Analyses et préconisations pour les centres de données virtualisés

Frédéric Dumont

Thèse de Doctorat

Frédéric DUMONT

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'École nationale supérieure des Mines de Nantes
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information, et mathématiques (STIM 503)

Discipline : Informatique, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)

Soutenue le 21 Septembre 2016
Thèse n° : 2016EMNA0249

Analyses et préconisations pour les centres de données virtualisés

JURY

Président :	**M. Noël De Palma**, Professeur HdR, Université Joseph Fourier Grenoble
Rapporteurs :	**Mme Vania Marangozova-Martin**, Maître de Conférences HdR, Université Joseph Fourier Grenoble
	M. Daniel Hagimont, Professeur HdR, INPT/ENSEEIHT Toulouse
Examinateurs :	**Mme Anne-Cécile Orgerie**, Chargée de Recherche CNRS, IRISA Rennes
	M. Noël De Palma, Professeur HdR, Université Joseph Fourier Grenoble
	M. Benoit Parrein, Maître de Conférences HdR, Polytech Nantes
Invité :	**M. Martin Dargent**, Président Directeur Général, Easyvirt
Directeur de thèse :	**M. Jean-Marc Menaud**, Professeur HdR, Mines de Nantes

Remerciements

Les travaux présentés dans ce manuscrit ont fait l'objet d'une thèse CIFRE entre la société Easyvirt et le Laboratoire d'Informatique de Nantes-Atlantique (LINA).

Tout d'abord, je tiens à remercier Jean-Marc Menaud, mon directeur de thèse mais avant tout un encadrant hors pair depuis 2010 (eh oui déjà !). Tu as été mon encadrant d'un module de Master 1 intitulé "Initiation à la Recherche", dans lequel j'ai fait mes premiers travaux dans le domaine de la virtualisation. Au départ, je ne savais pas trop si j'avais fait le bon choix mais j'ai continué l'aventure. Tu es devenu mon encadrant de stage de fin d'études de Master 2 puis lorsque j'étais Ingénieur R&D. J'ai ensuite intégré Easyvirt en 2013, tu es alors mon directeur de thèse. Merci pour ta disponibilité, tes conseils, tes remarques et de m'avoir aiguillé dans mes travaux. Sans toi cette thèse CIFRE n'aurait jamais vu le jour.

Merci également à Martin Dargent, Président Directeur Général d'Easyvirt, alors jeune start-up en 2013, d'avoir relevé le défi d'embaucher un doctorant CIFRE, de m'avoir fait confiance dans mon travail et d'avoir su concilier temps dédié à la thèse et celui à l'entreprise bien que cela n'ait pas toujours été simple.

Je souhaite aussi remercier les membres de mon jury : Vania Marangozova-Martin et Daniel Hagimont d'avoir accepté d'être rapporteur ainsi que Anne-Cécile Orgerie, Noël De Palma et Benoit Parrein d'avoir accepté de faire partie de ce jury. Merci à tous pour le temps consacré dans l'évaluation de mon travail.

J'adresse mes remerciements à mes collègues de bureau, Thierry Bernard et Ludovic Gaillard ainsi qu'à ceux de l'École des Mines de Nantes notamment Alexandre Garnier et Rémy Pottier, pour les bons moments partagés ensemble, que ce soit au bureau ou autour d'un (bon) café, à parler de sujets divers et variés, voir insensés et pour les blagues plus ou moins bonnes (surtout moins ...).

Et le meilleur pour la fin, merci à mes parents de m'avoir laissé faire le parcours que j'ai souhaité, même si ce dernier n'a pas toujours été exemplaire.

Table des matières

Liste des tableaux

Table des figures

1

Introduction

1.1 Contexte

Le Cloud Computing (l'informatique "dans les nuages") est un paradigme de l'informatique qui vise à mettre à disposition des services ou des infrastructures disposant de très grandes capacités de calculs et de stockages, facturés en fonction de la demande (modèle *pay-as-you-go*). Les différentes offres de Cloud Computing reposent principalement sur 3 niveaux : Infrastructure as a Service (IaaS), Plateform as a Service (PaaS) et Software as a Service (SaaS). Ainsi, les services offerts par le Cloud Computing peuvent aller de la mise à disposition d'applications en ligne (SaaS) jusqu'à la mise à disposition de ressources matérielles (IaaS). Au niveau Iaas, les fournisseurs de Cloud mettent à disposition ces ressources sous forme de machines virtuelles (VM). Pour l'utilisateur, les machines virtuelles sont en tout point identiques à des machines physiques, appelées "serveurs". La mise en place des machines virtuelles est possible grâce aux techniques de virtualisation permettant ainsi d'utiliser efficacement les ressources des serveurs (processeur, mémoire, E/S réseau, etc).

Aujourd'hui, les systèmes d'IaaS (VMware® VCenter, CloudStack, OpenStack, etc.) sont des produits matures et largement diffusés au sein des centres de données des entreprises. Pour le fournisseur de service IaaS, l'intérêt de proposer des machines virtuelles (VM) est double. Premièrement, il peut déployer des machines virtuelles dans des temps très courts, par rapport au déploiement d'un serveur physique. Deuxièmement, d'héberger sur un même serveur physique, plusieurs machines virtuelles et ainsi optimiser ses coût de fonctionnement et d'investissement. La conséquence de ce déploiement rapide et facile est que le nombre de machines virtuelles au sein de l'IaaS n'a cessé de croître, pour arriver, d'après une étude interne, à des taux de consolidation d'environ 3 machines virtuelles par cœur logique, soit environ 50 machines virtuelles pour un serveur de 16 cœurs logiques. Cette explosion du nombre de machines virtuelles incite les administrateurs à s'équiper d'outils d'analyses pour la gestion de leurs centaines à milliers de machines virtuelles.

Des outils, tels que VCOps de VMware® DCScope d'EasyVirt ou VM Turbo, analysent l'activité des machines virtuelles (principalement les ressources processeur et mémoire) dans l'objectif de déterminer leur comportement. L'analyse comportementale permet de détecter d'éventuels problèmes de performances, des problèmes de dimensionnement ou au contraire détecter des machines virtuelles probablement non inutilisées. Les solutions précédemment citées réalisent les mêmes analyses, faites à partir de comportements que nous définissons comme "pré-déterminés". En effet, ils sont "pré-déterminés" dans la mesure où il est difficile voir impossible de changer les paramètres de ces comportements. Par exemple, dans VCOps, une machine virtuelle est classée *idle* si son activité processeur et mémoire n'excède jamais 10% de ses capa-

cités. Définir un pourcentage fixe pour un centre de données dans lequel coexiste des serveurs physiques hétérogènes n'a pas de sens. Ces solutions ont également un deuxième inconvénient. Elles ne permettent pas de filtrer le bruit. Une machine virtuelle dispose toujours d'une activité (pics d'activités) même si elle n'est plus utilisée. Cette activité peut provenir du système d'exploitation (vérification/installation des mises à jour) ou d'une application ayant une activité temporaire (exécutions d'anti-virus). Il est important pour un administrateur de pouvoir paramétrer les comportements "pré-déterminés" en fonction de son centre et gérer la notion de bruits dans le but d'affiner la détection de comportements des machines virtuelles analysées. Il est nécessaire que l'outil d'analyses puisse déterminer également les machines virtuelles à comportement atypique, afin d'identifier facilement, dans un parc de plusieurs centaines de machines virtuelles, les machines virtuelles à surveiller. En effet, nous savons que globalement les machines virtuelles d'un même centre de données font principalement toutes plus ou moins la même chose. Une machine virtuelle est dite atypique si son profil (consommation ressources systèmes) s'éloigne des autres. Potentiellement, une machine virtuelle atypique peut s'expliquer par un fonctionnement normal mais unique de l'application qu'elle encapsule (analyseur de spams) mais peut également être provoqué par une application vérolée ou tombée en panne. Identifier de manière dynamique, rapide et automatique les machines virtuelles atypiques, permet des gains très important en terme de sûreté de fonctionnement d'un centre.

La surveillance des ressources systèmes consommées par les machines virtuelles sur un serveur physique est également un élément clé pour garantir le bon fonctionnement d'une plate-forme IaaS. Les métriques classiques sont le taux d'occupation processeur ou mémoire, les accès disques et réseaux ou la latence. Il existe de nombreux outils permettant de collecter ces métriques. Ils peuvent être classifiés suivant trois approches différentes et complémentaires. Les premiers utilisent des sondes systèmes au niveau de l'hyperviseur. Ces sondes, orientées serveur, permettent de collecter des données gros grain sur la consommation des ressources d'une ou de plusieurs machines virtuelles. Si ces sondes sont assez simples à mettre en œuvre, elles ne permettent pas de collecter la consommation ressource des processus s'exécutant dans la machine virtuelle. Ainsi, la seconde approche consiste à installer des sondes à l'intérieur des machines virtuelles. Ces sondes orientées processus accèdent, de la même manière que les sondes serveur, au métrique du système d'exploitation. Pour affiner l'analyse, la troisième approche consiste à instrumenter une application au sein d'une machine virtuelle en vue de récolter des métriques dans le but de s'assurer du bon fonctionnement de l'application (temps de réponse, latence applicative etc). Par rapport à un client d'une solution Cloud, l'installation de sonde peut être plus ou moins intrusive. Les sondes systèmes peuvent être déployées sans intrusion. Les sondes processus imposent l'installation dans la machine virtuelle du client d'un composant logiciel système. Les sondes applicatives peuvent nécessiter une réécriture de l'application du client. Dans de très nombreux contextes, essentiellement pour des raisons de sécurité et de maintenance, les clients ne souhaitent pas installer au sein de leurs machines virtuelles des composants logiciels non maîtrisés. De ce fait, les fournisseurs de Cloud ne peuvent installer que des sondes systèmes, perdant ainsi la finesse d'analyse que pourrait apporter des sondes processus ou applicatives.

1.2 Objectifs

L'objectif de cette thèse est de développer un système d'analyse avancée et d'optimisation d'infrastructures Cloud, allant de l'introspection à l'analyse comportementale des machines virtuelles. Dans ce but, cette thèse met en avant quatre sous-objectifs :

- **Métriques de performances avancées :** Mesurer et analyser l'activité des machines virtuelles et des serveurs est une préoccupation majeure des fournisseurs de Cloud, en vue de garantir les contrats de service négociés avec leurs clients. Malheureusement les métriques disponibles au niveau d'un IaaS sont d'assez gros grains et ne permettent pas une analyse fine des consommations ressources d'une machine virtuelle. L'analyse fine de l'activité des machines virtuelles repose alors sur une étude approfondie des métriques de performances. Pour une même ressource, le nombre de compteurs est important. Il faut alors bien comprendre la sémantique de chacun des compteurs dans le but de bien

choisir ceux représentant l'activité réelle de la machine virtuelle.

- **Introspection non intrusive :** Déployée au niveau de l'hyperviseur, les sondes systèmes non intrusives permettent de suivre l'activité des serveurs et des machines virtuelles. Ces sondes nous permettent ainsi de collecter les métriques de performances, principalement pour les ressources processeur et mémoire. La non intrusivité de ces sondes permet un déploiement sans contraintes chez les fournisseurs de Cloud dans le mesure où elles ne nécessitent aucun agent dans les machines virtuelles, respectant ainsi la sécurité et le bon fonctionnement des machines virtuelles des clients de solutions Cloud.

- **Analyses comportementales :** L'analyse fine des ressources systèmes peut être utilisée dans plusieurs contextes tels que la sécurité, la tolérance aux pannes, les fuites mémoires ou encore l'optimisation de l'infrastructure et sa consommation énergétique. Nous distinguons deux types d'analyses. La première recherche les machines virtuelles ayant des comportements pré-déterminés. Il s'agit de machines dont l'activité répond à des critères identifiés. Ce type d'analyse permet par exemple, de connaître rapidement les machines virtuelles "à risque", celles dont les consommations en ressources sont importantes. La seconde analyse recherche les machines virtuelles ayant des comportements atypiques. Il s'agit de rechercher automatiquement les machines virtuelles ayant un profil d'activités différent des autres.

- **Préconisations / Optimisations :** L'analyse des comportements pré-déterminés et atypiques permet à l'administrateur d'identifier des machines virtuelles sans activités (comportements *idle,lazy*), qui peuvent potentiellement être arrêtées. En arrêtant voir supprimant ces dernières, l'administrateur peut de ce fait libérer des ressources au sein du centre de données et donc éviter le rachat de nouveaux équipements. Cette analyse lui permet également d'identifier les machines virtuelles dont les capacités des ressources doivent être augmentées (*buzy, undersized*) ou au contraire diminuées (*oversized*). En redimensionnant les machines virtuelles, les gains en ressources peuvent être importants. En effet, le redimensionnement peut se faire sur 2 ressources : la ressource processeur, via l'affectation de vPCU (virtual CPU) et la ressource mémoire.

1.3 Contributions

Cette thèse répond aux différents objectifs précédemment cités grâce aux contributions suivantes :

- **Études des métriques de performances :** Nous avons commencé nos travaux par une étude approfondie des métriques utilisées dans VMware®. Dans premier temps, nous avons choisi de travailler sous VMware® dans le mesure où c'est le système d'IaaS le plus répandu dans les entreprises. Ce dernier dispose d'un nombre conséquent de compteurs et une gestion très sophistiquée de la mémoire (environ 50 compteurs rien que pour cette dernière). Il est donc indispensable de bien comprendre la différence entre ces compteurs afin de mieux les utiliser. Dans un second temps, nous avons travaillé sous KVM. Sur l'hyperviseur KVM, il est difficile d'obtenir précisément l'activité processeur et mémoire des machines virtuelles, surtout la mémoire. Nous avons alors étudié différentes librairies d'introspection et retenu LibVMi [XLXJ12, Pay12]. Une étude sur les structures du noyau Linux et sur les différents compteurs mémoire sous Linux ont été réalisées. Suite à l'étude des différentes métriques, nous avons développé nos outils d'introspection, non intrusifs. Le premier est développé à partir du Software Development Kit Java fourni par VMware®. Grâce à ce SDK, notre sonde se connecte à l'infrastructure et monitore l'activité des serveurs physiques et des machines virtuelles en fonction des métriques de performances précédemment retenues, en collectant les données directement depuis le VCenter. Le second outil est développé à partir de la librairie LibVMi. Il s'agit d'une extension à LibVMi pour capturer et surveiller, directement depuis l'hyperviseur, les processus s'exécutant au sein du système d'exploitation de chacune des machines virtuelles analysées.

- **Récoltes de traces réelles et développement d'un outil d'analyses comportementales :** Notre solution supervise les centres de données en collectant et en historisant les métriques de performances dans sa propre base de données. A partir de ces données, la solution permet d'analyser en temps réel et sur un créneau de temps donné des comportements pré-définis. Les comportements sont issus des spécifications et préconisations VMWare®. Cependant, afin d'obtenir une détection de comportements plus robuste, nous avons modifié les définitions de ces comportements de sorte à ajouter une notion de bruit. En effet, dans les définitions de base, la détection de comportement est stricte (100% des points doivent être supérieurs/inférieurs à un certain seuil). Cependant, une machine, bien que non utilisée dispose toujours d'une activité résiduelle. Cette dernière ne doit pas affecter l'analyse. Notre solution permet également de repérer les machines virtuelles atypiques, c'est à dire regrouper par similarité des ensembles de machines virtuelles puis de déterminer celles ne pouvant être regroupées avec les autres. Cette approche classique d'analyse de données s'appuie sur le *Clustering*. L'étude des algorithmes de partitionnement ne répondant pas à nos besoin, nous avons développé notre propre algorithme de partitionnement, spécifique, répondant à nos besoins, notamment insensible aux bruits.

