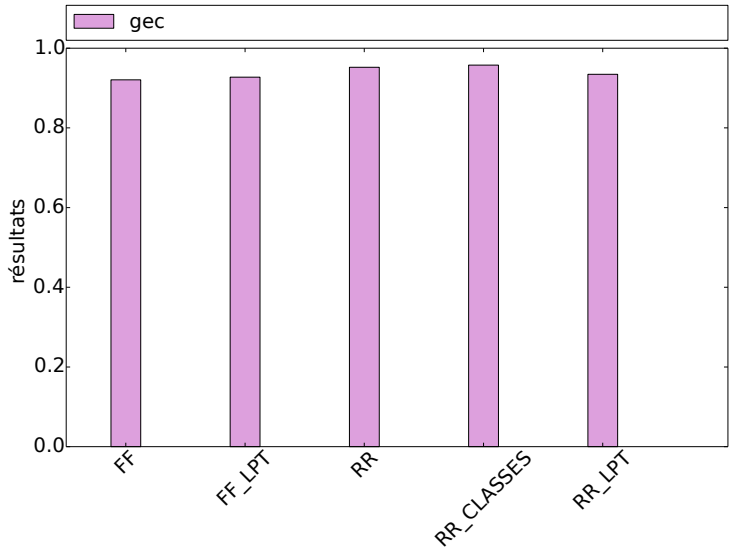


FIGURE 6.18 – Résultat du GEC pour chaque algorithme

TABLEAU 6.5 – Résultats des différentes métriques observées pour chaque algorithme

	RR	**FF**	**RR_CLASSES**	**RR_LPT**	**FF_LPT**
Nombre de VMs éjectées	14 (1.4%)	12 (1.2%)	49 (2.6%)	615 (61.5%)	585 (58.5%)
Nombre de VMs ordonnancées	986 (98.6%)	988 (98.8%)	981 (97.4%)	385 (38.5%)	415 (41.5%)
Quantité horaire ordonnancée (≈ Heures)	920	922	858	846	875
Quantité horaire éjectée (≈ Heures)	28	25	88	92.5	63
GEC	0.95	0.92	0.96	0.93	0.93

6.3.6 Bilan

Dans cette partie, il a été introduit une approche d'ordonnancement écologique d'une charge de travail. Grâce à ces expériences, les différents algorithmes permettent de comprendre la charge de travail agile. Toutes ces analyses sont récurrentes car nous faisons une expérimentation sur un cas d'étude précis. La stratégie d'ordonnancement énergétique proposée peut améliorer l'utilisation de l'énergie verte. L'algorithme RR_CLASSES possède les meilleurs résultats, mais pour configurer les classes aussi bien que possible, il faut connaître les caractéristiques de la charge de travail. Les algorithmes FF et FF_LPT semblent appropriés pour ordonnancer une charge de travail agile, étant donné qu'ils permettent au centre de calcul d'utiliser le nombre approprié de serveurs et d'utiliser la capacité de chaque serveur. Une perspective pourrait être d'ajouter un taux de QoS en tant que principale contrainte pour obtenir une meilleure QoS. Pour atteindre

ce résultat, nous pouvons utiliser les priorités et les caractéristiques de latence proposées par Google.
L'expérimentation suivante est une étude sur la corrélation entre la production solaire et la capacité de la batterie en fonction d'une charge de travail.

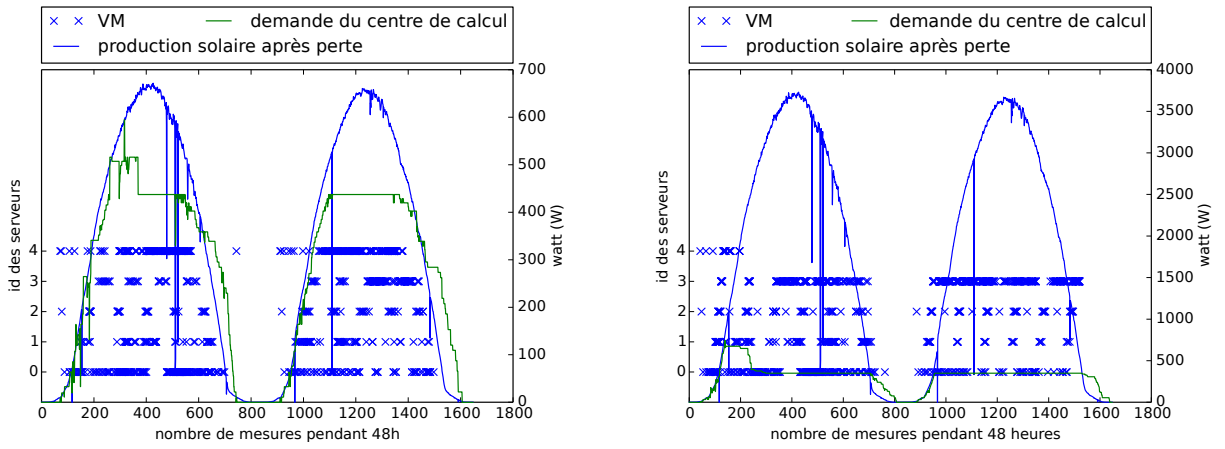
6.4 Dimensionnement de la taille de la batterie par rapport à la production solaire

À travers le simulateur RenewSim, l'étude suivante a permis de mettre en avant l'influence de la taille de la batterie et de la production solaire sur une charge de travail. Un ensemble de simulations a été réalisé sur une même charge de travail en faisant varier les données d'entrées du simulateur, dans ce cas, la capacité de stockage de la batterie et la production solaire.

Pour évaluer la taille de la batterie par rapport à la production des panneaux solaires, il a été choisi d'analyser le GEC de l'algorithme. L'ordonnancement utilisé est l'algorithme FF étant donné qu'il présente les meilleurs résultats en termes de compromis entre les différents indicateurs (tableau 6.5). La même charge de travail est maintenue, mais la taille de la batterie et la production solaire sont modulées pour évaluer le comportement du GEC.

Les figures 6.19a et 6.19b sont un exemple d'expérience où la production solaire évolue d'un ordonnancement à l'autre avec une même capacité de batterie de 10 000 joules. À chaque début d'expérimentation la batterie est initialisée à 0 joules, c'est-à-dire qu'elle n'est pas chargée. Lorsque la production solaire diminue, la consommation du centre de calcul dépasse la production. Dans la figure 6.19a, la consommation de charge de travail à la mesure de 600 à 900 W et de 1500 à 1600 W est toujours presque supérieure à la courbe de production solaire. Cette mesure correspond à l'après-midi lorsque le soleil descend. La raison pour laquelle la courbe de charge de travail n'est pas égale à la production solaire, est la méthode d'ordonnancement d'une VM. L'algorithme vérifie à l'instant t s'il y a suffisamment d'énergie pour ordonnancer une VM, mais l'algorithme ne vérifie pas au temps $t + n$, si l'énergie sera toujours disponible. Certaines VMs ont donc été ordonnancées, peu avant la fin de la journée, où la production solaire commence à décliner.

La figure 6.19b présente la distribution des VMs avec la même charge de travail mais avec une production des panneaux solaires plus élevée. L'algorithme utilisé est toujours le FF. La différence avec l'ordonnancement précédent est l'utilisation des serveurs. Il y a suffisamment d'énergie solaire et de ressources vCPU pour exécuter les différentes VMs. Par conséquent, le serveur *S4* n'est que très peu sollicité.



(A) La distribution des VMs avec consommation de la charge de travail et la production de panneaux solaires pour l'algorithme FF

(B) La distribution des VMs pour l'algorithme FF mais avec la production de panneaux solaires (plus élevées)

FIGURE 6.19 – Exemple d'ordonnancement pour une même charge de travail dont la production solaire a évolué

Les deux expérimentations suivantes sont le résultat d'un ensemble de simulations pour lesquelles la taille de la batterie et la production solaire ont évolué. Elles nous ont aidé à :

1. connaître la meilleure corrélation entre la taille de la batterie et la production solaire où le GEC est le meilleur ;
2. identifier un palier où l'augmentation taille de la batterie n'a plus d'influence sur le GEC.