1.4 Organisation du document

Ce document est organisé en 2 parties. La première partie présente le contexte de cette thèse. Dans le chapitre 2, nous abordons les thèmes du Cloud Computing et de la Virtualisation. Dans un premier temps, une présentation du Cloud Computing est réalisée. Nous le définissons et présentons les différents niveaux de services qu'offre le Cloud pour se focaliser ensuite au niveau Iaas (Infrastructure as a Service), niveau auquel porte les contributions de cette thèse. Dans un second temps, nous nous focalisons sur la virtualisation, la technologie clé sur laquelle repose le Cloud Computing. Cette partie permet de décrire le fonctionnement des hyperviseurs et d'introduire les machines virtuelles.
La deuxième partie présente les différentes contributions apportées autour de cette thèse. Le Chapitre 3 présente l'étude des métriques de performances dans lequel nous décrivons chaque compteur étudié ainsi que l'importance de l'étude de chacun d'entres-eux. Nous discutons également de notre contribution autour de l'introspection de machines virtuelles, consistant en l'ajout d'une extension à LibVMi dans l'objectif de mesurer finement, de manière non intrusive, les consommations en processeur et en mémoire des machines virtuelles s'exécutant au sein de l'hyperviseur KVM. Dans le Chapitre 4, nous discutons des différentes façons d'analyser les centres de données, de l'analyse avancée des métriques de performances à la détection comportementale au niveau du centre de données. Le Chapitre 5 comporte les évaluations de nos différentes contributions.
Pour finir, le Chapitre 6 clore cette thèse en revenant sur les différentes contributions apportées, met en avant l'apport de ces travaux dans la société ainsi que les perspectives liées à ces derniers.

1.5 Diffusion scientifique

Cette thèse a fait l'objet d'une publication internationale et d'une publication nationale. Dans [1] nous présentons nos travaux autour d'un outil d'aide à la décision permettant de détecter des machines virtuelles ayant un comportement atypique, basé sur un algorithme de partitionnement insensible aux bruits. Dans [2] nous détaillons le développement d'une sonde système, basée sur l'introspection non-intrusive, permettant de monitorer finement l'activité mémoire des machines vituelles s'exécutant sur un hyperviseur KVM. Des travaux en dehors de la présentation de cette thèse ont également été réalisés dans [3].

Conférence Internationale avec comité de lecture

[1] Frédéric Dumont, Jean-Marc Menaud. Synthesizing Realistic CloudWorkload Traces for Studying Dynamic Resource System Management. 2015 International Conference on Cloud Computing and

Big Data, Huangshan, China, 2015.

[3] Ahmed El Rheddane, Noel De Palma, Fabienne Boyer, Frédéric Dumont, Jean-Marc Menaud, Alain Tchana Dynamic scalability of a consolidation service. International Conference on Cloud Computing, Indus Track, pages 01–09, 2013.

Conférence Francophone avec comité de lecture

[2] Frédéric Dumont, Jean-Marc Menaud, Sylvie Laniepce. Monitoring de l'utilisation mémoire et CPU par processus, basé sur l'introspection de machines virtuelles. Conférence d'informatique en Parallélisme, Architecture et Système (Compas'2015), Lille, France, 2015.

I

Contexte

2

Le Cloud Computing et la Virtualisation

Dans ce chapitre, nous discutons des différents domaines dans lesquels portent les contributions de cette thèse. Nous y présentons le Cloud Computing ainsi que la Virtualisation, technologie sous-jacente au Cloud Computing.

Sommaire

CETTE thèse propose des outils d'analyses d'infrastructures virtualisées. Afin de disposer de connaissances complètes dans les domaines sur lesquels portent cette thèse, nous présentons dans ce chapitre une étude sur les domaines connexes à savoir, le Cloud Computing et la Virtualisation. Dans une première section, nous décrivons le Cloud Computing, concept apparu après l'explosion de l'usage d'Internet. Ensuite, dans une seconde section, nous décrivons la Virtualisation, technologie permettant d'optimiser l'utilisation des ressources d'un centre de données.

2.1 Le Cloud Computing ou l'informatique dans les nuages

Le terme Cloud Computing est employé pour la première fois publiquement en 2006 par Eric Schmidt, directeur général de Google®, lors de la conférence SES (Search Engine Strategies Conference & Expo)

[LZ14]. Le Cloud Computing est considéré comme une technologie qui a révolutionné l'usage d'Internet. A ses débuts, Internet était utilisé pour accéder facilement à l'information (via la consultation des sites web) ou faciliter la communication (messageries électroniques). Aujourd'hui, son utilisation est plus particulièrement tourné vers l'accès à des logiciels ou le stockage de données en ligne, appelés services, détournant l'usage classique (historique) de l'ordinateur personnel.

2.1.1 Définition

Le Cloud Computing est un terme généralement employé pour désigner le paradigme de gestion et de prestation de services à la demande sur Internet. Il décrit un modèle, représentant l'infrastructure informatique comme un "nuage", accessible de n'importe où dans le monde, offrant de la puissance de calculs, de l'espace de stockage ou encore des logiciels en ligne à la demande. Les clients paient uniquement les services et les ressources utilisées, en fonction d'accords de qualité de service (SLA pour Service Level Agreement). Un SLA est un contrat passé entre le client et le fournisseur de services dans lequel sont indiquées les modalités d'exécutions des tâches. Ainsi, ce contrat peut indiquer une échéance de terminaison de la disponibilité du service, des temps d'exécutions inférieurs à un certain seuil, un délai maximum de réponse, etc. Bien qu'il s'agisse d'un concept récent, dans la littérature, les définitions du Cloud Computing sont variées. Deux d'entres-elles, retenues dans les travaux de Saurabh Kumar Garg and Rajkumar Buyya [GB12] présentent le Cloud Computing de manières différentes. La première définition est focalisée sur la technique alors que la seconde est axée sur l'aspect financier.

Ainsi, le NIST (National Institute of Standards and Technology)[MG11] défini le Cloud computing comme étant un modèle permettant d'accéder à la demande à des ressources partagées (serveurs, réseaux, stockages, applications et services). Celles-ci pouvant être rapidement réservées et libérées en fonction des besoins. De plus, le NIST indique que ce modèle favorise la disponibilité et qu'il est composé de :

- 5 caractéristiques principales : disponibilité en libre-service à la demande, ressources mutualisées, élasticité rapide, accès réseau universel (un environnement de type Cloud nécessite et est accessible via le réseau de n'importe où), service mesurable et facturable (l'utilisation est mesurée et facturée au détail).

- 4 modèles de déploiement : privé (ressources appartenant uniquement à un client), communautaire (ressources appartenant à et partagées pour plusieurs clients), public (ressources partagées pour tout les monde) et hybride (à la fois privé et public).

- 3 niveaux de services : Infrastructure-as-a-Service (IaaS), Plateform-as-a-Service (PaaS) et Software-as-a-Service (SaaS)

Buyya et al.[BYV08], donnent une définition du Cloud computing non pas d'un point de vue technologique mais du point de vue client. Ainsi, le Cloud Computing est défini comme étant un modèle distribué orienté "marketing", "constitué d'un ensemble de serveurs virtualisés et interconnectés, dynamiquement provisionnés et présentés comme une ou plusieurs ressources informatiques unifiées, basées sur des SLA (Service Level Agreements) établis entre le fournisseur de services et les consommateurs". De ce fait, de nombreuses sociétés ne voient pas seulement le Cloud uniquement comme un "service à la demande" mais plutôt comme un marché.

Il n'y a pas de définition standard du Cloud Computing. En général, toutes les définitions sur le Cloud Computing semblent se concentrer uniquement sur certains aspects technologiques [BBG10].

2.1.2 Les niveaux de services

Comme mentionné précédemment, le Cloud Computing propose des services basés sur 3 modèles de services : Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), et Software-as-a-Service (SaaS).

2.1.2.1 IaaS - *Infrastructure-as-a-Service*

Les solutions de type IaaS fournissent aux clients des infrastructures basées sur des ressources physiques ou virtuelles. Ces ressources répondent aux exigences de l'utilisateur final en terme de quantité de mémoire, nombre de processeurs, capacité de stockage et réseau. L'utilisateur final dispose ainsi d'une infrastructure répondant à ses besoins et dispose du contrôle total sur cette dernière. Il peut ainsi installer, configurer et exploiter son système d'exploitation et ses applications. Les clients sont facturés en fonction de l'utilisation des ressources. Des ressources gérées et hébergées dans des centres de données. Le fournisseur d'IaaS public le plus populaire est Amazon Elastic Compute Cloud (EC2). Il met à disposition une infrastructure permettant de déployer facilement et rapidement des serveurs sous forme de machines virtuelles. Amazon propose également un service de stockage, Amazon Simple Storage Service (S3) permettant l'accès à un espace de stockage disponible depuis n'importe où. Un certains nombre de solutions de type IaaS open-source ont été développées afin de créer des cloud privés, comme CloudStack [KJM+14], OpenStack [SAE12] ou encore OpenNebula [MLM11]. Toujours dans le domaine du cloud privés, les solutions commerciales les plus diffusées dans les entreprises sont VMware® VCenter et Microsoft® Hyper-V. Parfois l'IaaS est également appelée HaaS (Hardware-as-a-Service).

2.1.2.2 PaaS - *Platform-as-a-Service*

Les solutions de type PaaS fournissent aux clients des plate-formes de développement (ainsi que des API) sur lesquelles les clients peuvent développer et déployer leurs propres applications. Ainsi l'utilisateur final dispose d'une plate-forme de travail prête à l'emploi, ce qui lui permet un gain de temps dans la mesure où il n'a pas à gérer son environnement. Les fournisseurs de PaaS les plus populaires sont Google AppEngine et Microsoft Azure. Google AppEngine est une plate-forme pour développer des applications Web hébergées dans les centres de données gérés par Google. Il propose des API qui permettent aux développeurs de tirer parti des services proposés par Google, tels que Maps ou Datastore. Google offre un service gratuit mais limité. De la même façon, Microsoft Azure met à disposition des plate-formes de développements ainsi qu'un SDK permettant de développer des services tirant parti du framework .NET.

2.1.2.3 SaaS - *Software-as-a-Service*

Il s'agit d'applications hébergées en ligne et donc accessibles depuis Internet. Contrairement aux solution PaaS qui permettent aux clients de déployer leurs applications personnalisées, les solutions Saas offrent des applications prêtes à l'emploi. Les applications sont alors gérées directement par le fournisseur et non l'utilisateur. De plus, les utilisateurs peuvent y accéder facilement depuis n'importe quelle plate-forme ou environnement. En effet, il n'y a plus besoin d'installer et d'exécuter l'application sur l'ordinateur client. Les solutions SaaS ont pour avantage d'être moins chères à l'achat et ne disposent pas de coût de maintenance. Les solutions SaaS les plus populaires sont les offres Google Apps (tels que Maps, Mail, Docs) ou encore Salesforce.

Les contributions présentées dans cette thèse portent au niveau IaaS. Dans la suite de ce chapitre, nous nous focalisons sur les différents systèmes d'IaaS.

2.1.3 Les systèmes IaaS

L'Infrastructure-as-a-Service est la couche la plus basse du Cloud Computing. Au niveau IaaS, un client a la possibilité de louer des ressources informatiques telles que des processeurs, de la quantité de mémoire, du stockage et du réseau. Ces ressources sont mises à disposition des clients sous forme de machines virtuelles, environnements d'exécution virtualisés. Les utilisateurs peuvent ensuite exécuter leurs applications au sein de ces machines virtuelles comme s'ils disposent de machines physiques. Ainsi, les systèmes IaaS fournissent des ressources matérielles (puissance de calculs, stockage) sans avoir à héberger et à maintenir le

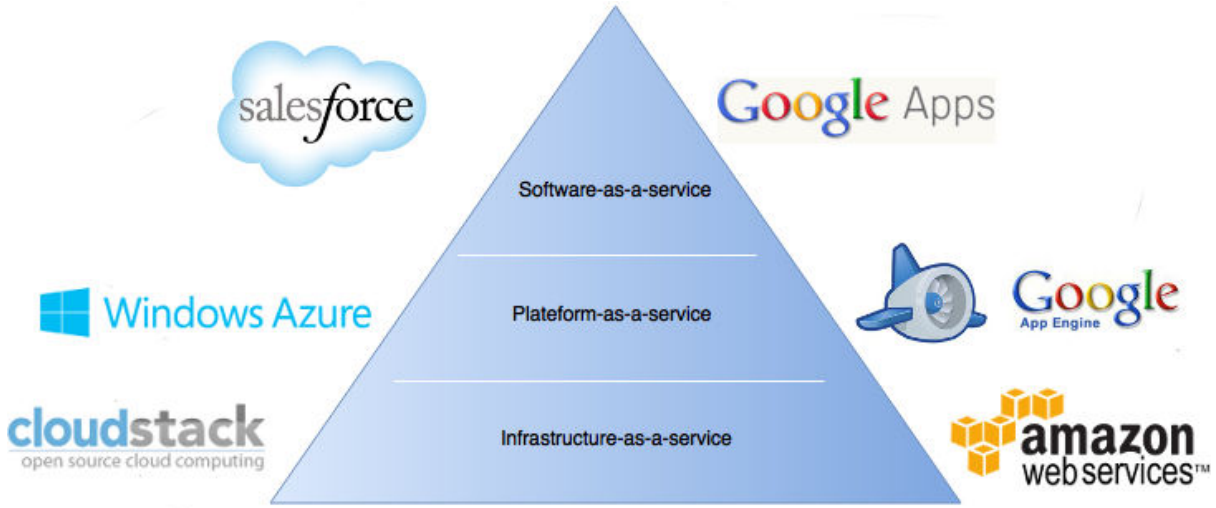


FIGURE 2.1 : Typologie du Cloud Computing

dit matériel, permettant aux utilisateurs des gains aussi bien financiers (pas de matériels à acheter, à mettre à jour) que de temps (pas de temps à passer à réinstaller ou mettre à jour une machine). Les technologies de virtualisation telles que Xen, KVM, ou VMware ESX servent de fondations pour permettre la virtualisation des serveurs. Ces derniers, appelés hyperviseurs, permettent la virtualisation des serveurs et optimiser l'utilisation des ressources des centres de données.

2.1.3.1 Les caractéristiques

L'Infrastructure-as-a-Service dispose de plusieurs caractéristiques :

- accessibilité : accéder, contrôler et gérer l'infrastructure facilement, de n'importe où, sans installation matériels ou logicielles spécifiques.
- élasticité : augmenter ou réduire dynamiquement les besoins en ressources. Ces ressources provisionnées sont disponibles immédiatement.
- disponibilité : accéder aux ressources sans tenir compte de la possibilité d'une panne matérielle.
- optimisation : les ressources matérielles sont partagées entre les différents clients, permettant d'optimiser l'utilisation des capacités matérielles. Le partage des ressources physiques entre les clients est réalisé de manière dynamique et optimisé.
- gestion : les clients doivent être en mesure de contrôler pleinement les ressources qui leurs sont allouées. Côté IaaS, ces ressources sont réservées par le biais de processus automatisés, contrôlant les disponibilités des ressources matérielles. Des techniques d'auto-gestion permettent d'assurer la gestion efficace des ressources de manière dynamique (sans interaction humaine) via des algorithmes et/ou mécanismes capables de reconfigurer dynamiquement l'infrastructure tels que DPM (Dynamic Power Management) ou DRS (Dynamic Resource Scheduling).
- interopérabilité : l'utilisation des ressources virtuelles est enregistrée et un système de facturation détermine à partir des données de chaque client le montant à facturer.

2.1.3.2 Le monitoring

Un des principaux problèmes dans ce modèle de services est la gestion des ressources. En effet, les centres de données continuent de croître sans cesse rendant la gestion de l'infrastructure de plus en plus complexe.

Dans ce contexte, il est nécessaire de mesurer finement et précisément l'activité de son infrastructure. Le monitoring est un processus d'une grande importance, aussi bien du côté du fournisseur (provider) que du client. Le monitoring fournit des informations ainsi que des métriques de performances (KPI pour Key Performance Indicator). Monitorer le Cloud permet au fournisseur et aux clients de mesurer la charge de travail (workload), suivre la performance de leurs machines physiques et virtuelles et ainsi suivre la qualité de service offerte par le fournisseur. C'est alors un élément important pour le suivi des SLA (délai, disponibilité), contractualisées entre le fournisseur et le client.

L'IaaS implique de nombreux services pour lesquels le monitoring est une tâche primordiale :

- Performance : Les performances sont primordiales surtout s'il s'agit de machines virtuelles exécutant des tâches critiques telles que des calculs scientifiques. Le monitoring des ressources permet de s'assurer des bonnes performances des machines virtuelles. De plus, dans le cadre des SLA, le monitoring permet aux fournisseurs d'IaaS et aux clients de savoir si les engagements sont bien respectés. Le monitoring permet également de détecter d'éventuels problèmes de performances (Troubleshooting en anglais). Dans une infrastructure complexe, il est difficile de déterminer rapidement la source du problème (matériel, logiciel) et quel composant est responsable (réseau, disque, etc). Un monitoring fin et complet est donc nécessaire à la fois pour les fournisseurs d'IaaS et les utilisateurs afin de localiser si le problème provient de l'infrastructure, du réseau ou de l'application cliente.

- Sécurité : La sécurité est une préoccupation majeure pour les fournisseurs et les utilisateurs de solutions IaaS. En effet, c'est une des raisons pour laquelle les entreprises ne souhaitent pas délocaliser entièrement leurs données dans le Cloud. La sécurité porte aussi bien sur l'architecture physique en elle-même (sécurisation du site, chiffrement des données stockées, confidentialité) que sur les machines virtuelles. La virtualisation apporte une première couche de sécurité dans la mesure où ce sont des environnements isolés. Chaque machine virtuelle est isolée des autres. Si une machine virtuelle est vérolée, les autres machines virtuelles n'ayant pas connaissance des autres ne seront pas impactées. Cependant, ces dernières étant accessibles depuis Internet, elles peuvent quand même être la cible d'attaques malveillantes. Le monitoring des machines virtuelles est également utilisé dans le domaine de la sécurité, par exemple systèmes de détection d'intrusion (IDS) [GR+03, PCSL08] ou recherche de processus malveillants [DGPL11, HN08].

- Planification des capacités/ressources : (Capacity Planning en anglais) Afin de garantir le bon fonctionnement des prochaines machines virtuelles déployées, il est nécessaire pour un fournisseur d'IaaS de pouvoir simuler leur ajout afin de déterminer si l'infrastructure est capable ou non en l'état de supporter ces nouvelles machines virtuelles. La planification des capacités permet ainsi de simuler les nouvelles machines virtuelles et permet de quantifier si besoin l'achat de nouveaux matériels. S'il est simple de connaître l'allocation des ressources virtuelles qui sera faite aux machines virtuelles, il est en revanche plus difficile de simuler l'utilisation des ressources de ces dites machines. Grâce au monitoring, il est possible de planifier leur utilisation en se basant sur des profils de consommations de machines virtuelles et appliquer ces profils aux nouvelles machines virtuelles. A l'inverse la planification des capacités et ressources permet de simuler la suppression de machines virtuelles ou de machines physiques afin de déterminer les capacités libérées dans le premier cas ou simuler une éventuelle panne ou indisponibilité dans le second cas. La planification est donc importante car elle permet de garantir le bon fonctionnement de l'infrastructure sur le long terme mais également de gérer efficacement les capacités afin de réduire les coûts (réduction de l'achat de machines, réduction de la consommation énergétique).

2.1.4 La virtualisation, la technologie du Cloud Computing

La virtualisation est la technologie pouvant être décrite comme une technologie clé sur laquelle repose le Cloud Computing. Toutes applications déployées dans le Cloud sont exécutées au sein de plate-formes

virtuelles, communément appelées machines virtuelles. La virtualisation permet la séparation de la couche matérielle et logicielle. Au sein d'une machine virtuelle, il n' y a aucune connaissance de l'architecture sous-jacente, architecture matérielle sur laquelle elle s'exécute. L'hyperviseur joue le rôle de passerelle entre la couche matérielle physique et les machines virtuelles. L'infrastructure informatique traditionnelle exécute plusieurs applications et consacre un serveur physique à chaque tâche. Cette approche est clairement coûteuse et peut être pas la meilleure utilisation des ressources dans la mesure où la tâche ne consomme pas la totalité des capacité des ressources 100% du temps. En effet, The Green Grid [BL09] montre dans son étude que les serveurs sont sous-utilisés. Le taux d'utilisation processeur moyen est de 5% avec un maximum de 10%. Dans cette étude 85% des serveurs ont un taux d'utilisation CPU inférieur ou égal à 6%. Cette même étude montre qu'en fonction des capacités et utilisations des serveurs physiques, une consolidation de 10 :1 est possible, sans surcharge processeur. C'est-à-dire qu'un serveur physique est capable de supporter la charge de 10 autres serveurs. Cette consolidation est possible en utilisant la virtualisation. Une autre étude [KT15], plus récente, montre qu'un tiers des serveurs est inutilisé. Ce pourcentage représente potentiellement 10 millions de serveurs dans le monde, portant la virtualisation ou non. Lié à cette sous-utilisation des serveurs physiques, le coût énergétique peut également devenir un énorme problème dans les centres de données de grandes tailles. Les serveurs sont certes inutilisés mais consomment toujours de l'énergie, augmentant le gaspillage énergétique.

Les différentes études mettent donc en avant l'imparfaite exploitation des centres de données. La virtualisation permet de mutualiser plusieurs systèmes physiques sur une seule et même machine, sous forme de machines virtuelles. Elle réduit alors le nombre de serveurs physiques au sein des centres de données et permet de réduire le coût lié à l'inutilisation des ressources.

Dans la section suivante nous nous intéressons à la virtualisation, technologie maîtresse du Cloud Computing.

2.2 La Virtualisation

Le concept de la virtualisation de serveur est apparu pour la première fois dans les années 1960, introduit par IBM [CB05] dont l'objectif était de partitionner efficacement des environnements matériels (mainframe). Ce partitionnement permet de faire cohabiter différents systèmes d'exploitation sur un seul et même ordinateur pouvant être accédés de façon concurrente. Pour cela, IBM a développé une couche logicielle entre le matériel et les systèmes d'exploitation. Cette couche est nommée VMM pour Virtual Machine Monitor (moniteur de machines virtuelles) [RG05] ou hyperviseur [Gol74]. La VMM découple la couche matérielle de la couche applicative (logicielle) en formant une couche d'abstraction. Les systèmes, s'exécutant au dessus de cette couche, accèdent à la couche matérielle comme si ces dernières s'exécutaient nativement sur le matériel.

2.2.1 Définition

La virtualisation [Gol74] consiste en l'exécution parallèle de plusieurs systèmes différents sur une même machine physique, nommé serveur. Chaque machine virtuelle (VM) aussi appelée instance, dispose de son propre système d'exploitation et applications. Elles sont indépendantes et isolées les unes des autres. De ce fait, si pour une raison particulière un service s'exécutant au sein d'une machine virtuelle devenait instable, cette dernière n'impactera pas le bon fonctionnement des autres services s'exécutant au sein des autres machines virtuelles, elles-mêmes s'exécutant à côté de cette dernière. Cette isolation apporte une sécurité pour le système d'exploitation et des applications qui y sont installées. La virtualisation permet ainsi l'exécution d'applications développées pour des systèmes d'exploitation différents sur la même machine. Ainsi, la virtualisation optimise l'utilisation des ressources matérielles du serveur. En effet, en mutualisant les services exécutés sur un même serveur, les ressources de ce derniers sont mieux exploitées. Sans la virtualisation, le serveur ne peut exécuter qu'un seul système d'exploitation. En général, pour des raisons

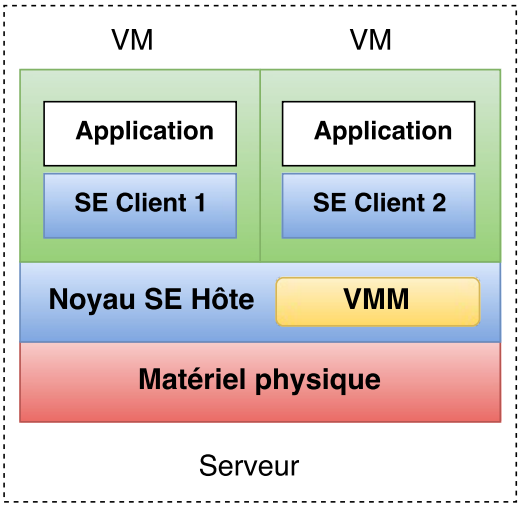


FIGURE 2.2 : Architecture hyperviseur de type 1

de sécurité et de fonctionnement, une seule application était déployée sur le système. Une application ne consommant pas la totalité des capacités du serveur, des ressources matérielles étaient alors inexploitées.

En établissant des allocations processeurs et mémoires précises aux machines virtuelles, l'utilisation des ressources matérielles des serveurs physiques est alors optimisée au maximum. La virtualisation dispose ainsi de plusieurs avantages [Men05] :

- sécurité : Les machines virtuelles hébergées sur le serveur sont isolées.

- disponibilité : Chaque machine dispose de son propre système qui fonctionne indépendamment du status des autres machines. Si une machine virtuelle a besoin d'être redémarrée, cette dernière peut l'être sans interférer sur les autres machines. De plus, le déploiement de machines virtuelles est simple et rapide. Elles peuvent également être migrées d'un serveur à un autre si besoin.

- coût : En mutualisant les services, la virtualisation permet de réduire le nombre de machines physiques mises en place dans les centre de données. Ainsi réduire le nombre de machines physiques permet de réduire les coûts d'acquisition de matériel, les coûts de maintenance, l'encombrement (la diminution du nombre de machines physiques permet un gain de place dans les centres de données) et indirectement les coûts d'énergie.

A l'inverse, la virtualisation dispose également d'inconvénients :

- administration : La facilité de déploiement de machines virtuelles a pour conséquence une explosion de machines virtuelles créées. Ainsi le nombre de serveurs virtuelles à gérer est beaucoup plus important.

- pannes matériels : Si un serveur physique est défaillant, ce sont tous les serveurs virtuels hébergés sur ce serveurs qui sont impactés.

- dégradations des performances : Une application s'exécutant à l'intérieur d'une machine virtuelle n'a pas les mêmes performances que si elle était exécutée sur une machine non virtualisée (natif). Ces dégradations de performances, notamment surcharges processeur, sont causées par la VMM [HLAP06]. Des travaux ont mis en avant des techniques cherchant à obtenir des performances proches du natif [WCC+08, YAMV15].

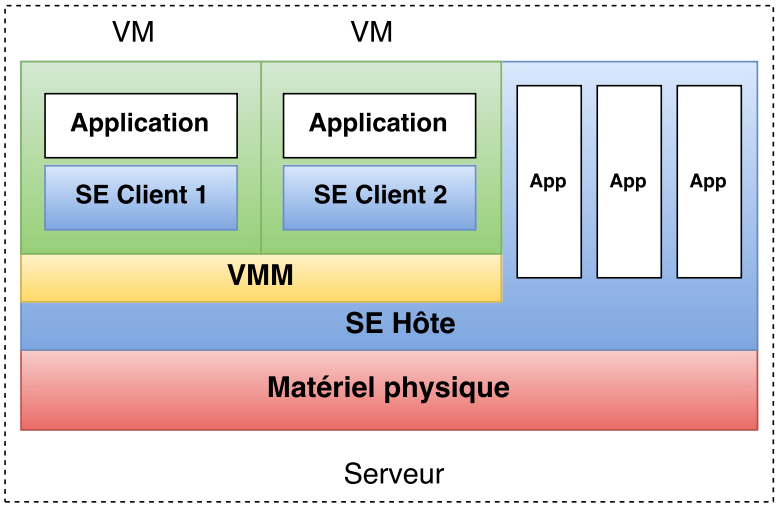


FIGURE 2.3 : Architecture hyperviseur de type 2

2.2.2 Les différents types d'hyperviseurs

Un hyperviseur est une couche logicielle sur laquelle s'exécute les machines virtuelles. Il permet la gestion des machines virtuelles, c'est à dire son cycle de vie. Le cycle de vie d'une machine virtuelle consiste en sa création et sa configuration (la configuration de ses ressources virtuelles, c'est à dire l'allocation d'une quantité de mémoire, d'un nombre de cœurs virtuels (VCPU), des périphériques virtuels et la capacité de stockage), son exploitation (utiliser la machine virtuelle et gérer son état) puis sa destruction. L'hyperviseur gère également l'accès aux ressources matériels des machines virtuelles. Dans [PG74], les auteurs Gerald J.Popek et Robert P.Goldberg classifient les hyperviseurs selon deux types.

2.2.2.1 Les Hyperviseurs de type 1

Les hyperviseurs de ce type sont des hyperviseurs dits "natifs" ou "bare metal" car ils reposent directement sur le matériel. En effet, ce sont des systèmes d'exploitation à part entière. La VMM est directement intégrée dans le noyau du système d'exploitation. Contrairement à des systèmes d'exploitation standards, les hyperviseurs ne sont utilisés que pour supporter les tâches liées à la virtualisation. Les hyperviseurs de type 1 les plus connus sont VMware ESX/ESXi [MW05], Microsoft Hyper-V [VV09], XEN [BDF+03] et KVM [KKL+07].

2.2.2.2 Les Hyperviseurs de type 2

Contrairement aux hyperviseurs de type 1, les hyperviseurs de type 2 ne reposent pas sur le matériel mais s'appuient sur un système d'exploitation existant. Ce sont les hyperviseurs les plus utilisés puisqu'ils sont destinés grand public. En effet, ces hyperviseurs sont simples d'utilisation et d'installation. Ils s'installent comment n'importe quelle autre application sur le système. Ainsi, la mise en place d'une machine virtuelle est rapide et permet de switcher rapidement entre le système hôte (celui qui supporte la virtualisation) et la machine cliente (la machine virtuelle). En revanche, ils sont moins performants que les hyperviseurs de type 1 dans la mesure où l'hyperviseur de type 2 s'exécute à côté des autres applications du système d'exploitation. De plus, ces hyperviseurs n'ont pas accès directement aux matériels, ils doivent communiquer avec le système d'exploitation sur lequel ils reposent. Les hyperviseurs de type 2 les plus répandus sont VMware Workstation/Player [vmwc], Microsoft Virtual PC [msv] et Oracle VirtualBox [ora].