Les objectifs de performance de chaque dimensionnement est l'utilisation :

- **Dimensionnement 1** : d'un algorithme d'ordonnancement consciencieux de l'environnement ;
- **Dimensionnement 2** : d'un algorithme d'ordonnancement moins consciencieux de l'environnement mais utilisant plus la batterie.

6.4.1 Dimensionnement 1 : évolution du GEC sur une charge de travail agile

Dans cette partie, nous présentons un ensemble de simulation réalisée avec RenewSim. Chaque résultat de simulation est enregistré dans un historique, cela nous permet d'analyser et comparer chaque résultat.

Conditions initiales 1

Pour réaliser, l'ensemble des simulations nous avons :

- utilisé l'algorithme FF présenté dans le chapitre 5 ;
- initialisé la batterie à 0 joule ;

- augmenté la production solaire ≈ 2kWh d'une simulation à l'autre ;
- modulé la taille de la batterie de 40 000 joules 200 000 joules ;
- adopté la charge de travail de la partie précédente 6.3, c'est-à-dire 1000 VMs (600 courtes, 200 moyennes, 200 longues) ;
- disposé du même parc informatique de la partie précédente 6.3, c'est-à-dire 5 serveurs avec respectivement les capacités vCPU suivantes : 8, 8, 16, 16, 20 ;
- appliqué la même stratégie énergétique que dans la partie précédente 6.3.

Analyse 1

La figure 6.20 montre l'évolution du GEC lorsque la taille de la batterie et de la production solaire sont modifiées. Cela permet de comprendre la configuration optimale, où ces deux éléments obtiennent le meilleur résultat en terme de GEC. Lorsque la taille de la batterie et de la production des panneaux solaires augmentent de niveau (de 0 à 18 kWh pour la production des panneaux solaires et de 0 à 200 000 joules pour la batterie), cela augmente le GEC. Cependant, la batterie ne permet pas au réseau intelligent d'améliorer le GEC, car dans l'algorithme, la batterie est une sauvegarde pour la simulation et non une contrainte. Il n'apparaît pas dans l'équation du calcul du GEC. Ce n'est pas le facteur principal de son résultat.

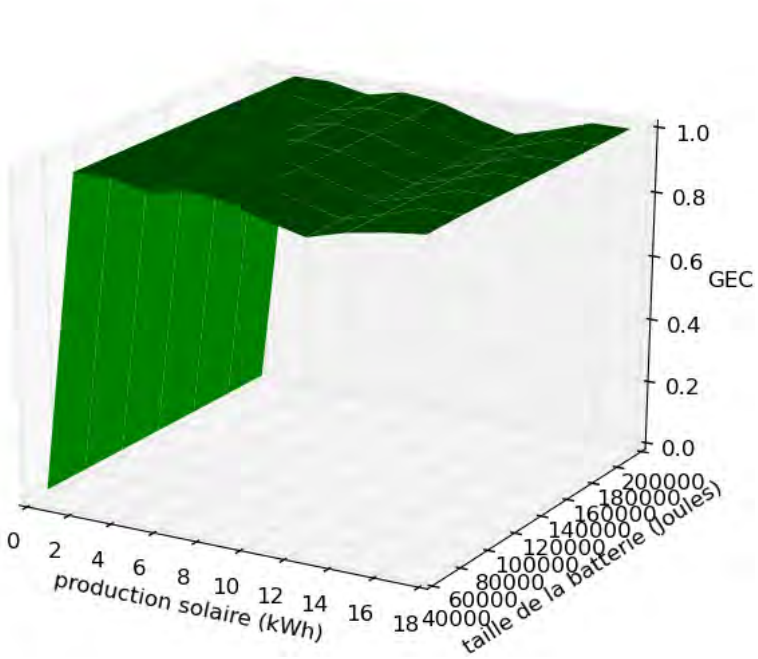


FIGURE 6.20 – Évolution du GEC par rapport à la production des panneaux solaires et à la taille de la batterie

La figure 6.21 est un exemple de la façon dont la batterie peut augmenter le GEC. Lorsque l'exécution d'une VM a débuté, et lorsque l'énergie solaire est suffisante, la VM continue même s'il n'y a plus d'énergie verte. S'il y a suffisamment d'énergie dans la batterie, il n'est pas nécessaire d'acheter de l'énergie au fournisseur, cet achat rentre dans le calcul du GEC, de sorte que le résultat du GEC sera meilleur.
Le GEC augmente rapidement en raison de son calcul. S'il n'y a pas de VMs ordonnancées et un peu de panneau solaire, le GEC sera égal à 1 parce que le réseau intelligent n'achète pas d'électricité. Le module de contrôle vendra uniquement et stockera de l'énergie.
Dans ce cas, la batterie a une faible influence sur le calcul du GEC, étant donné que ce n'est que durant quelques heures, où le réseau intelligent se retrouve dans le cas de la figure 6.21 et le module de contrôle achète quand même de l'électricité en raison de la puissance maximale de la batterie. Il ne suffit pas de fournir à l'instant t toutes les VMs pour obtenir un meilleur GEC.

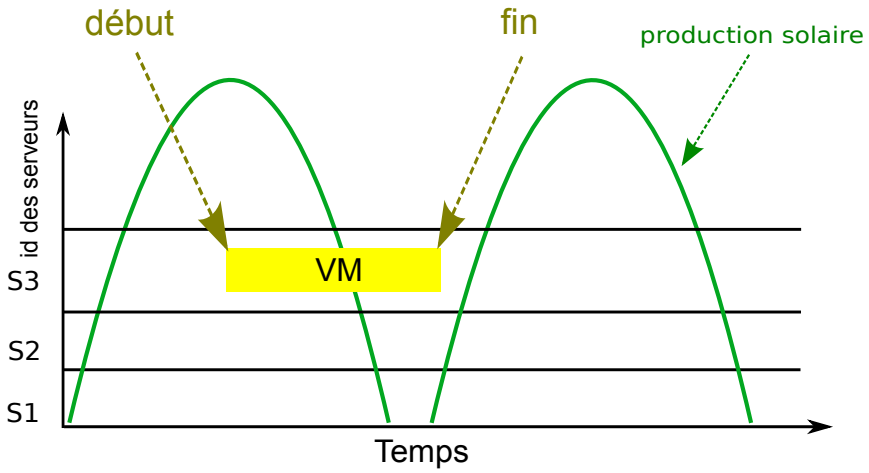


FIGURE 6.21 – Exemple de levier de prise de décision pour la batterie pour le GEC

6.4.2 Dimensionnement 2 : Évolution du GEC sur une charge de travail constante

Le premier dimensionnement ne permet pas de voir l'évolution du GEC car même si la majorité des VMs sont éjectées et que l'achat d'énergie est faible, sa valeur peut être proche ou égal à 1. Pour palier à cela, il a été utilisé un autre algorithme d'ordonnancement.

Conditions initiales 2

L'algorithme FF_2 présenté ci-dessous est basé sur l'algorithme FF dont la contrainte énergie a été supprimée. La seul contrainte est la disponibilité des serveurs en terme de vCPU.

Algorithme 8 FF_2

```
while VM < nbVmsOrdonnancee do
  for tps=0;tps<fenetreOrdonnancement;tps++ do
    if serveurs[s][tps] == vrai then
      VM ⇒ serveur[s][tps]
      if serveurPlein == vrai then
        serveurs[s + 1]
      end if
    end if
  end for
end while
```

La figure 6.22 est un exemple d'ordonnancement pour l'algorithme FF_2. Cette illustration montre les VMs programmées dans le temps, avec un pic de production de panneaux solaires de près de 700 W et une capacité de batterie égale à 10 000 joules. Le nombre de serveurs utilisés est inférieur à l'algorithme FF puisque les VMs possèdent assez de vCPU pendant 48 heures sur les différentes machines. La charge de travail diminue dans le temps car le nombre de VMs ordonnancées est plus faible dans le temps. Le serveur *S3* ne possède une charge de travail que le premier jour.

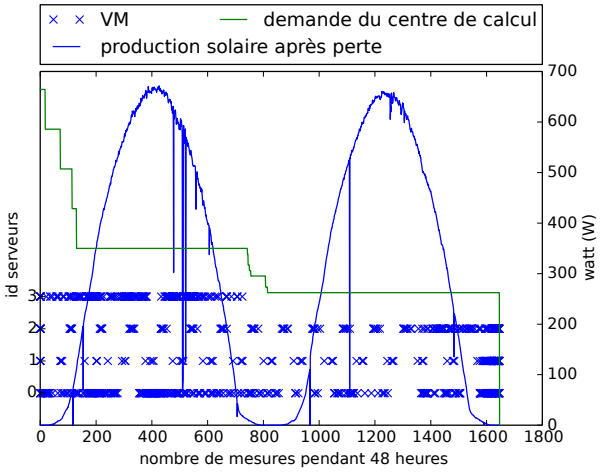


FIGURE 6.22 – La distribution des VMs avec consommation de la charge de travail et la production de panneaux solaires pour l'algorithme FF_2

Les conditions initiales sont identiques à l'expérimentation précédente 6.4.1 sauf sur l'utilisation de l'algorithme. Comme dans le dimensionnement précédent, la même charge de travail est maintenue, mais la taille de la batterie et la taille du panneau solaire sont modulées pour évaluer le comportement du GEC.

Analyse 2

La figure 6.23 montre l'évolution du GEC. Il est meilleur lorsque la production solaire augmente. L'influence de la taille de la batterie apparaît au moment où la production solaire commence à atteindre 10kWh. Effectivement, plus la batterie est grande plus le GEC augmente. Cela permet d'acheter de moins en moins chez le fournisseur d'électricité. Quand la production solaire est de 18 kWh, la batterie a une influence prononcée sur le GEC.

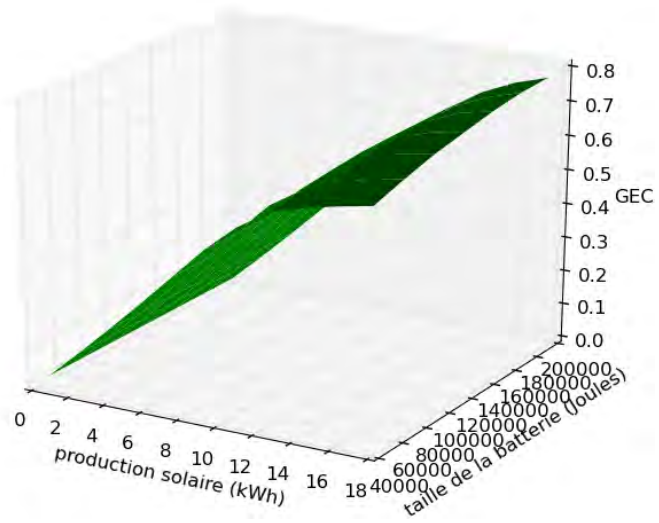


FIGURE 6.23 – Évolution du GEC par rapport à la production des panneaux solaires et à la taille de la batterie

Le tableau 6.6 présente certaines valeurs du GEC :

- **Pour une production solaire de 0kWh** : le GEC est à 0 ceci s'explique par la seule utilisation d'énergie non verte et le fonctionnement de la batterie. Au début de chaque simulation, elle est vide et ne peut se charger que pendant la durée de la simulation. Dans ce cas, il n'y a pas de production solaire, donc pas de chargement.
- **Pour une production solaire de 8kWh** : le GEC est le même pour n'importe quelle taille de batterie. Dans ce cas, la batterie n'a pas d'influence sur le GEC.
- **Pour une production solaire de 10kWh et 12kWh** : le GEC augmente légèrement avec la taille de la batterie.
- **Pour une production solaire de 16kwh et 18kWh** : le GEC augmente de façon significative avec la taille de la batterie. Dans ces cas là, la taille de la batterie a réellement une influence sur l'indicateur.

Tableau 6.6 – GEC en fonction de la taille de la batterie et de la production solaire

Batterie \ Solaire	0 kWh	8kWh	10kWh	12kWh	16 kWh	18 kWh
50 000 joules	0	0.45	0.53	0.59	0.64	0.65
100 000 joules	0	0.45	0.55	0.62	0.69	0.71
200 000 joules	0	0.45	0.56	0.63	0.73	0.76

6.4.3 Bilan du dimensionnement

Dans cette partie, il a été présenté l'influence de la taille de batterie et de la production solaire et leur corrélation, sur le GEC. Dans notre cas, l'indicateur obtient de meilleurs résultats si l'algorithme utilisé a un objectif de performance consciencieux de l'environnement (Partie 6.4.1). Cependant, cet objectif ne met pas en avant l'utilisation de la batterie. Dans le deuxième dimensionnement (Partie 6.4.2), la batterie a un rôle d'optimisation d'énergie verte, cela s'explique par l'augmentation du GEC à partir d'une association batterie/production solaire élevée.
Les résultats de dimensionnement ne sont utilisés que sur une charge de travail donnée. Une perspective de simulation pourrait porter sur la charge de travail, par exemple, en modulant le nombre de VMs longues/moyennes/courtes pour obtenir le nombre de serveurs adéquat.

6.5 Synthèses et conclusion des expérimentations

Le simulateur RenewSim nous a permis de réaliser l'ensemble des expériences présentées dans ce chapitre. Les objectifs atteints par ces expérimentations sont :

- l'utilisation du module de contrôle pour créer des politiques de décisions sur l'utilisation de l'énergie renouvelable ;
- l'utilisation d'un ordonnanceur afin de mettre en avant les flux énergétiques et informatiques ;
- la création d'une charge de travail agile.

Le travail de validation s'est appuyé sur des données réelles. Nous avons utilisé la production solaire émanant des panneaux solaires du bâtiment ADREAM [4]. Puis, nous avons généré un ensemble de traces synthétiques provenant des traces Google [8] et présentée dans le chapitre 4. Les expérimentations réalisées ont été faisables grâce au simulateur RenewSim car contrairement aux autres simulateurs décrits dans le chapitre 2, RenewSim permet :

- l'utilisation de différents modèles d'optimisation et pas seulement réduire l'aspect économique ;
- la communication entre le réseau physique électrique et la demande du centre de calcul ;

4. https://www.laas.fr/public/fr/adream

- l'application d'un ensemble de modèles électriques ou d'ordonnancements et de les comparer entre eux via des métriques ;
- la co-simulation via sa facilité de s'intégrer avec des simulateurs dédiés.

Dans la partie 6.2, la stratégie énergétique a été modélisée à travers un modèle présenté dans le chapitre 3. Dans ces expériences, les sources et destinations sont dotées d'une notation pour connaître leur ordre d'utilisation les unes par rapport aux autres. Cette modélisation en matrice permet de prendre une politique de décision pour l'utilisation des sources hétérogènes. Les modèles présentés sont polymorphes, c'est-à-dire, qu'ils peuvent être étendus avec d'autres composants. Ils sont applicables pour diverses architectures de réseau intelligent. À l'avenir, le modèle de représentation est apte à inclure des modèles de prévisions et/ou des modèles d'apprentissage.

Dans la partie 6.3, il a été introduit une approche d'ordonnancement permettant de corréler les différents types de flux entre eux afin de prendre une décision. Ces expériences ont permis de réaliser une charge de travail agile, d'analyser les résultats et de les comparer. De plus, les algorithmes utilisés ont permis d'obtenir une base d'outils pouvant être améliorer en ajoutant, par exemple la QoS comme une contrainte des différents algorithmes. Pour atteindre ce résultat, nous pouvons utiliser les priorités et les caractéristiques de latence proposées par Google.

Dans la partie 6.4, la dernière expérience réalisée a analysé le comportement d'un indicateur (GEC) en fonction de la taille des éléments d'entrées du simulateur (batterie et production solaire). Cela a permis de connaître les capacités des éléments à s'intégrer dans une dynamique permettant d'utiliser au mieux l'énergie verte. Une évolution possible de ce dimensionnement est de connaître les capacités des éléments en modulant la charge de travail, en terme de caractéristiques de VMs utilisées (longues, courtes, moyennes).

À travers l'ensemble de ces expérimentations, l'utilisation d'algorithmes et de modèles nous aide à répondre à des spécifications attendues par l'administrateur du centre de calcul. La figure 6.24 est une illustration de la manipulation des algorithmes.