2.2.3 La mise à disposition des ressources

L'administrateur alloue des ressources aux machines virtuelles, en fonction des besoins de ces dernières. De son côté, l'hyperviseur gère la distribution des ressources matérielles entre les différentes machines virtuelles. Ainsi, il met à disposition des ressources matérielles à chacune des machines virtuelles. Ces ressources sont gérées par le système d'exploitation de chaque machine virtuelle. Au sein de la machine virtuelle, il ne s'agit plus de ressources matérielles mais virtuelles. Pour expliquer cette différenciation, nous détaillons dans cette section les ressources matérielles qui sont allouées aux machines virtuelles et nous expliquons ce que représentent les ressources virtuelles.

2.2.3.1 Les ressources matérielles

- Processeur : aussi nommé CPU (Central Proccessing Unit), il s'agit du principal composant électronique d'un serveur. Il effectue les différents calculs des applications s'exécutant au sein du système d'exploitation. Les processeurs se distinguent par plusieurs caractéristiques :
  - le nombre de cœurs physiques : chaque cœur physique dispose de son propre ensemble de circuits et sa propre mémoire cache (cache de niveau 1, L1), lui permettant de lire et d'exécuter des instructions de manière autonome, c'est à dire séparément des autres cœurs physiques. Plus un processeur dispose de cœurs physiques, plus il est capable d'exécuter de tâches en parallèle.
  - le nombre de cœurs logiques : Un cœur physique peut être composé d'un ou deux cœurs logiques. Lorsqu'un cœur physique en comporte deux, cela signifie que le processeur supporte la technologie du multi-threading ou Hyper-Threading (HT). Dans le cadre de l'Hyper-Threading, implémentée par Intel [Mar02], chaque cœur physique dispose de deux cœurs logiques. Par exemple, un processeur de 2 cœurs physiques et disposant de la technologie HT a 2x2 soit 4 cœurs logiques. Tout comme les cœurs physiques, les cœurs logiques sont autonomes dans l'exécution d'une tâche.
  - la fréquence : généralement exprimée en MHz (Mega Hertz) voir GHz (Giga Hertz avec 1 GHz = 1000 MHz), elle exprime le nombre d'instructions effectuées par seconde. Par exemple, un processeur ayant une fréquence de 2,1 GHz est capable d'effectuer 2,1 milliards d'opérations par seconde.
- Mémoire : aussi nommée RAM (Random Access Memory), il s'agit de composants électroniques ayant la capacité de stocker des informations, sous forme de "bit" (0 ou 1). Elle stocke les données utiles aux programmes en cours d'exécution. Sa capacité s'exprime généralement en Mo (Méga Octet) ou Go (Giga Octet avec 1 Go = 1024 Mo). La mémoire vive dispose de deux caractéristiques :
  - temps d'accès : le stockage en mémoire vive permet au processeur d'accéder beaucoup plus rapidement aux informations souhaitées que via l'accès au disque dur. En effet, le temps d'accès à l'information, c'est à dire le temps entre la demande de l'information et son obtention, est de l'ordre de la dizaine de nanosecondes contre la centaine de millisecondes pour un disque dur. La mémoire vive est alors 1 000 000 fois plus rapide que le disque dur.
  - volatilité : la mémoire vive est volatile, c'est à dire qu'elle perd les informations qu'elle contient dès que le serveur est éteint. En effet, la mémoire vive étant composée de transistors et de condensateurs, la mémoire vive doit être constamment sous tension pour stocker les informations.
- Stockage : contrairement à la mémoire, le stockage sur disque dur est non volatile, ce qui permet de stocker les données sur la durée. Y sont stockées toutes données importantes comme le système d'exploitation et les applications des machines virtuelles. Ces données sont écrites sur des disques

magnétiques rotatifs, appelés plateaux, en code binaire (0 ou 1) grâce à un bras mécanique comportant des têtes de lecture et écriture. Un disque dur peut comporter plusieurs plateaux. Par un exemple, un disque dur de 2 To peut comporter deux plateaux de 1 To chacun. Il existe également des disques durs disposant non pas de plateaux magnétiques mais de la mémoire flash. Ces disques durs sont appelées SSD (Solid-State Drive). Il s'agit de la même technologie que les clés USB. Ces nouveaux composants sont moins fragiles car ils ne disposent pas de parties mécaniques et sont plus performants. Cependant ils sont beaucoup plus chers à l'achat ce qui explique qu'ils ne sont pas encore très répandus dans les centres de données. Le besoin de stockage étant de plus en plus important, cette ressource dispose de matériels spécifiques, appelés baies de stockage. Une baie de stockage est un équipement informatique, composé d'un ensemble de disques, disposés en tiroirs. Cet équipement est lié aux serveurs physiques via le réseau. Les disques durs se distinguent par plusieurs caractéristiques :

  - capacité : aujourd'hui, la capacité d'un disque dur est de l'ordre du To (Téra Octet avec 1 To=1024 Go). Plus le disque dur dispose d'une grande capacité, plus il stocke de données.
  - performance : la performance d'un disque dur dépend du temps d'accès à la donnée et de son débit. Ces paramètres varient en fonction de l'architecture du disque et de son interface de connexion. Ainsi, le temps d'accès est dépendant du nombre de plateaux, de la vitesse de rotation de ces plateaux et du temps de déplacement du bras magnétique. Le temps d'accès moyen à une donnée est autour de la centaine de millisecondes. Le débit est de plus dépendant de l'interface. L'interface est le connecteur qui permet de relier le disque dur à la carte mère via une nappe.

- Réseau : il permet la communication entre les serveurs et/ou entre les machines virtuelles. Généralement, un serveur physique dispose de plusieurs cartes réseaux afin d'éviter les goulots d'étranglements liés aux besoins des machines virtuelles. Le réseau permet également aux machines virtuelles d'accéder à leurs disques virtuels.

2.2.3.2 Les ressources virtuelles

- Processeur : l'allocation de ressources processeur à une machine virtuelle consiste en l'attribution d'un nombre de coeur virtuel. Cette ressource virtuelle est nommé vCPU (virtual Central Processing Unit). Côté ressource matérielle, son équivalent correspond au cœur logique. En effet, prenons l'exemple d'une machine virtuelle à laquelle 2 vCPU ont été attribués. Côté machine virtuelle, ces 2 cœurs virtuels sont vus comme 2 cœurs logiques. Le système d'exploitation de la machine virtuelle dispose d'un processeur à 2 cœurs. Côté hyperviseur, cette attribution de 2 cœurs virtuels à une machine correspond à la mise à disposition de 2 cœurs logiques. Ces 2 cœurs logiques ne sont pas exclusivement dédiés à cette machine. Les machines virtuelles partagent les ressources matérielles. Le nombre de vCPU maximum qui peuvent être alloués à une machine virtuelle correspond au nombre de cœurs logiques de l'hyperviseur. Il est possible d'attribuer à l'ensemble des machines virtuelles hébergées sur un hyperviseur un total de vCPU supérieur au nombre de cœurs logiques possédés par l'hyperviseur. Ce procédé est appelé "sur-provisionnement" (en anglais, "over-commitment"). Par exemple, un serveur physique disposant d'un processeur à 2 cœurs logiques peut héberger 2 machines virtuelles ayant chacune une attribution de 2 vCPU. Il y a alors un total de 4 vCPU pour 2 cœurs logiques. Pour gérer cela, l'hyperviseur ordonnance en fonction de la disponibilité et la demande. Dans notre exemple précédent, si une des 2 machines virtuelles a une forte activité processeur, l'hyperviseur mettra à disposition de cette machine virtuelle les 2 cœurs logiques.

- Mémoire : l'allocation de ressources mémoire à une machine virtuelle consiste en l'attribution d'une quantité de mémoire. Cette quantité de mémoire s'exprime généralement en Mega octets (Mo) ou

Giga octets (Go). De la même manière que pour la ressource processeur, il est possible d'allouer aux machines virtuelles plus de mémoire que ne possède l'hyperviseur. Il s'agit de la sur-allocation de mémoire. Cette sur-allocation est possible sans dégradation de performances. En effet, il est possible d'avoir 2 machines virtuelles disposant chacune de 4 Go de mémoire s'exécutant sur un serveur ne contenant que 4 Go de mémoire physique. Généralement, le système d'exploitation client (celui qui s'exécute au sein d'une machine virtuelle) consomme toute la mémoire mise à sa disposition. Plus il a de mémoire disponible, plus il en utilise. Cette utilisation n'est pas faite de manière efficace. En effet, le système d'exploitation ne libère pas forcément les pages mémoires inutilisées et réserve les pages mémoire disponibles. Pour éviter ce "gaspillage", il existe plusieurs techniques sophistiquées de gestion de la mémoire par les hyperviseurs. Ces procédés sont présentés dans l'ordre de performances. Plus de détails sont disponibles dans [Wal02] :

- TPS (Transparent Page Sharing pour partage de pages mémoire) [BDGR97] : Lorsque plusieurs machines virtuelles sont en cours d'exécution sur le même serveur, il est possible qu'elles disposent de pages mémoires identiques. C'est notamment le cas si plusieurs d'entres-elles exécutent une même version d'un système d'exploitation ou les mêmes applications. Des pages mémoires peuvent donc être partagées entre les machines virtuelles. Le principe du partage mémoire consiste à supprimer les pages mémoires identiques et n'en garder qu'une seule copie dans la mémoire physique de l'hôte, qui sera par la suite partagée par les machines virtuelles. Ce procédé permet ainsi de réduire la consommation mémoire physique hôte par les machines virtuelles.
- Ballonning [Wal02, SFS06] : Lorsque plusieurs machines virtuelles sont en cours d'exécution, la quantité de mémoire physique hôte disponible devient faible. En raison de l'isolement des machines virtuelles, le système d'exploitation client n'a pas conscience qu'il est exécuté à côté d'autres machines. Par conséquent, il n'a pas connaissance de l'état des autres machines virtuelles ni de l'état du serveur. Il ne peut donc pas libérer de mémoire physique hôte. Le ballonning est une technique permettant au système d'exploitation client de libérer de la mémoire à la demande de l'hyperviseur. La mémoire libérée peut ensuite être utilisée par l'hyperviseur pour une autre machine virtuelle. Pour cela, l'hyperviseur "gonfle un ballon" au sein de la machine virtuelle. En gonflant ce ballon, l'hyperviseur force le système d'exploitation invité à libérer les pages mémoires inutilisées. L'hyperviseur détecte ainsi les pages mémoires qui peuvent être libérées au niveau de la mémoire physique hôte.
- Memory Compression [TG05] : Lorsque les techniques précédentes ne répondent plus aux besoins, l'hyperviseur va mettre en application une technique qui vise à compresser la mémoire. Lorsque des pages virtuelles ont besoin d'être swappées, l'hyperviseur va d'abord tenter de compresser ces dernières. Pour ce faire il calcule le ratio de compression. Si le gain est supérieur à 50%, la page virtuelle est compressée. Dans le cas contraire, elle est swappée. Cette technique reste plus performante que le swapping dans la mesure où l'accès à la mémoire est plus rapide que l'accès au disque.
- Swapping : Lorsque la mémoire physique est saturée, c'est à dire que l'hyperviseur n'est plus en mesure de fournir de pages mémoire à une machine virtuelle, ce dernier utilise le disque dur comme mémoire. Un fichier de swap est alors créé. Cette technique implique de fortes dégradations de performances dans la mesure où les temps d'accès aux informations sont importants.

• Stockage : Une machine virtuelle dispose de disques durs virtuels sous forme de fichiers. L'allocation de ressource de ce type à une machine virtuelle consiste en l'attribution d'une quantité d'espace maximale qui sera utilisée par les fichiers de la machine virtuelle. Ces fichiers sont stockés physiquement soit sur les disques durs locaux des hyperviseurs, soit sur des périphériques dédiés aux stockages (baies de stockage), accessibles depuis le réseau. Avec la forte croissance des besoins en stockage due aux système d'exploitation et applications de plus en plus volumineuses, accouplée à la faci-

lité de mise en place de machines virtuelles, cette ressource est parfois limitante dans les centres de données. Elle requiert à elle-seule une administration et une surveillance particulière. En effet, pour garantir aux machines virtuelles des temps d'accès rapides à leurs données, il faut analyser finement l'activité des périphériques de stockage sinon des goulots d'étranglements peuvent subvenir. Ces derniers auront pour conséquence de dégrader les performances des machines virtuelles accédant à la baie de stockage. En fonction des hyperviseurs, l'espace disque utilisée par les fichiers de stockage des machines virtuelles est configurable :

- Thin : il s'agit littéralement de "provisionnement fin". Le fichier de stockage de la machine virtuelle va augmenter sa taille sur le disque dur en fonction des besoins réels de la machine virtuelle. Par exemple, une machine virtuelle dont l'allocation disque est de 40 Go, utilise seulement 2 Go. Alors, l'image du disque dur de la machine virtuelle aura une taille non pas de 40 Go mais uniquement 2 Go. Dans cette configuration, l'hyperviseur ne créé pas directement un fichier d'une taille de 40 Go mais augmente la taille de ce dernier dynamiquement en fonction des besoins de la machine virtuelle. De cette façon, de la même façon que pour la mémoire, il est possible de faire du sur-provisionnent. Ainsi, sur un disque dur physique de 100 Go, il est possible de créer 4 machines virtuelles ayant une allocation disque de 50 Go chacune.
- Thick : il s'agit littéralement de "provisionnement épais". A l'inverse du thin, la taille du fichier du disque dur de la machine virtuelle correspond à l'allocation de départ. De ce fait, en reprenant l'exemple précédent, bien que la machine virtuelle n'utilise que 2 Go, la taille du fichier sera de 40 Go.

Il existe de nombreux fichiers qui sont liés à la machine virtuelle. En plus de son image disque, une machine virtuelle génère d'autres fichiers tels que des fichiers de logs, un fichier de configuration, des fichiers de snapshots ou encore un fichier de swap. Il existe différents formats d'images disque :

- Raw : il s'agit du format "brut", c'est à dire une image binaire simple, sans optimisations de stockage. C'est une image de type "thick".
- Qcow ("QEMU Copy On Write") : il s'agit d'un format de fichiers utilisé pour les images disques de machines virtuelles sous QEMU. Ce format utilise une stratégie d'optimisation de stockage de type "thin", c'est à dire que l'image disque a une taille qui correspond à l'espace disque utilisée vue par la machine virtuelle. Qcow2 est une amélioration de qcow dont la principale différence est la prise en charge d'images disques multiples (snapshots). Ainsi, la lecture peut se faire dans une image et l'écriture dans une autre ce qui permet de garder un état de la machine virtuelle et y revenir si besoin.

- Réseau : Comme un serveur, une machine virtuelle peut disposer de plusieurs périphériques virtuels. Le réseau permet la communication de la machine virtuelle avec l'hyperviseur mais également accéder à d'autres machines virtuelles, périphériques de stockage (baie de stockage afin d'accéder à son disque) ou internet. La virtualisation permet de virtualiser les sous-réseaux, alors nommé VLAN pour Virtual Local Area Network. Un sous-réseau n'est ni plus ni moins qu'une division d'un réseau informatique. La machine virtuelle se connecte à un ou plusieurs VLAN via ses interfaces réseaux virtualisées. Pour ces dernières, plusieurs configurations sont disponibles :

  - Bridge : La machine virtuelle est connectée au réseau physique, sur le même réseau que le serveur sur lequel elle est hébergée. Elle dispose de sa propre adresse IP et peut communiquer et être joignable par n'importe quelle machine du réseau.
  - NAT (Network Address Translation) : La machine virtuelle utilise la même adresse IP que l'hyperviseur pour accéder au réseau. Cependant, contrairement au mode Bridge, la machine virtuelle ne peut pas être joignable.

  - Host-Only : La machine virtuelle est sur un réseau privé. Elle ne peut communiquer qu'avec le serveur sur lequel elle est hébergée.

2.2.4 Le cycle de vie des machines virtuelles

Le cycle de vie d'une machine virtuelle se décompose en plusieurs étapes :

- Création : la création d'une machine virtuelle peut se faire de deux façons. Soit la machine virtuelle est créée de zéro, soit elle est importée ou clonée. Dans le premier cas, cela consiste à créer une machine virtuelle "nue", c'est-à-dire définir ses caractéristiques (donner un nom, définir le serveur physique sur lequel elle est hébergée) et configurer ses ressources virtuelles (définir un nombre de VCPU, une quantité de mémoire virtuelle, un ou plusieurs disques durs, un ou plusieurs périphériques de stockage). La machine virtuelle "nue" ne dispose d'aucun système d'exploitation, elle n'est pas "prête à l'emploi". Dans le second cas, il s'agit d'importer ou cloner une machine virtuelle déjà existante. Ce procédé permet de déployer facilement et rapidement une nouvelle machine virtuelle. Facilement dans la mesure où sa mise en place ne nécessite que quelques actions par l'administrateur. Rapidement car le déploiement consiste uniquement en la copie de la machine virtuelle importée ou clonée. Le temps de déploiement est donc plus rapide que le temps passé à la création d'une machine virtuelle de zéro (création, installation du système d'exploitation, environnement logiciel, etc...). De plus, une fois son déploiement effectuée, la machine virtuelle est "prête à l'emploi".

- Exploitation : l'exploitation d'une machine virtuelle consiste en son utilisation. Au cours de cette période, tout comme pour un serveur physique, une machine virtuelle peut être démarrée, redémarrée ou arrêtée. La virtualisation apporte de plus d'autres possibilités :

  - suspendre : il s'agit de "mettre en pause" l'activité de la machine virtuelle. Cette dernière cesse son exécution, libérant ainsi les ressources processeur, mémoire (la mémoire utilisée par la machine virtuelle est alors stockée sur disque dur) et réseau.
  - sauvegarder : il s'agit de copier l'état actuel de la machine virtuelle. La copie créée est appelée *snapshot*. Ces sauvegardes permettent de redémarrer la machine virtuelle à un état antérieur en cas de besoin (suite à une application vérolée par exemple).
  - migrer : il s'agit de déplacer la machine virtuelle d'un serveur à un autre. La migration est souvent liée aux besoins en ressources. Il existe deux types de migrations. La migration dite à "froid", qui consiste à déplacer une machine virtuelle suspendue et à "chaud" [CFH+05], qui consiste à déplacer une machine virtuelle en cours d'exécution.

- Destruction : il s'agit du processus de dé-commissionnement de la machine virtuelle. Une fois que cette dernière n'est plus utile, la machine virtuelle est soit désinscrite, soit supprimée. La désinscription consiste en la suppression de la liste des machines virtuelle à administrer. L'ensemble des fichiers liés à la machine virtuelle sont sauvegardés. La suppression consiste en la suppression de tous les fichiers liés à cette machine virtuelle.

2.3 Conclusion

Ce chapitre a introduit le Cloud Computing et la Virtualisation qui est une technologie largement utilisée dans les centres de données. Avec l'explosion de l'usage d'Internet, une explosion liée à l'externalisation des services, les infrastructures des centres de données ne cessent de croître. La virtualisation permet de mutualiser les ressources et ainsi diminuer le nombre de serveurs physiques. Elle permet également d'apporter de la flexibilité dans la gestion des capacités de l'infrastructure, via le déploiement et la gestion rapide des machines virtuelles. Cependant, son élasticité et sa souplesse ont conduit à l'explosion des environnements

virtuels à gérer. Il est commun pour un administrateur système de gérer plusieurs centaines de machines virtuelles. Sans outil approprié, les différentes tâches d'administration sont délicates à réaliser. Dans la suite de cette thèse, nous abordons les différentes contributions répondant à notre problématique "Comment réaliser un système d'analyses fines pour la gestion des machines virtuelles". Pour cela, nous avons divisé nos travaux en deux chapitres. Dans un premier chapitre, nous discutons de la supervision des machines virtuelles, allant de l'étude des compteurs de performances à monitorer, à l'introspection sous Linux. Le second chapitre présente nos travaux autour de l'analyse fine des ressources systèmes monitorées.

II

Contributions

3

Supervision des ressources systèmes dans les centres de données

Dans ce chapitre, nous discutons des métriques de performances analysées, métriques fondamentales dans la réalisation d'un outil d'analyses fines des performances des machines virtuelles. Suite à cette discussion, nous abordons nos travaux autour de l'introspection non intrusive sous Linux, procédé permettant de superviser les comportements des machines virtuelles.

Sommaire

En vue de garantir les contrats de services négociés avec leurs clients, mesurer et analyser l'activité des machines virtuelles et des serveurs est une préoccupation majeure des fournisseurs de Cloud. L'analyse fine de l'activité des machines virtuelles nécessite une étude approfondie des métriques de performances. Pour une même ressource, le nombre de compteurs est important. Il faut alors bien comprendre la sémantique de chacun des compteurs dans le but de bien choisir ceux représentant l'activité réelle de la machine virtuelle. Déployées au niveau de l'hyperviseur, les sondes systèmes non intrusives permettent de suivre l'activité des serveurs et des machines virtuelles. Ces sondes collectent ainsi les métriques précédemment étudiées, principalement pour les ressources processeur et mémoire. La non intrusivité de ces sondes permet également un déploiement sans contraintes chez les fournisseurs de Cloud dans le mesure où elles ne nécessitent aucun agent dans les machines virtuelles, respectant ainsi la sécurité et le bon fonctionnement des machines virtuelles des clients de solutions Cloud.

Dans ce chapitre, nous abordons dans une première section, nos différentes études sur les compteurs de performances. Ces dernières ont été réalisées à partir de l'hyperviseur le plus usité au sein des entreprises, l'hyperviseur VMware [vmwb]. Dans une seconde section, nous présentons les travaux effectués autour de l'introspection non intrusive de machines virtuelles, technique permettant d'inspecter les machines virtuelles depuis l'extérieur en vue d'étudier leur comportement.

3.1 Études des compteurs de performances

3.1.1 Les compteurs processeur

Une machine virtuelle ressemble en tout point à une machine physique dans la mesure où elle dispose également d'un ensemble de ressources matérielles mais virtualisées. Comme une machine physique, la machine virtuelle possède un nombre de processeurs virtuels appelés vCPU. De la même manière qu'un processeur ordonnance des processus, l'hyperviseur, qui met à disposition des machines virtuelles les ressources matérielles, ordonnance les accès aux différents cœurs logiques (autrement appelés "Threads") du processeur. De cette façon, chaque vCPU dispose d'un état. Lors du démarrage d'une machine virtuelle, les vCPU associés à cette machine virtuelle sont créés. Chacun de ces vCPU est à l'état "RUN" ou à l'état "READY", en fonction de la disponibilité de la ressource matérielle. Un vCPU à l'état "RUN" dispose de l'accès au processeur. Un vCPU à l'état "READY" est en attente d'accès au processeur, ce dernier n'étant pas disponible, probablement dû à une surcharge du processeur. Il rentre dans l'état "RUN" une fois que l'ordonnanceur lui donne accès au processeur. Il peut ensuite être dés-ordonnancé et rentre dans l'état "READY" ou "COSTOP" ou être mis en attente si ce dernier est bloqué par l'accès à une ressource (disque ou réseau). Il sera plus tard réveillé une fois que la ressource bloquante est disponible. L'état "COSTOP" n'existe que pour des machines virtuelles ayant plusieurs vCPU. En effet, le "COSTOP" est un état dans lequel un vCPU est en attente d'autres vCPU afin d'être ordonnancés simultanément. La figure 3.1 illustre les transitions d'états. Les sous-sections suivantes présentent les différents états.

3.1.1.1 Run

Cette métrique représente le temps, exprimé en millisecondes, passé par un ou plusieurs vCPU de la machine virtuelle dans l'état "RUN". Cet état correspond à l'état dans lequel les vCPU ont accès au processeur afin d'exécuter leurs tâches. Lorsque la valeur du "RUN" est proche de (100 x le nombre de vCPU) % au sein de la machine virtuelle, cela signifie que la machine virtuelle est en pleine charge (en anglais, busy). A l'inverse, si cette valeur est proche de zéro, cela ne signifie pas que la machine virtuelle ne fait rien (en anglais, idle). Il faut alors vérifier la valeur des compteurs "READY", "WAIT" et "COSTOP" pour déduire dans quel état se trouve la machine virtuelle.

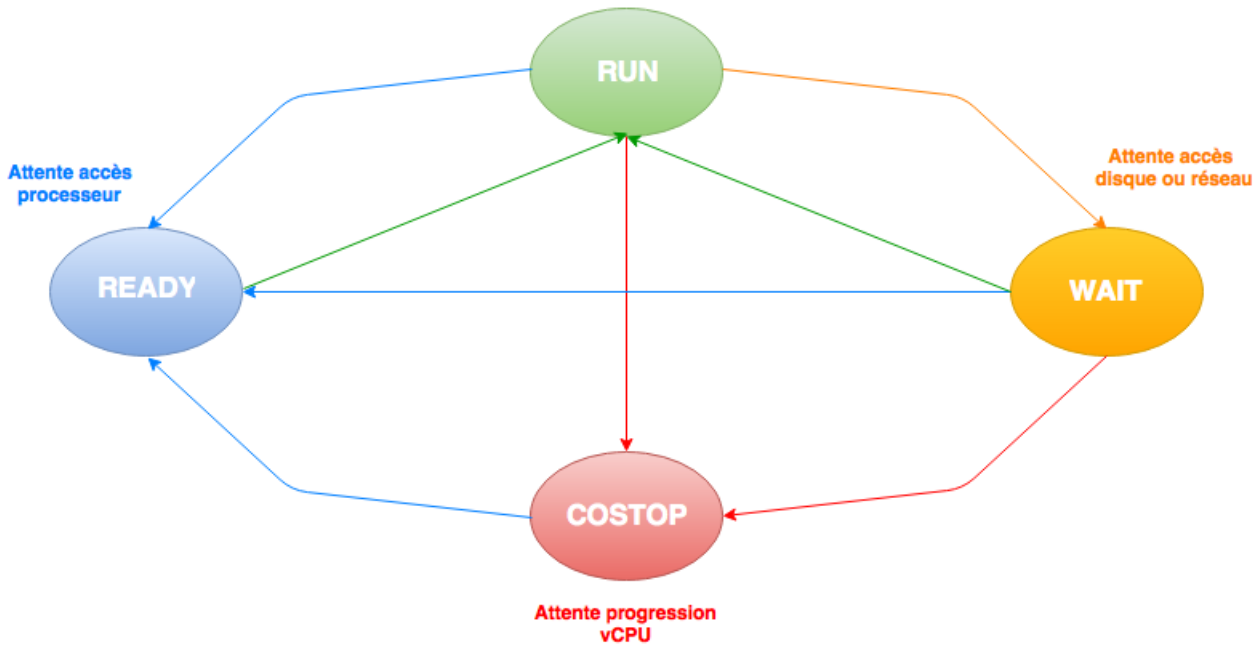


FIGURE 3.1 : Transition des états [VMw13]

3.1.1.2 Ready

Cette métrique représente le temps, exprimé en millisecondes, qu'un ou plusieurs vCPU ont attendu avant d'avoir accès à un ou plusieurs cœurs logiques par l'hyperviseur afin de continuer leur exécution. Il représente donc la somme des temps d'attente de tous les vCPU d'une machine virtuelle. Ce compteur est crucial dans la mesure où il permet de suivre les performances de la machine virtuelle. Plus le "READY" d'une machine virtuelle est élevé, plus ses performances sont dégradées, impliquant un mauvais ressenti pour l'utilisateur de la dite machine. La valeur du "READY" pour une machine virtuelle correspond à la somme des tous les "READY" des vCPU de cette machine. Par exemple, la valeur maximale possible pour une machine virtuelle avec un vCPU est de 100%. Elle sera de 200% pour une machine virtuelle ayant 2 vCPU. Dans la pratique, afin de garantir de bonnes performances, le "READY" doit être inférieur à 5% par vCPU. Il peut éventuellement atteindre 10%. Au delà de ce seuil, la dégradation de performance est ressentie par l'utilisateur.

Les facteurs responsables d'une augmentation de la valeur du "READY" sont les suivants :

- Limite (en anglais, CPU Limit) : Cette dernière permet de limiter le temps d'exécution d'une machine virtuelle sur l'hyperviseur. L'avantage étant d'éviter qu'une machine virtuelle monopolise la ressource processeur, risquant de pénaliser les autres machines virtuelles qui s'exécutent sur ce même serveur. Cependant, imposer une limite CPU sur la machine virtuelle n'est en réalité pas une bonne pratique. En effet, cette limite doit être utilisée avec parcimonie dans la mesure où l'activité d'une machine virtuelle est difficilement prévisible. Ainsi, la machine virtuelle a plus de risques d'atteindre cette limite, entraînant alors une augmentation du "READY". Pour vérifier que la limite CPU est responsable de l'augmentation du "READY", il faut alors vérifier le compteur "MLMTD". La métrique "MLMTD" correspond à la somme des temps d'attente des vCPU, exprimée en millisecondes, dû à la contention. La valeur de cette dernière est incluse dans celle du "READY". En soustrant la valeur de "MLMTD" au "READY", on peut en déduire si l'augmentation du "READY" est causée par cette limite.

- Sur-allocation (en anglais, CPU Over-subscription) : L'allocation d'un trop grand nombre de vCPU sur le serveur peut également être responsable d'une dégradation des performances. En effet, plus il y a de vCPU à vouloir accéder au processeur, plus le temps d'accès à ce dernier sera long. L'hyperviseur, via l'ordonnanceur présent dans le VMKernel, passera beaucoup de temps à ordonnancer

les différentes requêtes des machines virtuelles. Pour éviter ces dégradations de performances, la pratique recommande un ratio de consolidation (nombre de vCPU par cœur logique) compris entre 1 :3 et 1 :5. C'est à dire au maximum 5 vCPU par cœur logique [Low13]. Par exemple, un serveur physique disposant d'un processeur à 8 cœurs logiques peut héberger des machines virtuelles dont la somme de tous leurs vCPUs ne dépassent pas 40 vCPU. Une consolidation supérieure à 5 risque d'entraîner une dégradation des performances.