Par rapport à ce que nous avons annoncé dans le chapitre 3 partie 3.1.4, nous répondons à des objectifs du réseau intelligent. Chaque algorithme certifie un objectif :

- **QoS** : assurer la qualité de service : pour ordonnancer des VMs (nombre de VMs ordonnancées, ou favoriser les VMs longues) ;
- **Environnemental** : favoriser l'utilisation d'énergie renouvelable ;
- **Lucratif** : rendre le réseau intelligent lucratif (vente d'énergie renouvelable).

Le tableau 6.7 donne une appréciation à chaque algorithme en fonction de la spécification attendue :

- + : acceptable ;
- ++ : favorable ;
- +++ : très favorable.

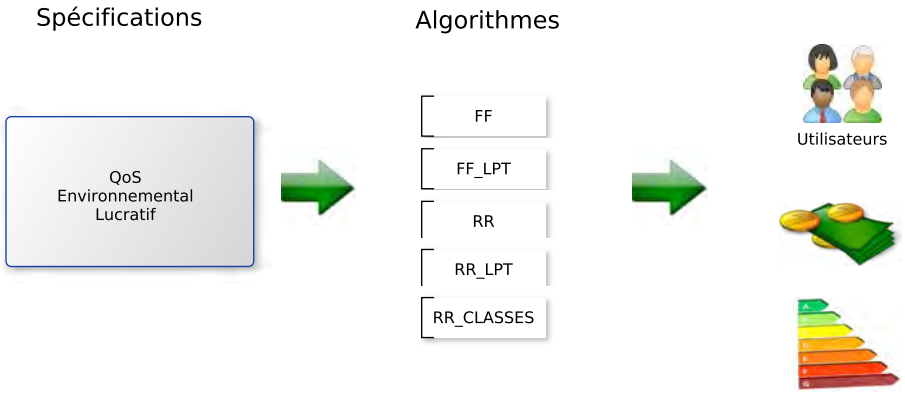


FIGURE 6.24 – Illustration de la prise de décision par rapport à une spécification

TABLEAU 6.7 – Les politiques de décisions en fonction des algorithmes

Objectif		RR	FF	RR_CLASSES	RR_LPT	FF_LPT
QoS	Nombre de VMs ordonnancées	++	+++	++	+	+
	VMs longues	+	+	++	+++	+++
Environnemental	Utilisation de l'énergie renouvelable	++	++	+++	+	+
Lucratif	Vente d'énergie renouvelable	++	++	+++	++	+

Le chapitre 7 suivant conclu ce manuscrit de thèse. Il résume l'ensemble des contributions et expérimentations réalisées dans ces 6 chapitres. Puis, il énonce des perspectives de recherches pouvant compléter et étendre les propositions d'algorithmes, de modèles et de méthodes présentées dans ces travaux de thèse.

Chapitre 7

Conclusions et perspectives

Sommaire

Le *Cloud Computing* est présent dans beaucoup d'applications que nous utilisons au quotidien via les réseaux sociaux, les services de *cloud* à la demande, ou même à travers une simple recherche dans un navigateur. Leur coût de fonctionnement est de plus en plus onéreux. Les fournisseurs de services distribués actuels, se penchent sur l'étude d'infrastructures moins onéreuses, sur des problématiques d'optimisations de réseaux informatiques, de transports d'énergie, et d'utilisations d'énergies renouvelables.

Dans cette thèse, nous nous sommes intéressés :

- aux architectures de réseau intelligent pour l'alimentation de centres de calcul via des énergies vertes ;
- aux charges de travail de centres de calcul à grande échelle ;
- aux algorithmes d'ordonnancement permettant de prendre en compte à la fois les contraintes énergétiques et informatiques pour tendre vers une charge de travail agile ;
- aux dimensionnements des espaces de stockage répondant à une production d'énergie renouvelable et une charge de travail donnée.

Ce chapitre dresse dans un première partie le bilan des différentes contributions de cette thèse. Dans une deuxième partie sont présentés les bilans des expérimentations et des simulations. Enfin dans une troisième partie, les perspectives de ces travaux de thèse sont exposées.

7.1 Bilan des contributions

La création d'un simulateur dédié à l'alimentation d'un centre de calcul via plusieurs sources d'énergie - chapitre 3

Dans ces travaux de thèse, nous avons proposé RenewSim. Il permet de recréer un réseau intelligent polymorphe. Il offre la possibilité de modéliser son propre réseau via des modèles, de brancher d'autres simulateurs pour des parties définies du réseau. Le verrou principal levé par RenewSim est son module de contrôle. Celui-ci autorise la communication entre les flux énergétiques et informatiques, pour prendre une décision commune, et plus consciencieuse, de l'environnement.
Le simulateur RenewSim a été utilisé pour :

- La conception du modèle de décisions présentée dans le chapitre 3 ;
- La validation d'algorithmes présentée dans le chapitre 5 ;
- La modélisation d'un environnement polymorphe et configurable dans le chapitre 6 ;
- L'analyse du dimensionnement de la batterie par rapport à la production d'énergie verte, via un des simulations réalisées dans le chapitre 6.

L'analyse de traces de centre de calcul à grande échelle pour la création de traces synthétiques - chapitre 4

Dans cette thèse, nous avons abordé la problématique de la gestion d'une charge de travail dans un centre de calcul. Pour répondre à cette problématique, nous nous sommes intéressés à un ensemble de traces [8], ainsi qu'à la création de traces synthétiques. La plupart des traces mises à disposition ne se focalisent pas sur l'aspect consommation et en particulier, avec quels types d'énergies (renouvelable ou non) elles sont alimentées.
Dans cette contribution, nous avons analysé le fonctionnement de traces sous divers angles :

- le cycle de vie d'une tâche ;
- le nombre de tâches sur une période donnée ;
- le temps d'exécution d'une tâche ;
- les ressources informatiques utilisées pour une tâche ;
- la QoS d'une tâche.

Cette analyse a permis de mettre en avant l'hétérogénéité de la charge de travail parce qu'elle propose un ensemble de tâches ayant à la fois des temps d'exécution longs et courts. Cependant, cette charge de travail ne prends pas en compte la consommation d'énergie et de surcroît l'aspect énergie verte. Le schéma de ces traces ne permettait pas le calcul de la consommation d'une tâche. Les traces synthétiques proposées permettent de garder cette hétérogénéité, et d'encapsuler une tâche dans une VM. Mais via cette synthèse des traces, on perd la notion de services composés permettant de calculer la consommation d'une charge à un instant donné. Cependant, la VM garde l'idée de composition pour de futures traces. De plus, à travers l'étude de ces traces, on

peut mettre en avant le contexte applicatif de ces traces et générer des charges de travail plus génériques et indépendantes de la notion de consommation [87].

Proposition d'utilisation, d'adaptation et de simulation d'algorithme d'ordonnancement sous contraintes - chapitre 5

Dans ces travaux de thèse, plusieurs algorithmes sont utilisés et adaptés. Ils possèdent les mêmes types d'entrées afin de les rendre comparables et mesurables entre eux. Les contraintes sont les mêmes, et correspondent à la place disponible en termes de VCPU pour chaque serveur et l'énergie produite disponible. Ces algorithmes nous permettent d'obtenir une charge de travail agile et modulable. Cependant, les algorithmes ont besoin de prédictions météorologiques afin de réaliser un ordonnancement. Actuellement, l'ordonnancement ne s'adapte pas en fonction de la production réelle. L'objectif de cette thèse n'est pas nécessairement la définition d'un algorithme de placement optimal, mais plutôt améliorer les moyens permettant de le faire. Ces algorithmes ont mis en avant l'importance d'utilisation des informations énergétiques pour l'ordonnanceur, afin de lui même prendre une décision sur son centre de calcul. De plus, l'utilisation d'algorithmes et de modèles nous aident à répondre à des spécifications attendues par l'administrateur du centre de calcul. Chaque algorithme est plus ou moins favorable à une politique de décision par rapport à la QoS attendue, l'aspect environnemental ou l'aspect lucratif.