- Affinité (en anglais, CPU Affinity) : il s'agit d'imposer à l'hyperviseur une association vCPU / cœur logique. Cette dernière permet de placer les vCPU d'une machine virtuelle sur les cœurs logiques d'un même processeur physique. L'objectif de l'affinité étant d'optimiser l'accès à la mémoire. En effet, en plaçant les différents vCPU sur le même processeur physique, ces derniers partagent alors les différents niveaux de cache d'un même processeur. La mise en œuvre de l'affinité implique un ordonnancement non optimal des processus par l'hyperviseur. En effet, bien que des cœurs logiques soient disponibles, si ces derniers ne correspondent pas à l'affinité, l'hyperviseur de donnera pas cette ressource à la machine virtuelle. Elle a donc plus de risque d'attendre la disponibilité des ressources souhaitées. De plus, il est possible que plusieurs machines virtuelles disposent d'une affinité, basée sur les mêmes cœurs logiques. Les conséquences de l'affinité sont identiques au cas précédent. Cependant, imposer une affinité CPU à des vCPU a un sens pour les machines virtuelles dont les vCPU accèdent souvent à la mémoire. En effet, comme discuté précédemment, mapper ses cœurs virtuels sur des cœurs logiques appartenant à un même processeur permet d'optimiser l'accès à la mémoire.

3.1.1.3 Wait

Cette métrique représente le temps, exprimé en millisecondes, passé par le ou les vCPU de la machine virtuelle à attendre l'accès à une ressource (disque ou réseau). Une forte valeur de cette métrique peut être le résultat d'un périphérique de stockage peu performant, sur lequel réside les fichiers de la machine virtuelle. Le principal facteur responsable d'un fort "WAIT" est la latence. Ce dernier est le temps passé à accéder à une information. Ainsi, elle est présente qu'elle que soit le périphérique interrogé (périphérique de stockage, USB, point de montage ISO). En plus de l'attente d'accès, le "WAIT" comporte également la valeur de la métrique "IDLE". Cette dernière correspond au temps passé par un ou plusieurs vCPU de la machine virtuelle à ne rien faire.

Pour distinguer si une machine virtuelle disposant d'un fort "WAIT" est en attente ou ne fait rien ("IDLE"), il faut soustraire la valeur du compteur "IDLE" au compteur "WAIT".

3.1.1.4 Costop

Cette métrique représente le temps, exprimé en millisecondes, pendant lequel un ou plusieurs vCPU d'une même machine virtuelle sont "mis en pause" par l'ordonnanceur de l'hyperviseur. Avec le sur-provisionnement, l'hyperviseur doit gérer un nombre de vCPU alloués aux machines virtuelles, supérieur au nombre de cœurs logiques dont il dispose. Un système d'exploitation traditionnel requiert que les cœurs du processeur travaillent de manière synchrone, sinon le fonctionnement du système d'exploitation invité risque d'être instable, voir "planter". Ainsi, cette synchronisation de progression des vCPU est gérée par l'hyperviseur, par des algorithmes d'ordonnancement.

Les travaux effectués dans [VMw13] permettent de mieux comprendre le fonctionnement des différents algorithmes d'ordonnancements mis en place au sein de l'hyperviseur. Ainsi, dans ce papier sont expliqués les deux algorithmes d'ordonnancement :

- Strict Co-Scheduling (ESX 2.x) : dans cet algorithme, l'ordonnanceur évalue la non-progression de chaque vCPU de la machine virtuelle. Un vCPU ne progresse pas s'il n'est pas dans l'état "RUN" ou "IDLE". Si la non-progression d'un vCPU dépasse un certain seuil, de l'ordre de quelques millisecondes, la totalité des vCPU de la machine virtuelle sont dés-ordonnancés et rentrent dans l'état

"COSTOP" et ne seront ordonnancés que lorsqu'il y aura assez de cœurs logiques disponibles pour ordonnancer l'ensemble des vCPU simultanément (on parle alors de "co-start"). Pour illustrer les faits, prenons un exemple dans lequel nous disposons d'un processeur à 2 cœurs logiques, d'une machine virtuelle avec 1 vCPU "VM1" et d'une machine virtuelle avec 2 vCPU "VM2". Lorsque l'ordonnanceur donne l'accès au processeur à "VM1", l'autre cœur logique disponible sur le serveur ne peut pas être utilisé pour ordonnancer un des 2 vCPU de "VM2". Cet algorithme dit "strict", peut donc provoquer une "fragmentation" CPU se traduisant pas des retards d'ordonnancement et une faible utilisation CPU. Cependant, il est à noter qu'un vCPU dans l'état "IDLE" est considéré comme un vCPU faisant des progrès dans la mesure où il n'a rien à exécuter. De ce fait, en reprenant l'exemple précédent, le deuxième cœur logique du serveur physique peut être utilisé par "VM2" pour ordonnancer un vCPU si et seulement si un des 2vCPU de "VM2" est "IDLE". La "VM2" peut ainsi continuer son exécution.

- Relaxed Co-Scheduling (ESX 3.x et supérieur) : Cet algorithme remplace le précédent et a été modifié dans le but d'améliorer l'utilisation CPU et de supporter les machines virtuelles disposant de plus de 4 vCPU. Alors que dans l'algorithme précédent, l'existence d'un vCPU ne progressant pas a pour conséquence la mise en pause de tous les vCPU de la machine virtuelle, dans cet algorithme, le vCPU ayant la plus forte progression décide lui-même s'il doit se mettre en "COSTOP". Si le temps de non-progression du vCPU ayant la plus faible progression dépasse un certain seuil, le vCPU "leader" (celui ayant la plus forte progression) se met dans l'état "COSTOP". De part ce changement, cet algorithme réduit considérablement le risque de fragmentation CPU et améliore nettement l'utilisation globale du processeur dans la mesure où il ne requiert pas d'ordonnancer un groupe de vCPU simultanément.

Bien entendu, cette métrique n'a de sens que pour la supervision de machines virtuelles disposant de plus de 1 vCPU.

3.1.2 Les compteurs mémoire

3.1.2.1 Terminologie

La quantité de mémoire disponible sur l'hyperviseur est généralement responsable du nombre de machines que ce dernier peut héberger. En effet, la mémoire est le point de contention généralement constaté dans les infrastructures virtualisées. Ainsi, une attention particulière doit être portée sur l'analyse fine de cette dernière. De plus, plusieurs mécanismes permettent à un hyperviseur d'allouer aux machines virtuelles plus de mémoire que ce qu'il possède (la somme totale des mémoires allouées est supérieure à la capacité du serveur), autorisant alors le sur-provisionnement. Avant de rentrer dans le détail de chacun de ces mécanismes, il est nécessaire de commencer cette section par l'explication de différents termes qui seront employés dans les sous-sections suivantes.

- La mémoire virtuelle cliente (en anglais, guest virtual memory) : espace d'adressage virtuelle représenté par le système d'exploitation pour les applications s'exécutant au sein de ce dernier. Il s'agit alors de la mémoire visible par les applications.

- La mémoire physique cliente (en anglais, guest physical memory) : mémoire visible par le système d'exploitation de la machine virtuelle,

- La mémoire physique machine (en anglais, machine memory) : mémoire visible par l'hyperviseur.

- Le concept du sur-provisionnement mémoire (en anglais, memory overcommitment) : La mémoire physique machine est sur-provisionnée lorsque la somme des quantités de mémoire physique cliente des machines virtuelles s'exécutant sur le serveur est supérieure à la quantité de mémoire physique machine. Le sur-provisionnement mémoire apporte deux bénéfices non négligeables :

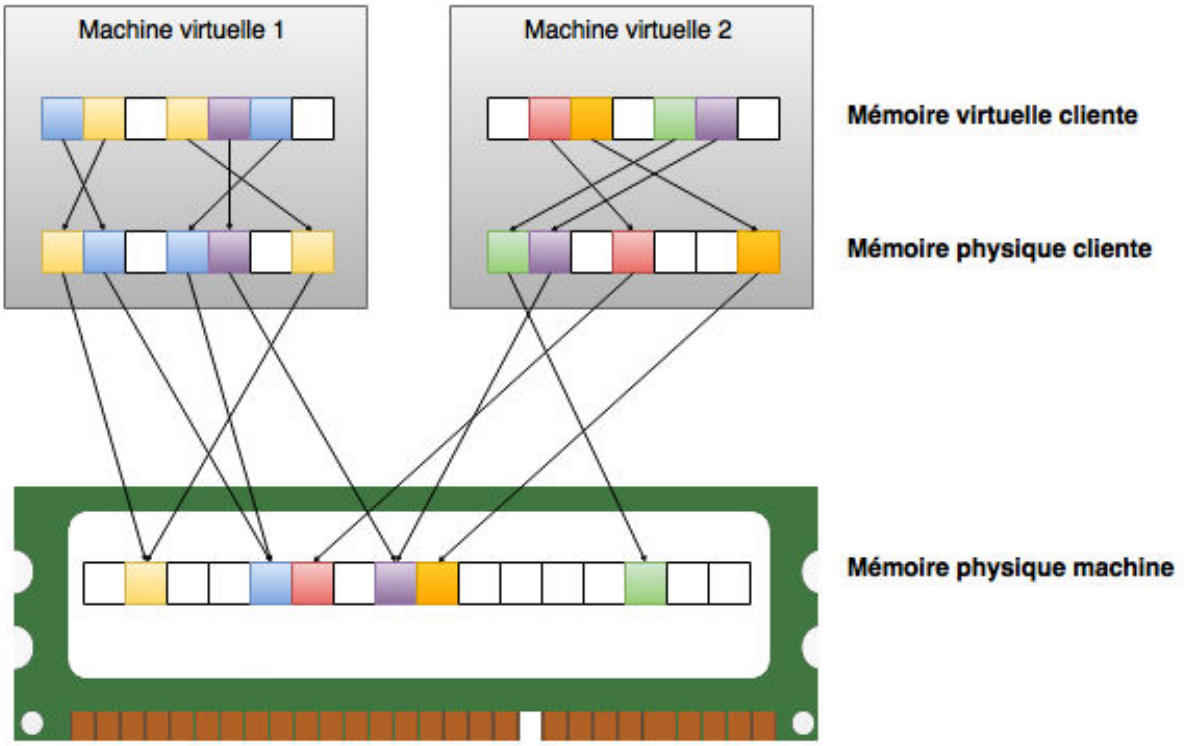


FIGURE 3.2 : Exemple d'utilisation de la mémoire [VMw11]

- meilleure utilisation : en général, dans les centres de données, certaines machines virtuelles peuvent nécessiter une plus faible charge mémoire que d'autres. Leur mémoire est donc rarement utilisée et est alors gaspillée. L'hyperviseur utilise des techniques de réclamation mémoire afin de récupérer les pages mémoires inutilisées ou inactives des machines virtuelles pour les fournir à d'autres machines virtuelles qui en ont activement besoin.
- meilleure consolidation : grâce aux techniques de sur-provisionnement, chaque machine virtuelle s'exécutant sur le serveur physique ne laisse qu'une faible empreinte mémoire sur la mémoire physique machine. ce qui permet alors d'héberger un plus grand nombre de machines virtuelles tout en respectant les performances.

3.1.2.2 Partage de pages mémoires

Le partage de pages mémoires [BDGR97] est la première technique employée par l'hyperviseur en cas de sur-provisionnement mémoire. Au sein d'un centre de données, il est commun qu'un grand nombre de machines virtuelles exécutent la même version d'un système d'exploitation. De ce fait, ces machines virtuelles disposent des contenus de pages mémoires identiques. Cette observation offre alors la possibilité de partager entre les machines virtuelles, les pages mémoires aux contenus identiques. Ainsi, la technique du partage de pages mémoires permet à l'hyperviseur de consolider les pages mémoires en ne gardant qu'une seule copie du contenu au sein de la mémoire machine physique. Ces pages mémoires sont ensuite partagées entre toutes les machines virtuelles. Ce procédé permet donc d'optimiser l'utilisation de la mémoire physique machine.
Au sein de l'hyperviseur VMware, les pages mémoires dites "redondantes" sont identifiées par leur contenu. Cela signifie que les pages aux contenus identiques peuvent être partagées indépendamment de quand, où, et comment elles sont générées. L'hyperviseur analyse le contenu de chacune des mémoires physiques clientes des machines virtuelles afin d'en déterminer des opportunités de partage. Au lieu de comparer chaque byte des pages mémoire physique cliente, l'hyperviseur utilise le hachage pour identifier les pages potentiellement identiques. Si la valeur de hachage d'une page de la mémoire physique cliente correspond à une entrée existante, une comparaison complète du contenu de la page est effectuée pour exclure un faux positif. Une

fois le contenu de la page confirmée, le mapping entre la page de la mémoire physique cliente et la page de la mémoire physique machine est changé pour pointer vers le bloc au contenu identique. Le bloc "redondant" est ensuite libéré.
La figure 3.2, issue des travaux effectués dans [VMw11] permet d'illustrer la technique du partage de pages mémoire. Le mémoire physique cliente de la machine virtuelle 1 dispose de 5 blocs (2x'bleu', 1x'violet', 2x'jaune'). Celle de la machine virtuelle 2 dispose de 4 blocs (1x'rouge', 1x'orange', 1x'vert', 1x'violet'). Les blocs partagés sont ceux présents sur la mémoire physique machine et disposant de plusieurs flèches. Il s'agit des blocs 'bleu', 'violet' et 'jaune' (avec 'bleu' et 'jaune' partagés par la même machine virtuelle 1 et 'violet' partagé par les deux). Cet exemple permet également une meilleure compréhension de certains compteurs mémoire.

Ainsi, de cela nous mettons en avant plusieurs compteurs :

- **Memory Consumed** : Ce compteur permet de connaître la quantité de mémoire physique machine utilisée par la machine virtuelle. Autrement dit celle réellement consommée par la machine virtuelle sur le serveur. Pour calculer la valeur du compteur Memory Consumed de la machine virtuelle 1, il faut compter le nombre de blocs de la mémoire physique machine dont les flèches proviennent de la mémoire physique cliente. Il y a trois blocs ('bleu', 'violet', 'jaune'). Cependant, le bloc 'violet' est partagé avec la machine virtuelle 2. Il suffit alors de calculer la valeur des 2 blocs (2*4 ko) et d'ajouter la moitié du troisième, soit une Memory Consumed de 10 ko dans cet exemple.

- **Memory Granted** : Ce compteur permet de connaître la quantité de mémoire physique cliente consommée par la machine virtuelle. Pour calculer la valeur du compteur Memory Granted de la machine virtuelle 1, il faut compter le nombre de blocs de la mémoire physique cliente disposant de flèches vers la mémoire physique machine. Il y a donc 5 blocs (2x'bleu', 1x'violet', 2x'jaune'), soit une Memory Granted de 20 ko (5*4 Ko). La différence avec le compteur Memory Consumed montre que la mémoire peut être économisée grâce aux techniques de partage.

- **Memory Shared** : La mesure Memory Shared de l'hôte correspond à la somme des compteurs Memory Shared de chacune des machines virtuelles. Autrement dit il s'agit de la somme des blocs de la mémoire physique cliente partagée au niveau de la mémoire physique machine. Pour mesurer la Memory Shared de l'hôte, il faut sommer le nombre de blocs ayant des flèches allant de la mémoire physique cliente en direction de la mémoire physique machine, possédant eux-mêmes plusieurs flèches pointant dans leur direction pour chacune des machines virtuelles. Il y a un total de 6 blocs, 5 blocs (2x'bleu', 1x'violet', 2x'jaune') pour la machine virtuelle 1 et 1 bloc ('violet') pour la machine virtuelle 2, soit une Memory Shared pour l'hôte de 24 ko (6*4 ko).

- **Memory Shared Common** : Ce compteur permet de connaître la quantité de mémoire physique machine partagée par les machines virtuelles. Pour quantifier la valeur Memory Shared Common de l'hôte, il faut compter le nombre de blocs de la mémoire physique machine ayant plusieurs flèches en leur direction. Il y a 3 blocs ('bleu', 'violet', 'jaune'), soit un total de 12 ko.