Dimensionnement d'un stockage énergétique par rapport à une production d'énergie verte - chapitres 3, 6

À travers l'analyse des architectures présentées dans le chapitre de l'état de l'art dans la partie 2.1, peu de travaux se portent sur la corrélation entre la production d'énergie verte, et la capacité d'une batterie. Dans ces travaux de thèse, nous avons essayé de comprendre l'impact énergétique, en fonction de la production de panneaux solaires et de la taille disponible dans un espace de stockage. Pour savoir si la combinaison des deux composants était adéquate, nous avons adapté un indicateur, le GEC. Ce dernier permet de quantifier la part d'énergie renouvelable consommée par un centre de calcul. Les objectifs étaient :

- d'analyser l'impact de la batterie sur l'empreinte carbone ;
- de connaître le seuil de capacité où la batterie n'a plus d'influence sur le GEC ;
- de mettre en évidence la meilleure association : production solaire totale et capacité de la batterie.

Des expérimentations ont été réalisées dans le chapitre 6, dans la partie 6.4.

7.2 Bilan des expérimentations

Dans le chapitre 6 est présenté un ensemble d'expérimentations réalisées à l'aide du simulateur RenewSim :

- des politiques de décisions à travers un modèle de description des données (cf. partie 6.2) ;
- l'évaluation des algorithmes utilisés dans cette thèse (cf. partie 6.3) ;
- des dimensionnements de la taille de la batterie par rapport à la production solaire (cf. partie 6.4).

Politiques de décisions à travers un modèle de description des données
L'objectif de cette expérimentation est de présenter un modèle pour représenter chaque élément d'un réseau intelligent et améliorer la prise de décision dans RenewSim. Ce modèle permet de résoudre les conflits de priorités entre les différentes sources énergétiques, et de réaliser des politiques de décisions. À l'aide de deux scénarios distincts, le module de contrôle peut prendre une décision dans un but :

- **le modèle environnemental** : utiliser plus d'énergie renouvelable et l'espace de stockage pour alimenter le centre de calcul ;
- **le modèle lucratif** : réduire la facture d'électricité.

Évaluation et comparaison des algorithmes utilisés
Dans cette expérimentation, il a été introduit une approche d'ordonnancement écologique d'une charge de travail via l'ensemble des algorithmes présentés dans le chapitre 5. Les résultats des expérimentations montrent la validation des algorithmes dont le but était la création d'une charge de travail agile, c'est-à-dire, d'obtenir une corrélation entre la production d'énergie verte et la consommation du centre de calcul. Les expériences ont été réalisées via une production solaire. Cependant, les algorithmes peuvent s'adapter à n'importe quelle production d'énergie verte.

Dimensionnement d'espace de stockage pour un centre de calcul alimenté par des énergies solaires
Dans cette expérience, il a été présenté l'influence de la taille de batterie et de la production solaire et leur corrélation, sur l'indicateur GEC. Malgré, l'utilisation de la batterie la production d'énergie solaire reste le facteur le plus important du GEC. Cependant, la batterie permet d'obtenir un taux légèrement supérieur. Ces expérimentations ont été réalisées via des concepts d'algorithmes présentés dans le chapitre 5.

La prochaine section introduit les perspectives futures de ces travaux de thèse.

7.3 Perspectives

Les contributions proposées au cours de cette thèse, ont amené à faire certains choix sur la modélisation, et les approches adoptées. Plusieurs points de développement semblent intéressants, ils sont classés du plus court au long terme :

1. l'amélioration des modèles de conversion du réseau intelligent en vue de diminuer les pertes énergétiques ;
2. l'ordonnancement à partir de la prédiction d'énergie verte et de la consolidation pour corriger les erreurs liées à la production réelle ;
3. l'utilisation d'heuristiques d'ordonnancements non gloutonnes ;
4. le branchement de simulateurs de différents domaines à RenewSim ;
5. la simulation dans un environnement réel.

Les modèles de conversion

Dans nos expériences, les pertes générées par les modules de conversions sont des pourcentages. Ces modules ont pour but de simuler la conversion des différentes tensions électriques des composants du réseau intelligent, pour se connecter au bus électrique commun.

Point de discussion 1 : il existe deux types de conversion, courant continu vers courant continu (*DC/DC*) et courant continu vers courant alternatif (*DC/AC*). Le *DC/DC* est la conversion qui génère le moins de perte. De plus, suivant le type de technologie de convertisseur utilisé, la perte générée est moindre.

Point de discussion 2 : l'utilisation de *DC/DC* peut obliger tout les éléments a être de type courant continu, y compris le centre de calcul. Certains travaux portent sur l'utilisation de panneaux solaires DC [72, 99].

La prédiction d'énergie

Dans nos expériences, nous utilisons une production solaire réelle pour prendre une décision. D'autres méthodes prennent des décisions à l'avance afin d'anticiper les bonnes ou mauvaises productions d'énergie renouvelables. Certains travaux portent sur la prédiction d'énergie [97, 98]. Cependant, la prévision d'énergie sur des longues fenêtres reste incertaine.

Dans nos travaux de thèse, un système de prédiction météorologique pourrait être ajouté au module de contrôle. Ce dernier aiderait à la prise de décision d'ordonnancement sur le centre de calcul et par la suite, comparer la production réelle à la prédiction, pour réajuster l'ordonnancement sur le centre de calcul. Pour optimiser ce système, nous pouvons utiliser des principes de reconfigurations dynamiques [40] via par exemple, de la migration en temps réel. Une fois

la reconfiguration réalisée, il nous est possible de créer un historique de décisions réalisées en fonction d'une charge de travail et une prédiction, pour apprendre des erreurs. Ceci nous amène à la perspective suivante.

Les heuristiques non gloutonnes

L'utilisation de méta-heuristiques permet contrairement aux algorithmes gloutons, de résoudre un problème d'optimisation dans des calculs de temps encore raisonnables. Si ses heuristiques sont bien paramétrées, elles obtiennent de meilleurs résultats que des algorithmes gloutons, étant donné qu'elles explorent plusieurs espaces de solutions.

Dans ces travaux de thèse, nous avons utilisé un algorithme glouton pour illustrer le lien entre les flux informatiques et énergétiques. Cela nous a permis d'obtenir une charge de travail agile de manière rapide. Cependant, l'utilisation de méta-heuristiques plus sophistiquées, nous donnerait :

- une prise en compte plus importante de *QoS* ;
- un ordonnancement moins consommateur.

Approche 1 : la création d'un algorithme multi-objectifs. Dans nos simulations, nous n'avons pas utilisée la QoS comme contrainte. Celles-ci peuvent se rajouter en utilisant les caractéristiques des tâches Google présentées dans le chapitre 4. Dans le modèle de traces synthétiques, nous avons gardé la priorité et la sensibilité à latence sans nous en servir. Le but d'un tel algorithme serait d'utiliser ces caractéristiques pour ordonnancer majoritairement les tâches ayant une priorité élevée et une sensibilité à latence élevée. De plus, comme l'algorithme RR_CLASSES présenté dans le chapitre 5, nous pourrions allouer certains serveurs dont les ressources VCPU sont les plus grandes pour satisfaire ces tâches et ne pas les surcharger avec des tâches de priorités plus faibles.

Approche 2 : la création d'un algorithme génétique (GA). Ces capacités sont plus puissantes. Il nous permettrait de donner une note (fonction objective) à un placement. Dans notre cas la fonction objective doit permettre de :

- minimiser la consommation générale du centre de calcul ;
- minimiser l'achat d'énergie aux fournisseurs ;
- maximiser la vente d'énergie aux fournisseurs ;
- maximiser l'utilisation de l'énergie verte.

Plusieurs problématiques sont soulevées par un tel algorithme. Tout d'abord, il y a beaucoup de paramètres. Cela augmente le temps d'exécution de l'algorithme. Si les prises de décisions tardent à arriver, la météo et la production d'énergie verte peuvent déjà avoir changée. L'inclusion de la batterie dans la fonction objective est complexe et arbitraire. Faut-il minimiser sa décharge ou maximiser sa décharge ? La représentation du chromosome du GA peut porter à confusion

pour la représentation du temps et de l'énergie renouvelable. Effectivement, un chromosome à un gène qui offre deux caractérisations possibles : *la représentation du gène* par exemple, une machine physique et *la valeur du gène*, par exemple, l'identification de la machine physique. Comment faire apparaître ces notions dans le chromosome ? la valeur d'un gène peut-il être le résultat d'une autre heuristique de placement ? Cependant, certains des objectifs peuvent être en conflits tels que la minimisation de la consommation du centre de calcul et la maximisation de l'utilisation de l'énergie verte. Le choix de mettre la maximisation de l'énergie verte, peut signifier, que nous ne voulons pas consommer moins.

Approche 3 : l'utilisation du principe de machine learning. Ce type d'algorithme permet d'apprendre des décisions passées pour en réaliser de meilleures dans le présent. Quelques travaux portent dessus dans l'ordonnancement de charges [100]. Nous pouvons utiliser une approche de *machine learning* afin de prendre des décisions sur la gestion du réseau intelligent. Par exemple, à une charge de travail donnée, et une prévision météo donnée, il faut aller chercher dans le passé les décisions réalisées sur ces données d'entrées. Ensuite, si cette décision s'est avérée juste dans le passé, la réaliser sinon, prendre une autre décision et être capable par la suite d'analyser ses résultats pour l'utiliser dans d'autres situations.

L'utilisation du DVFS

Le DVFS peut être utilisé dans la gestion des centres de calcul [101, 39] pour la gestion de la tension des machines physiques. Il change dynamiquement la fréquence des CPUs des machines physiques en fonction de leurs utilisations. L'avantage de cet outil est la possibilité de diminuer la consommation en ralentissant l'exécution des VMs.

Approche 1 : Les propriétés des VMs comme levier d'utilisation du DVFS. Dans nos travaux, nous n'avons pas utilisé les caractéristiques des priorités et de la sensibilité à la latence dans les VMs. Dans le cadre de l'utilisation du DVFS, les VMs ayant ses caractéristiques faibles, pourraient constituer une classe. Les serveurs tournant au ralenti pourraient leurs êtres alloués.

Approche 2 : Les prédictions météorologiques comme levier d'utilisation du DVFS. L'utilisation du DVFS dans nos travaux permettrait de diminuer la tension électrique des serveurs du centre de calcul en cas de pénurie d'énergie. Cela peut être couplé à de la prédiction, et anticiper les pertes d'énergie pour accumuler des réserves. Par exemple, la figure 7.1 est une illustration de l'utilisation du DVFS lorsqu'une prédiction météorologique explique une perte de 10% de production d'énergie renouvelable pendant 2 heures. Les serveurs commenceront à tourner au ralenti, dès l'alerte météorologique, pour déjà commencer à économiser de l'énergie.

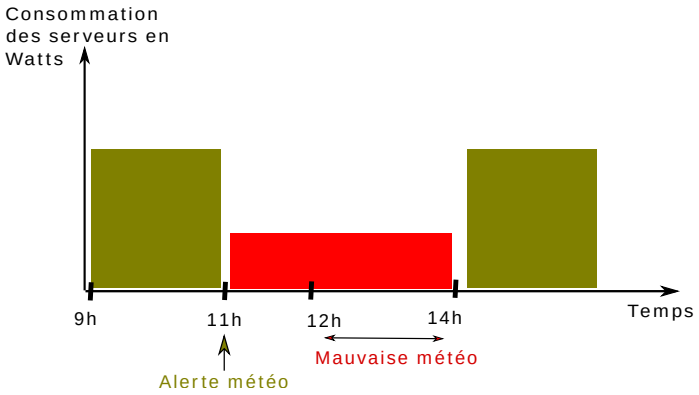


FIGURE 7.1 – Illustration d'utilisation du DVFS pour un serveur alimenté par des sources vertes

Approche 3 : la connexion d'un autre simulateur permettant le DVFS à RenewSim. Ceci nous amène à la perspective suivante.

La co-simulation par une approche pluridisciplinaire

RenewSim est un simulateur dont les spécifications sont décrites à haut niveau. Ce niveau d'abstraction permet la communication avec d'autres simulateurs. Par exemple, dans le chapitre 2, dans la partie 2.3, sont présentés plusieurs simulateurs. Le simulateur DCWorms [6] offre un ensemble de module pour la gestion des prévisions météorologiques. En utilisant ces prévisions nous serons donc capables de prendre des décisions sur une fenêtre d'ordonnancement et améliorer nos algorithmes. La co-simulation permettra d'améliorer et d'utiliser des modèles plus précis dans certains domaines. Par exemple, l'utilisation d'un simulateur pour la charge/décharge de la batterie avec des modèles, venant de la communauté scientifique du domaine, beaucoup plus précis et adaptés aux types de batteries utilisés (lithium, plomb).

La simulation en environnement réel

La simulation en milieu réelle permettra de valider l'analyse et le comportement d'un réseau intelligent alimentant un centre de calcul via des énergies renouvelables. Dans l'environnement réel, l'ensemble des modèles de batteries, d'algorithmes d'ordonnancement, ou de prévisions solaires, peuvent être testés et ajustés. Cependant, cette perspective arriverait en processus de fin et une fois l'ensemble des simulations bien testé car une mauvaise gestion d'une puissance pourrait entraîner par exemple des sur-tensions et détériorer le matériel composant le réseau intelligent.

L'utilisation d'algorithmes types « Machine Learning »pour améliorer la prise de décision dans un réseau intelligent

Couplé à d'autres approches énoncées précédemment, les algorithmes d'apprentissages pourraient améliorer la prise de décision dans de tels réseaux intelligents. La politique de décision peut émaner d'une multitude de paramètres dans un réseau intelligent pour l'alimentation d'un centre de calcul car les éléments qui le compose, s'influencent entre eux. En addition des différentes perspectives présentées, les sujets intéressants sont :

- l'étude du comportement des éléments du réseau intelligent entre eux, afin d'établir un historique ;
- le couplage des politiques de « Machine Learning » à la décision intrinsèque des différents éléments du réseau.

Publications Scientifiques

Conférences et workshops internationaux

- de Courchelle, I., Monteil, T., Labit, Y., Guerout, T. (2016, Juillet). **A data model for supplying a Data Center with several energy sources**. *In workshop WSSC* [80].
- Da Costa, G., Grange, L., De Courchelle, I. (2016, Novembre). **Modeling and Generating large-scale Google-like Workload**. *In Green and Sustainable Computing Conference*. IEEE [87]

- de Courchelle, I., Roose, P., Dalmau, M., Etcheverry, P. (2015, Février). **Txupito : An interactor component model for ambient HCI**. *In Pervasive and Embedded Computing and Communication Systems (PECCS), 2015 International Conference* IEEE.
- Karchoud, R., Roose, P., Dalmau, M., de Courchelle, I., Dibon, P. (2015, Mars). **Kalimucho for smart-$*$: One step towards eternal applications**. *In Industrial Technology (ICIT), 2015 IEEE International Conference*

Présentation de Poster

- **Journée neOCampus** - poster et présentation orale *2015 - 2016 -2017*
- **Salon MIDINOV** - poster - *Labège en 2014*

Bibliographie

[1] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and greenswitch : Managing datacenters powered by renewable energy," in *ACM SIGARCH Computer Architecture News*, vol. 41, pp. 51–64, ACM, 2013.

[2] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud : a new architecture for green data center," in *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pp. 29–38, ACM, 2009.

[3] Y. Li, A.-C. Orgerie, and J.-M. Menaud, "Opportunistic scheduling in clouds partially powered by green energy," in *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pp. 448–455, IEEE, 2015.

[4] T. Guérout, S. Medjiah, G. Da Costa, and T. Monteil, "Quality of service modeling for green scheduling in clouds," *Sustainable Computing : Informatics and Systems*, vol. 4, no. 4, pp. 225–240, 2014.

[5] Y. Kessaci, N. Melab, and E.-G. Talbi, "Optimisation multi-critère pour l'allocation de ressources sur clouds distribués avec prise en compte de l'énergie," in *Rencontres Scientifiques France Grilles 2011*, 2011.

[6] K. Kurowski, A. Oleksiak, W. Piatek, T. Piontek, A. Przybyszewski, and J. Weglarz, "Dcworms–a tool for simulation of energy efficiency in distributed computing infrastructures," *Simulation Modelling Practice and Theory*, vol. 39, pp. 135–151, 2013.

[7] C. de régulation de l'énergie. http://www.smartgrids-cre.fr/index.php?p=definition-smart-grids.