A travers le précédent exemple, nous avons expliqué en quoi consistait chacun des compteurs mémoire permettant de déterminer si la technique de Transparent Page Sharing est mis en place ou non sur l'hyperviseur. Ce mécanisme optimise l'utilisation de la mémoire physique machine, permettant ainsi de sur-allouer de la mémoire. Cependant, dans certains cas de contention, il est nécessaire de libérer de la mémoire physique. Les techniques présentées dans la suite de cette section permettent d'optimiser l'utilisation ou libérer de la mémoire physique.

3.1.2.3 NUMA

NUMA (de l'anglais Non-Uniform Memory Access) est une architecture matérielle multiprocesseurs. Cette dernière permet de relier par des bus spécifiques chaque processeur disposant de leur propre mémoire.

Ainsi, dans ce système, le processeur accède plus rapidement à la mémoire que dans le système SMP (Symmetric Multiprocessing), autrement appelé UMA. En effet, dans le cas de SMP, tous les processeurs partagent la même mémoire. Chaque processeur composé de sa mémoire forment un nœud NUMA. Chacun de ces nœuds peut être utilisé simultanément. Au sein de ce système, l'hyperviseur utilise les optimisations apportées par l'architecture NUMA afin d'ordonnancer les vCPU d'une même machine virtuelle sur le même noeud NUMA. Pour illustrer ce fait, prenons un exemple composé d'un hyperviseur bi-processeur à 4 coeurs et d'une machine virtuelle configurée à 2 vCPU. Alors que dans une architecture UMA les vCPU sont affectés à n'importe quel cœur des 2 processeurs du serveur (vCPU1 sur le processeur A et vCPU2 sur le processeur B), au sein d'une architecture NUMA, les 2 vCPU sont affectés à un même nœud NUMA, donc sur le même processeur. Cette affectation permet d'optimiser l'accès à la mémoire par la machine virtuelle. Il s'agit du même principe que l'affinité CPU (section 3.1.1.2). Si une machine virtuelle dispose de plus de vCPU que ne dispose le nœud NUMA, la répartition sera effectuée de la même manière que sur une architecture UMA (les vCPU seront affectés à différents nœuds). La machine virtuelle ne bénéficiera donc pas de l'optimisation qu'apporte l'architecture NUMA.
Il en est de même avec la quantité de mémoire allouée à la machine virtuelle. Si cette dernière dispose d'une mémoire allouée supérieure à la capacité du nœud NUMA, l'hyperviseur gérera cette machine virtuelle de la même manière que sur une architecture UMA. La mémoire de la machine virtuelle se retrouvera dispersée sur différents nœuds, impliquant des temps de latence lors de l'accès à la mémoire dû à ce partage. La quantité de mémoire disponible pour chaque nœud est équitablement partagée entre ces derniers. Par exemple, pour un serveur composé de 2 processeurs et 32 Go de mémoire, chaque nœud NUMA sera alors composé d'un processeur et de 16 Go de mémoire (moins une réservation de l'hyperviseur pour la console de services).

NUMA Local Memory
Ce compteur permet de récupérer le pourcentage de mémoire allouée à la machine virtuelle, présente localement sur un nœud. Si toute la mémoire allouée à la machine virtuelle est présente sur un même nœud, la valeur de ce compteur sera de 100%.

3.1.2.4 Ballooning

En raison de l'isolement de la machine virtuelle, le système d'exploitation invité ne sait pas qu'il est en cours d'exécution dans une machine virtuelle et ne connaît pas les états des autres machines virtuelles s'exécutant en parallèle sur le même hôte. Lorsque l'hyperviseur exécute plusieurs machines virtuelles et que la quantité de mémoire physique machine devient faible, aucune des machines virtuelles ne peut détecter une surcharge mémoire au niveau serveur. Sans mécanisme dédié, l'hyperviseur n'aurait pas d'autre choix que de swapper des pages mémoires. Cependant, l'hyperviseur n'est pas celui qui est le plus au fait de savoir quelles sont les pages mémoire qui doivent être swappées. De ce fait, il est donc plus astucieux de forcer le système d'exploitation invité à swapper lui-même les pages mémoire. Le ballooning [Wal02, SFS06] est une technique de réclamation mémoire, permettant de rendre possible ce procédé et ainsi permettre de résoudre le manque de mémoire physique. Pour que cette technique soit utilisée, il faut qu'au préalable les outils additionnels soient installés dans les machines virtuelles. Ces outils permettent entres-autres d'établir une communication, via des canaux privés, entre la machine virtuelle et l'hyperviseur. Au sein des ces outils, un pilote gérant le ballonning est présent. Ce dernier sonde l'hyperviseur afin de savoir s'il a besoin de récupérer de la mémoire. Si oui, l'hyperviseur va envoyer au pilote une valeur cible. Cette valeur cible correspond à la quantité de mémoire réclamée par l'hyperviseur, que la machine virtuelle doit donc libérer. Le pilote "gonfle" alors le ballon en allouant des pages mémoire de la mémoire physique cliente.
Afin d'expliquer son fonctionnement, nous allons détailler les différents processus du ballooning dans un exemple. Une machine virtuelle dispose de quatre pages mémoire physique cliente mappées sur la mémoire physique machine. Deux de ces pages sont utilisées par une application s'exécutant au sein de la machine virtuelle. Les deux autres pages sont identifiées comme libres par le système d'exploitation invité. Cepen-

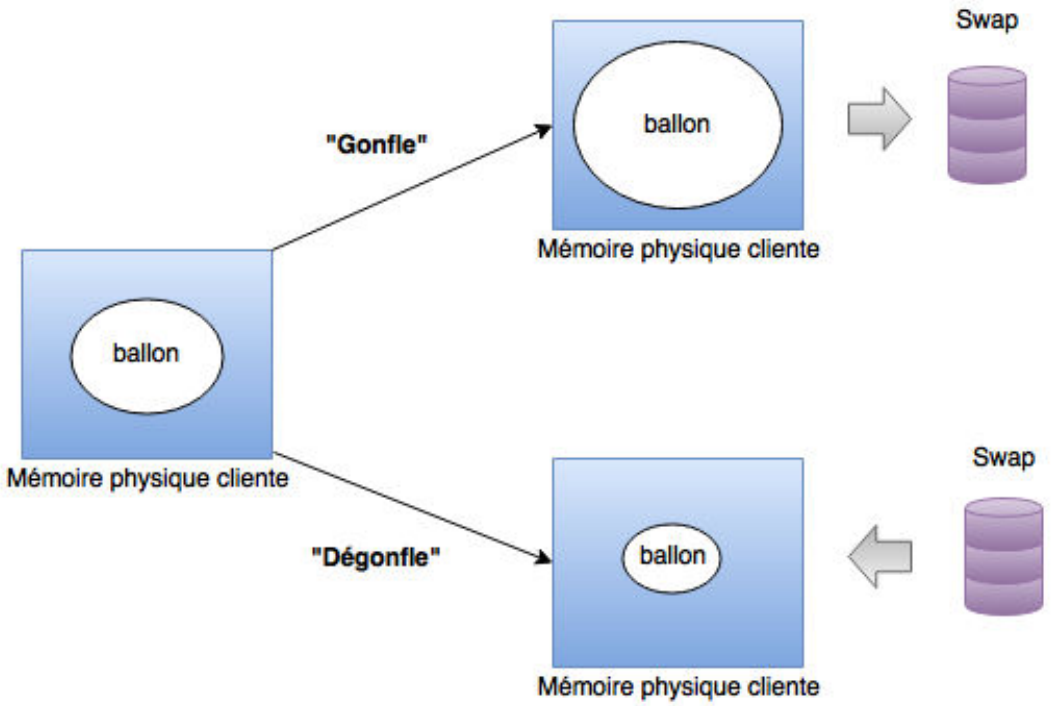


FIGURE 3.3 : Illustration du ballooning [Wal02]

dant, tant que l'hyperviseur ne peut identifier ces deux pages mémoire libres, il ne peut pas libérer les pages mémoire de la mémoire physique machine associées. En supposant que l'hyperviseur doit récupérer deux pages, il va donner au pilote présent dans la machine virtuelle une valeur cible de 2 pages. Après l'obtention de la taille cible du ballon, le pilote va "épingler" deux pages mémoire de la mémoire physique cliente à l'intérieur de la machine virtuelle (les deux pages mémoires "libres"). "Épingler" des pages mémoire signifie dans ce contexte, le fait que le système d'exploitation invité s'assure qu'elles ne soient pas accédées par une application pendant tout le processus de réclamation. Le pilote informe ensuite l'hyperviseur des pages mémoire qui ont été "épinglées" dans le but que ce dernier puisse libérer les pages mémoire physique machine associées. L'hyperviseur peut récupérer en toute sécurité ces pages mémoire dans la mesure où aucun processus de la machine virtuelle n'accédera intentionnellement à ces pages. Lorsque l'hyperviseur décide de dégonfler le ballon, en fixant une taille cible du ballon plus petite, le pilote libère les pages mémoire "épinglées" au niveau de la mémoire physique cliente, les rendant à nouveau disponibles.

Le principe du ballooning expliqué, nous nous intéressons maintenant aux compteurs qui permettent de suivre son utilisation :

- **Taille cible du ballon** : Ce compteur permet d'obtenir la taille cible du ballon affectée à la machine virtuelle par l'hyperviseur. Il s'agit de la quantité de mémoire réclamée par l'hyperviseur, qui doit être libérée par la machine virtuelle.

- **Taille du ballon** : Ce compteur permet d'obtenir la mémoire physique cliente actuellement libérée par la machine virtuelle. Il s'agit des pages mémoire qui ont été "épinglées" par le driver, c'est à dire celles qui ont été swappées par la machine virtuelle et déclarées comme disponible pour l'hyperviseur. Si la taille du ballon est inférieure à la taille cible, cela signifie que le ballon est en train de se gonfler ("inflate") dans le but d'atteindre cette valeur cible. A l'inverse si la taille du ballon est supérieure à la cible, cela signifie que le ballon est en train de se dégonfler ("deflate").

Le ballooning permet de libérer de la mémoire physique machine à moindre coût. Cependant, ce mécanisme impose l'installation d'un pilote au sein des machines virtuelles. Cette solution n'est alors pas toujours envisageable. Le mécanisme de la compression mémoire est une autre technique d'optimisation de

l'utilisation mémoire, agissant de manière plus transparente et ne nécessitant aucune installation de pilote dans les systèmes d'exploitation invités. Cette technique est expliquée dans la sous-section suivante.

3.1.2.5 Compression mémoire

En utilisant la compression mémoire [TG05], l'hyperviseur va stocker dans une zone mémoire des pages mémoires qui auraient dû être swappées. Seules ces pages mémoires sont sujettes à la compression. L'hyperviseur détermine si une page mémoire peut être compressée en contrôlant le taux de compression de cette dernière. La page mémoire est compressée si son taux de compression est au moins supérieur à 50%, sinon cette dernière est swappée.
Afin d'expliquer son fonctionnement, nous allons détailler les différentes étapes dans un exemple. Une machine virtuelle dispose de quatre pages mémoire physique cliente mappées sur la mémoire physique machine. L'hyperviseur a besoin d'une quantité de mémoire de 8 Kb (kilo-bytes) au niveau de la mémoire physique machine. Sans compression mémoire, deux pages seraient swappées. En effet, chaque page mémoire ayant une taille de 4 Kb, il faut donc deux pages pour obtenir les 8 Kb. En utilisant la compression mémoire, les quatre pages mémoires seront compressées. Chaque page mémoire compressée a une taille passant de 4 Kb à 2 Kb. Pour obtenir les 8 Kb, il faut donc compresser les 4 pages mémoires (4*2 Kb).
La compression mémoire est plus performante que le swapping dans la mesure où le prochain accès à la page mémoire compressée ne provoque uniquement qu'une décompression de cette dernière. Cette décompression reste plus rapide qu'un accès au fichier de swap dans la mesure où ce dernier implique des entrées/sorties importantes. Certes, la décompression implique une consommation processeur. Cependant, nous verrons dans la suite de cette thèse que la consommation processeur liée à cette décompression n'est pas un problème. Les compteurs qui permettent ainsi de détecter l'utilisation de la compression mémoire sont : la "mémoire compressée", qui correspond à la quantité de mémoire stockée dans le cache de compression ; le débit de compression mémoire exprimé en ko/s et le débit de décompression mémoire exprimé en ko/s.

Lorsque l'hyperviseur ne peut plus utiliser cette technique, le mécanisme classique du swapping est alors employé.

3.1.2.6 Swapping

Lorsqu'une machine virtuelle est démarrée, l'hyperviseur créé un fichier séparé, utilisé en cas de besoin. Ainsi, lorsque nécessaire, l'hyperviseur peut utiliser ce fichier en complément de la mémoire physique cliente en déplaçant les pages de cette mémoire dans un espace d'échange, appelé swap. Lorsque des pages mémoire sont écrites dans le swap (swap out), elles libèrent de la mémoire physique machine qui peut être utilisée pour d'autres machines virtuelles. Si une application tente d'accéder à une page mémoire swappée, l'hyperviseur déclenchera une faute de page, qui sera traitée par le système d'exploitation de la machine virtuelle. Ce dernier recherchera la page mémoire souhaitée depuis le fichier de swap (swap in). Le swapping est une technique très coûteuse en performances. En effet, au lieu d'impacter uniquement la mémoire comme c'est le cas pour le partage mémoire et le balloonning, il nécessite également l'accès au disque dur pour lire les informations stockées dans le fichier de swap. La latence correspondante peut être de quelques millisecondes, impactant donc les performances de la machine virtuelle. Cette technique est donc utilisée en dernier recours par l'hyperviseur. Les compteurs à suivre pour la détection du swapping sont la "mémoire swappée", qui correspond à la quantité de mémoire physique client actuellement écrite sur un fichier de swap (ce fichier de swap est stocké durant toute la durée d'exécution de la machine virtuelle), le taux de lecture (swapin), correspondant au taux de lecture du fichier de swap (l'information étant alors transférée depuis le fichier de swap vers la mémoire active) et le taux d'écriture (swapout), correspondant au taux d'écriture dans le fichier de swap (l'information étant alors transférée depuis la mémoire active vers le fichier de swap).