[8] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces : format + schema," technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2012.03.20. Posted at http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[9] B. Lab. https://eta.lbl.gov/publications/united-states-data-center-energy-usag, 2016.

[10] Google. https://environment.google/approach/#/intro/infographics-1.

[11] A. Energia. http://www.acciona-energia.com/pressroom/news/2017/january/acciona-covers-googles-electricity-consumption-chile-renewable-energy/, 2017.

[12] H. Lei, T. Zhang, Y. Liu, Y. Zha, and X. Zhu, "Sgeess : Smart green energy-efficient scheduling strategy with dynamic electricity price for data center," *Journal of Systems and Software*, vol. 108, pp. 23–38, 2015.

[13] D. S. Palasamudram, R. K. Sitaraman, B. Urgaonkar, and R. Urgaonkar, "Using batteries to reduce the power costs of internet-scale distributed networks," in *Proceedings of the Third ACM Symposium on Cloud Computing*, p. 11, ACM, 2012.

[14] V. Kontorinis, J. Sampson, L. E. Zhang, B. Aksanli, H. Homayoun, T. S. Rosing, and D. M. Tullsen, *Battery Provisioning and Associated Costs for Data Center Power Capping*. Department of Computer Science and Engineering, University of California, San Diego, 2012.

[15] S. K. Garg, C. S. Yeo, and R. Buyya, "Green cloud framework for improving carbon efficiency of clouds," in *European Conference on Parallel Processing*, pp. 491–502, Springer, 2011.

[16] N. Beldiceanu, B. D. Feris, P. Gravey, S. Hasan, C. Jard, T. Ledoux, Y. Li, D. Lime, G. Madi-Wamba, J.-M. Menaud, *et al.*, "Towards energy-proportional clouds partially powered by renewable energy," *Computing*, vol. 99, no. 1, pp. 3–22, 2017.

[17] T. Guérout and M. B. Alaya, "Autonomic energy-aware tasks scheduling," in *Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*, pp. 119–124, IEEE, 2013.

[18] F. D. Sacerdoti, M. J. Katz, M. L. Massie, and D. E. Culler, "Wide area cluster monitoring with ganglia," in *null*, p. 289, IEEE, 2003.

[19] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system : design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[20] M. C. Doug Cutting. http://hadoop.apache.org/.

[21] X. U. Manual. http://bits.xensource.com/Xen/docs/user.pdf, 2008.

[22] C. Li, W. Zhang, C.-B. Cho, and T. Li, "Solarcore : Solar energy driven multi-core architecture power management," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 205–216, IEEE, 2011.

[23] C. Stewart and K. Shen, "Some joules are more precious than others : Managing renewable energy in the datacenter," in *Proceedings of the workshop on power aware computing and systems*, pp. 15–19, IEEE, 2009.

[24] Y. Zhang, Y. Wang, and X. Wang, "Greenware : Greening cloud-scale data centers to maximize the use of renewable energy," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 143–164, Springer, 2011.

[25] S. Bird, A. Achuthan, O. A. Maatallah, W. Hu, K. Janoyan, A. Kwasinski, J. Matthews, D. Mayhew, J. Owen, and P. Marzocca, "Distributed (green) data centers : A new concept for energy, computing, and telecommunications," *Energy for Sustainable Development*, vol. 19, pp. 83–91, 2014.

[26] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pp. 233–244, ACM, 2011.

[27] H. Li, W. Zhang, and D. Xu, "High-reliability long-backup-time super ups with multiple energy sources," in *Energy Conversion Congress and Exposition (ECCE), 2013 IEEE*, pp. 4926–4933, IEEE, 2013.

[28] W. Choi, P. Enjeti, and J. W. Howze, "Fuel cell powered ups systems : design considerations," in *Power Electronics Specialist Conference, 2003. PESC'03. 2003 IEEE 34th Annual*, vol. 1, pp. 385–390, IEEE, 2003.

[29] K. Neuhaus, J. Dulout, and C. Alonso, "Lvdc grid based on pv energy sources and multiple electrochemical storage technologies," in *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pp. 990–997, IEEE, 2016.

[30] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy, "Energy storage in datacenters : what, where, and how much ?," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, pp. 187–198, ACM, 2012.

[31] J. Shim, R. Kostecki, T. Richardson, X. Song, and K. A. Striebel, "Electrochemical analysis for cycle performance and capacity fading of a lithium-ion battery cycled at elevated temperature," *Journal of power sources*, vol. 112, no. 1, pp. 222–230, 2002.

[32] E. M. Erickson, F. Schipper, R. Tian, J.-Y. Shin, C. Erk, F. F. Chesneau, J. K. Lampert, B. Markovsky, and D. Aurbach, "Enhanced capacity and lower mean charge voltage of

li-rich cathodes for lithium ion batteries resulting from low-temperature electrochemical activation," *RSC Advances*, vol. 7, no. 12, pp. 7116–7121, 2017.

[33] M.-E. Choi, S.-W. Kim, and S.-W. Seo, "Energy management optimization in a battery/supercapacitor hybrid energy storage system," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 463–472, 2012.

[34] Y. Li, A.-C. Orgerie, and J.-M. Menaud, "Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a cloud data center," in *PDP 2017-25th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2017.

[35] J. Münzberg, S. Baum, and I. Stadler, "Economic evaluation, optimization and comparison of photovoltaic-battery-grid power supply system in single-and multi-family buildings with increasing share of renewable energy," in *Energy and Sustainability Conference (IESC), 2016 International*, pp. 1–11, IEEE, 2016.

[36] V. Villebonnet, G. Da Costa, L. Lefevre, J.-M. Pierson, and P. Stolf, "Towards generalizing" big little" for energy proportional hpc and cloud infrastructures," in *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*, pp. 703–710, IEEE, 2014.

[37] V. Villebonnet, G. Da Costa, L. Lefevre, J.-M. Pierson, and P. Stolf, ""big, medium, little" : Reaching energy proportionality with heterogeneous computing scheduler," *Parallel Processing Letters*, vol. 25, no. 03, p. 1541006, 2015.

[38] G. Da Costa and J.-M. Pierson, "Dvfs governor for hpc : Higher, faster, greener," in *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, pp. 533–540, IEEE, 2015.

[39] T. Guerout, T. Monteil, G. D. Costa, R. N. Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with dvfs," *Simulation Modelling Practice and Theory*, vol. 39, pp. 76 – 91, 2013. S.I.Energy efficiency in grids and clouds.

[40] V. Villebonnet, G. Da Costa, L. Lefevre, J.-M. Pierson, and P. Stolf, "Energy proportionality in heterogeneous data center supporting applications with variable load," in *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*, pp. 1023–1030, IEEE, 2016.

[41] L. Kleinrock, "Analysis of a time-shared processor," *Naval Research Logistics (NRL)*, vol. 11, no. 1, pp. 59–73, 1964.

[42] M. Yue, "A simple proof of the inequality ffd (l)$\leq$ 11/9 opt (l)+ 1,$\forall$ l for the ffd bin-packing algorithm," *Acta Mathematicae Applicatae Sinica (English Series)*, vol. 7, no. 4, pp. 321–331, 1991.

[43] A. Khosravi, S. K. Garg, and R. Buyya, "Energy and carbon-efficient placement of virtual machines in distributed cloud data centers," in *European Conference on Parallel Processing*, pp. 317–328, Springer, 2013.

[44] G. Grid, "The green grid data center power efficiency metrics : Pue and dcie," *Green Grid report*, 2007.

[45] J. H. Holland, "Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence," *Ann Arbor, MI : University of Michigan Press*, 1975.

[46] M. Srinivas and L. M. Patnaik, "Genetic algorithms : A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[47] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967–2982, 2014.

[48] T. Guérout, Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez, and T. Monteil, "Mixed integer linear programming for quality of service optimization in clouds," *Future Generation Computer Systems*, vol. 71, pp. 1–17, 2017.

[49] H. Lei, R. Wang, T. Zhang, Y. Liu, and Y. Zha, "A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center," *Computers & Operations Research*, vol. 75, pp. 103–117, 2016.

[50] J. Tu, L. Lu, M. Chen, and R. K. Sitaraman, "Dynamic provisioning in next-generation data centers with on-site power production," in *Proceedings of the fourth international conference on Future energy systems*, pp. 137–148, ACM, 2013.

[51] M. Nattaf, C. Artigues, and P. Lopez, "Programmation linéaire mixte et programmation par contraintes pour unprobì eme d'ordonnancement a contraintes energétiques," in *12e Journées Francophones de la Programmation par Contraintes (JFPC 2016)*, pp. 209–212, 2016.

[52] S. U. Ngueveu, C. Artigues, and P. Lopez, "Scheduling under a non-reversible energy source : An application of piecewise linear bounding of non-linear demand/cost functions," *Discrete Applied Mathematics*, vol. 208, pp. 98–113, 2016.

[53] N. B. Rizvandi, J. Taheri, A. Y. Zomaya, and Y. C. Lee, "Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 388–397, IEEE, 2010.

[54] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium*

on Networked Systems Design & Implementation-Volume 2, pp. 273–286, USENIX Association, 2005.

[55] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster computing*, vol. 16, no. 2, pp. 249–264, 2013.

[56] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 577–578, IEEE, 2010.

[57] A. Khosravi, A. Nadjaran Toosi, and R. Buyya, "Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers," *Concurrency and Computation : Practice and Experience*, 2017.

[58] H. Casanova, A. Legrand, and M. Quinson, "Simgrid : A generic framework for large-scale distributed experiments," in *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, pp. 126–131, april 2008.

[59] W. Liu, H. Li, W. Du, and F. Shi, "Energy-aware task clustering scheduling algorithm for heterogeneous clusters," in *Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications*, pp. 34–37, IEEE Computer Society, 2011.

[60] A. Lebre, A. Legrand, F. Suter, and P. Veyre, "Adding storage simulation capacities to the simgrid toolkit : Concepts, models, and api," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pp. 251–260, IEEE, 2015.

[61] J.-C. Charr, R. Couturier, A. Fanfakh, and A. Giersch, "Energy consumption reduction with dvfs for message passing iterative applications on heterogeneous architectures," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, pp. 922–931, IEEE, 2015.

[62] 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," *Software : Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[63] R. Buyya and M. Murshed, "Gridsim : A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation : practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[64] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenslot : scheduling energy consumption in green datacenters," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 20, ACM, 2011.

[65] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenhadoop : leveraging green energy in data-processing frameworks," in *Proceedings of the 7th ACM european conference on Computer Systems*, pp. 57–70, ACM, 2012.

[66] M. Brown and J. Renau, "Rerack : Power simulation for data centers with renewable energy generation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 3, pp. 77–81, 2011.

[67] S. Bak, M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek, and J. Waglarz, "Gssim–a tool for distributed computing experiments," *Scientific Programming*, vol. 19, no. 4, pp. 231–251, 2011.

[68] K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz, "Grid scheduling simulations with gssim," in *Parallel and Distributed Systems, 2007 International Conference on*, vol. 2, pp. 1–8, IEEE, 2007.

[69] M. Kocaoglu, D. Malak, and O. B. Akan, "Fundamentals of green communications and computing : Modeling and simulation," *Computer*, vol. 45, no. 9, pp. 40–46, 2012.

[70] B. Aksanli, J. Venkatesh, and T. Š. Rosing, "Using datacenter simulation to evaluate green energy integration," *Computer*, vol. 45, no. 9, pp. 56–64, 2012.

[71] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud : a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[72] J. Dulout, C. Alonso, L. Séguier, and B. Jammes, "Development of a photovoltaic low voltage dc microgrid for buildings with energy storage systems," in *ELECTRIMACS 2017*, vol. 2017, 2017.

[73] Y. Gu, X. Xiang, W. Li, and X. He, "Mode-adaptive decentralized control for renewable dc microgrid with enhanced reliability and flexibility," *IEEE Transactions on Power Electronics*, vol. 29, no. 9, pp. 5072–5080, 2014.

[74] U. Européenne. https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters.

[75] J. Singer, *Enabling Tomorrow's Electricity System : Report of the Ontario Smart Grid Forum*. Independent Electricity System Operator, 2010.

[76] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models.," *HotPower*, vol. 8, pp. 3–3, 2008.

[77] F. Quesnel, H. K. Mehta, and J.-M. Menaud, "Estimating the power consumption of an idle virtual machine," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pp. 268–275, IEEE, 2013.

[78] O. Perpinan, E. Lorenzo, and M. Castro, "On the calculation of energy produced by a pv grid-connected system," *Progress in Photovoltaics : research and applications*, vol. 15, no. 3, pp. 265–274, 2007.

[79] Y. Ru, J. Kleissl, and S. Martinez, "Storage size determination for grid-connected photovoltaic systems," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 68–81, 2013.

[80] I. De Courchelle, T. Monteil, Y. Labit, and T. Guerout, "A data model for supplying a Data Center with several energy sources," in *workshop WSSC*, (Toulouse, France), July 2016.

[81] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale : Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, p. 84, 2012.

[82] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale : Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, p. 7, ACM, 2012.

[83] F. Gbaguidi, S. Boumerdassi, É. Renault, and E. Ezin, "Characterizing servers workload in cloud datacenters," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pp. 657–661, IEEE, 2015.

[84] Z. Liu and S. Cho, "Characterizing machines and workloads on a google cluster," in *2012 41st International Conference on Parallel Processing Workshops*, pp. 397–403, IEEE, 2012.

[85] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in google cluster trace based on resource usage," *arXiv preprint arXiv :1501.01426*, 2015.

[86] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95*, vol. 94, 2010.

[87] G. Da Costa, L. Grange, and I. De Courchelle, "Modeling and Generating large-scale Google-like Workload," in *International Workshop on Resilience and/or Energy-aware techniques for High-Performance Computing* , (Hangzhou, China), Nov. 2016.

[88] Google. https://www.google.com/about/datacenters/efficiency/internal/.

[89] C. Garnier, M. Aggar, M. Banks, J. Dietrich, B. Shatten, M. Stutz, and E. Tong-Viet, "Data centre life cycle assessment guidelines," *The Green Grid, white paper*, vol. 45, p. v2, 2012.

[90] T. G. Grid. https://www.thegreengrid.org/en/newsroom/news-releases/global-leaders-industry-and-government-reach-agreement-measuring-data-center, 2012.

[91] K. Kritikos, B. Pernici, P. Plebani, C. Cappiello, M. Comuzzi, S. Benrernou, I. Brandic, A. Kertész, M. Parkin, and M. Carro, "A survey on service quality description," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 1, 2013.

[92] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing : a vision, architectural elements, and open challenges," *arXiv preprint arXiv :1006.0308*, 2010.

[93] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a consumer can measure elasticity for cloud platforms," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, pp. 85–96, ACM, 2012.

[94] F. D. Rossi, M. G. Xavier, C. A. De Rose, R. N. Calheiros, and R. Buyya, "E-eco : Performance-aware energy-efficient cloud data center orchestration," *Journal of Network and Computer Applications*, vol. 78, pp. 83–96, 2017.

[95] F. Alvarruiz, C. de Alfonso, M. Caballer, and V. Hern'ndez, "An energy manager for high performance computer clusters," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pp. 231–238, IEEE, 2012.

[96] M. Oswal, J. Paul, and R. Zhao, "A comparative study of lithium-ion batteries," *USA : University of Southen California*, 2010.

[97] E. Lorenz, J. Hurka, D. Heinemann, and H. G. Beyer, "Irradiance forecasting for the power prediction of grid-connected photovoltaic systems," *IEEE Journal of selected topics in applied earth observations and remote sensing*, vol. 2, no. 1, pp. 2–10, 2009.

[98] A. Dolara, S. Leva, and G. Manzolini, "Comparison of different physical models for pv power output prediction," *Solar Energy*, vol. 119, pp. 83–99, 2015.

[99] K. Sun, L. Zhang, Y. Xing, and J. M. Guerrero, "A distributed control strategy based on dc bus signaling for modular photovoltaic generation systems with battery energy storage," *IEEE Transactions on Power Electronics*, vol. 26, no. 10, pp. 3032–3045, 2011.

[100] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy, "Machine learning methods for solar radiation forecasting : A review," *Renewable Energy*, vol. 105, pp. 569–582, 2017.

[101] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pp. 1–6, IEEE, 2011.