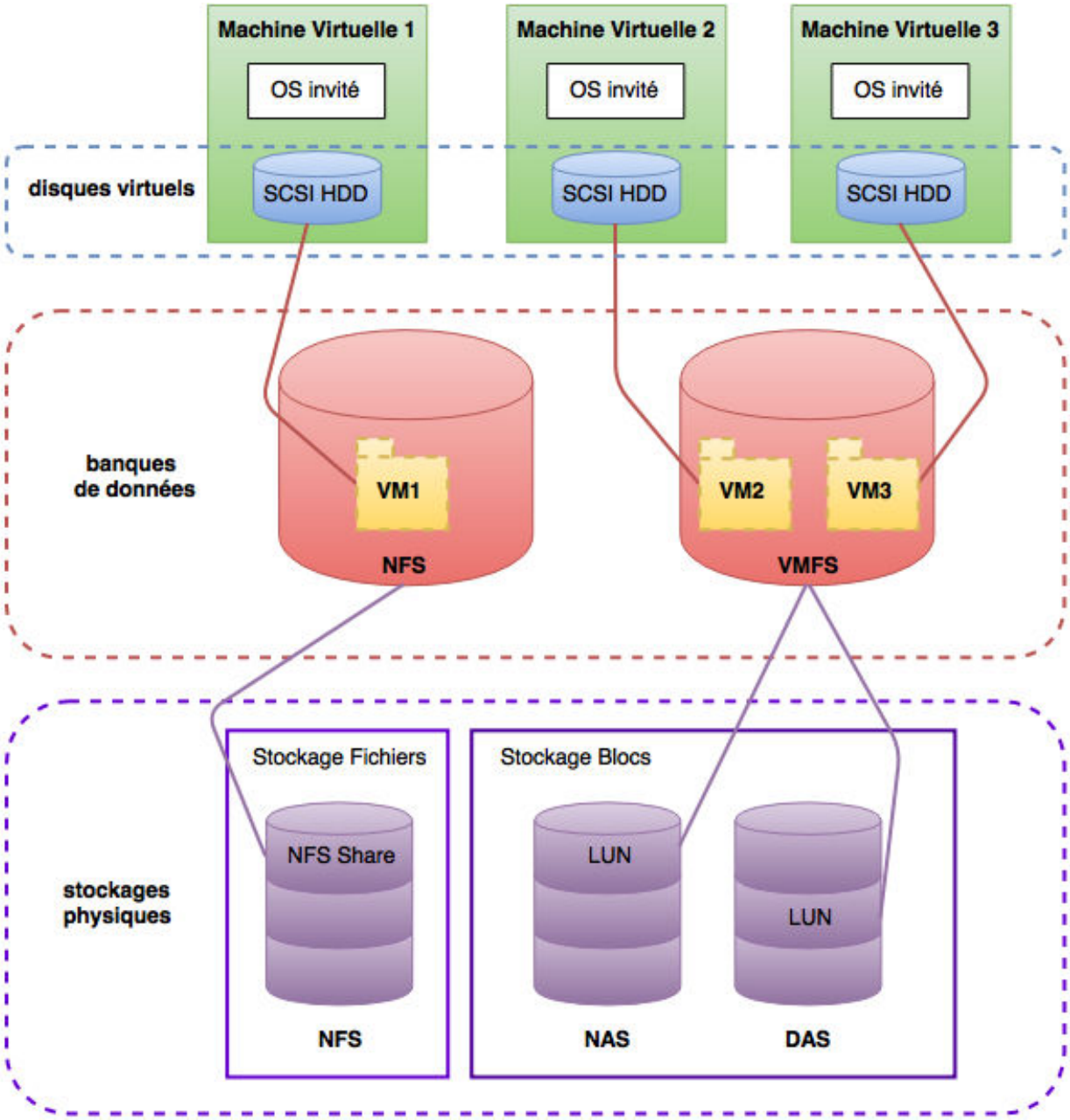


FIGURE 3.4 : L'architecture stockage

3.1.3 Les compteurs disque

Les bonnes performances d'une machine virtuelle ne se résume pas uniquement en l'étude des ressources processeur et mémoire. L'accès au disque de façon optimale est également un enjeu important. En effet, une application qui souffre d'une augmentation de ses temps de réponse sera ressentie par l'utilisateur. Il est alors important d'étudier finement la ressource disque afin d'éviter/prévoir d'éventuelles dégradations de performances. Dans le cas des infrastructures virtualisées, les machines virtuelles partagent les mêmes périphériques de stockage, ce qui par conséquent peut éventuellement dégrader les performances. Il est donc important de monitorer les indicateurs clés.

3.1.3.1 Architecture stockage

Chaque machine virtuelle est stockée comme un ensemble de fichiers dans un répertoire d'une banque de données (en anglais, datastore). Une banque de données peut être considérée comme une vue, disposant d'une capacité de stockage, faisant abstraction de la technologie de stockage sous-jacente, qui peut être par-

tagée entre plusieurs hyperviseurs afin de stocker plusieurs machines virtuelles. Chaque banque de données est un volume physique au format NFS (Network File System) ou VMFS (Virtual Machine File System). VMFS est un système de fichiers basé sur les blocs, optimisé pour la virtualisation, c'est à dire pour les accès multiples aux fichiers des machines virtuelles. En effet, un système de fichier dit "traditionnel" ne peut être accédé en lecture/écriture que par un seul serveur à la fois. Ainsi, sur un volume VMFS, plusieurs serveurs peuvent lire et écrire des données simultanément. Chaque volume VMFS peut s'éteindre sur un ou plusieurs sous-systèmes de stockage, appelé LUN. Un LUN (de l'anglais Logical Unit Number ou Numéro d'Unité Logique) est une référence logique à une partie du sous-système de stockage, qui peut être constituée d'une section de disque (partition), d'un disque entier, d'une section d'une baie de disques ou bien de la totalité d'une baie de disques. La capacité de stockage d'une banque de données VMFS peut être étendue à chaud, en rattachant une LUN précédemment créée. Ceci a pour avantage d'augmenter la capacité de la banque de données sans avoir à éteindre les machines virtuelles qui sont stockées sur cette dernière. Si une LUN devient inaccessible, seules les machines virtuelles utilisant cette LUN seront affectées.

DAS (Direct Attached Storage)
Un DAS désigne un périphérique de stockage directement lié à un matériel et donc non accessible directement par un autre matériel. Un exemple simple d'une périphérique DAS est un disque dur interne d'un ordinateur. L'architecture DAS est la première architecture de stockage utilisée dans les centres de données et continue d'être employée dans les entreprises. L'avantage de cette architecture est son faible coût. En effet, en investissant dans un ou plusieurs serveurs, possédant des disques durs incorporés, le coût du stockage en lui-même est faible. Cependant, cette architecture dispose d'inconvénients. Les besoins de stockages croissant sans cesses, l'ajout de nouveaux disques durs au sein d'une architecture DAS impose l'arrêt complet de la machine. De ce fait, pour des environnements critiques (temps de réponses rigoureux), cette architecture n'est pas un bon choix. Son évolutivité est limitée. En effet, l'interface de stockage d'un serveur ne supporte qu'un nombre limité de disques durs. De plus, il ne permet pas le partage de données entre plusieurs serveurs.

NAS (Network Attached Storage)
Un NAS désigne un périphérique de stockage spécialisé dans la mise à disposition de fichiers sur le réseau. Un périphérique NAS classique est un matériel complètement indépendant disposant de son propre système d'exploitation, d'un ensemble de périphérique de stockage et d'une interface de gestion (gestion des accès, des quotas, sauvegardes automatisées, réplications). Contrairement au DAS où le serveur communique avec le périphérique de stockage via un bus, le serveur communique avec le périphérique de stockage via le réseau. L'un des avantages d'un système de type NAS est sa facilité d'installation. La plupart de ces périphériques sont prêts à l'emploi, c'est à dire qu'ils ne requièrent aucune intervention particulière. Ils ont uniquement besoin d'être relié au réseau Ethernet. Certains périphériques NAS permettent le branchement et débranchement de disques durs "à chaud", ces derniers n'ont pas besoin d'être éteints lorsqu'il faut intervenir sur le périphérique.

SAN (Storage Area Network)
Un SAN désigne un réseau physique en fibre optique (en anglais Fiber Channel, FC) dont l'objectif est de connecter les différents périphériques de stockage, appelés baies de disques, aux serveurs via des périphériques réseaux tels que des switchs ou des routeurs. Le SAN est un réseau secondaire, exclusivement réservé au stockage. Il dispose de plusieurs topologies :

- "point à point" : c'est la topologie la plus simple. Elle relie 2 entités, un serveur et une baie de disque. Elle offre la meilleure bande passante.
- "boucle arbitraire" : c'est une topologie en forme d'anneau, reprenant les principes du protocole Token Ring. L'information passe d'un périphérique à un autre jusqu'à son destinataire. Les périphériques, qui peuvent aller jusqu'au nombre de 126, se partagent donc la bande passante. Cette topologie

à l'avantage d'être peu coûteuse. En effet, les serveurs et les baies de disques étant reliés entres-eux, il n'y a pas de périphériques réseaux à acheter. Cependant, si un seul nœud est inaccessible, c'est toute l'architecture qui tombe. Plus aucune communication ne peut transiter d'un nœud à un autre. De plus, l'information transitant de nœud en nœud, cela entraîne de la latence.

- "Fabric" : c'est une topologie en forme de maille dans laquelle chaque serveur est relié aux baies de disques par un périphérique réseau. Elle apporte une meilleure disponibilité que la topologie précédente dans la mesure où si un nœud tombe, elle n'a pas d'impact sur les autres nœuds. De plus, la bande passante n'est pas partagée.

Les SAN sont principalement utilisés pour des applications nécessitant des performances importantes, telles que des applications de bases de données ou pour la virtualisation. La mise en place d'un SAN est très onéreuse dans la mesure où ce dernier est un réseau qui exigent du matériel spécifique. De plus, son administration et sa maintenance sont complexes et nécessitent des ressources humaines dédiées. Quelque soit l'architecture retenue, les métriques de suivi des performances disque sont communes. Ces différentes métriques sont présentées dans les sous-sections suivantes.

3.1.3.2 IOPS

Le nombre d'opérations d'entrées/sorties par seconde (en anglais, I/O Operations Per Second) est un indicateur permettant de suivre l'activité d'un périphérique de stockage. Une opération est effectuée pour chaque lecture ou écriture sur le disque. Si ce dernier héberge plusieurs machines virtuelles, il peut y avoir plusieurs milliers d'IOPS simultanément. Cet indicateur permet ainsi de suivre l'activité disque des machines virtuelles mais également des centres de données. Chaque périphérique de stockage dispose d'un nombre d'IOPS limité. Le suivi des IOPS permet de détecter si le périphérique de stockage n'est pas surchargé. La performance d'un périphérique de stockage est essentiellement basée sur la vitesse de rotation. En effet, un disque dur standard ayant une vitesse de rotation de 7200 rpm (rotations par minutes) est capable de supporter environ 75 à 100 IOPS alors qu'un disque dur de 15000 rpm supporte entre 175 et 210 IOPS. Pour les SSD, qui ne sont pas liés à des composants mécaniques, peuvent quant à eux supporter environ 5000 IOPS. Les niveaux de RAID (Redundant Array of Independent Disks pour "regroupement redondant de disques indépendants") ont un impact sur le nombre d'IOPS supportées par le périphérique de stockage. Plus le niveau est élevé, plus la pénalité est élevée dans la mesure où les entrées/sorties sont répétées sur plusieurs disques.

Commandes échouées

Le nombre de commandes échouées est un indicateur permettant de savoir si un disque est capable de répondre aux requêtes. Ce compteur comptabilise les requêtes qui n'ont pas abouties dû au fait que le disque ne réponde pas. Trop de commandes échouées créé trop de tentatives affectant les performances.

3.1.3.3 Latence disque

La latence est un indicateur qui se concentre sur le temps mis par un système pour effectuer une opération (lecture ou écriture) sur un périphérique de stockage. Ce temps est généralement exprimé en millisecondes. Chaque fois qu'une machine virtuelle réalise une opération en lecture ou écriture sur disque virtuel, cette demande part du système d'exploitation invité jusqu'au périphérique de stockage physique. Le long de ce cheminement, des goulots d'étranglement peuvent se produire à différents niveaux. En effet, en allant du système d'exploitation invité jusqu'au disque dur, la donnée passe par une interface SCSI virtuelle, le VMKernel et une interface de stockage physique au niveau de l'hyperviseur pour atteindre le périphérique de stockage cible depuis le réseau de stockage. De ce fait, la latence totale, celle vue par le système d'exploitation invité, est le temps mis pour parcourir l'ensemble de cette pile.

Dans la mesure où des goulots d'étranglement peuvent se produire à plusieurs niveaux, il existe alors différents indicateurs permettant de mesurer la latence à ces différents niveaux.

- **Kernel Average Latency (KAVG)** Il s'agit de la latence au niveau Kernel, c'est à dire au niveau de l'hyperviseur. Cette métrique doit être à zéro dans un environnement idéal. Si cette dernière est au dessus de 2 millisecondes, cela signifie que l'hyperviseur a un problème de performance.

- **Device Average Latency (DAVG)** Il s'agit de la latence au niveau du périphérique de stockage. Elle comprend le temps d'aller-retour entre le HBA (Host Bus Adapter) et le disque. Ce compteur permet de surveiller un éventuel goulot d'étranglement sur le périphérique. Un goulot d'étranglement sur un disque impacte les performances de toutes les machines virtuelles hébergées sur ce disque. Il est alors nécessaire de migrer les machines virtuelles ayant d'importantes entrées/sorties disque afin d'équilibrer les charges sur les disques. Pour des performances optimales, la Device Average Latency ne doit pas excédée 20 millisecondes.[Mun15]

- **Guest Average Latency (GAVG)** Il s'agit de la latence vue par la machine virtuelle. Cette latence est la somme de la Kernel Average Latency et la Device Average Latency.

La dernière ressource importante à superviser est le réseau. L'étude ce dernier est décrite dans la sous-section suivante.

3.1.4 Les compteurs réseau

3.1.4.1 Architecture réseau

Chaque machine virtuelle dispose d'une carte réseau virtuelle (vNIC pour virtual Network Interface Card). Chaque carte virtuelle apparaît sur le réseau comme une carte physique, disposant des mêmes fonctionnalités, d'une adresse MAC, d'une ou plusieurs adresses IP, connectée sur le réseau via Ethernet. Pour permettre à l'hyperviseur de gérer toutes ces interfaces virtuelles, ce dernier dispose d'un ou plusieurs switchs virtuels, appelés vSwitchs. Chaque vSwitch peut disposer de plusieurs groupes de ports (Port Groups). Dans le monde physique, chaque interface réseau est reliée via un câble à un port du switch. Dans le monde virtuel, chaque vNIC est connectée à un Port Group. Ces groupes, identifiés par un label unique, peuvent être différemment configurés de sorte à améliorer la sécurité, les performances, la disponibilité et la gestion du réseau. Toutes les machines virtuelles dont les vNIC sont connectées à un même Port Group, appartiennent au même réseau virtuel (vLAN). De plus, un même Port Group peut appartenir à différents vSwitch. Ainsi, deux machines virtuelles hébergées sur deux hyperviseurs différents sont sur le même réseau virtuel si leurs vNIC respectives sont connectées au même Port Group. Les vSwitch sont connectés aux interfaces réseaux physiques de l'hyperviseur via des connexions uplinks. Une connexion uplink est une connexion entre deux switchs. Dans ce cas, il s'agit d'une connexion entre un vSwitch et le switch physique. Généralement, un hyperviseur dispose de plusieurs interfaces réseaux afin d'équilibrer les charges ou pour pallier à une défaillance d'un périphérique. De cette manière la défaillance est transparente pour les machines virtuelles qui continuent d'être reliées au réseau. Les vSwitch peuvent supporter plusieurs vLAN (virtual Local Area Network). Ainsi, les compteurs à monitorer permettant d'analyser l'activité réseau des machines virtuelles sont les "paquets émis/reçus perdus", permettant d'obtenir le nombre de paquets émis ou reçus perdus et les "paquets émis/reçus erronés", permettant d'obtenir le nombre de paquets émis ou reçus comportant des erreurs.

3.1.5 Disponibilités des métriques de performances au sein des hyperviseurs

Les précédentes études des compteurs ont été réalisées à partir de l'hyperviseur le plus utilisé au sein des entreprises. Il s'agit de l'hyperviseur ESX, solution développée par la société VMware. Pour faire un parallèle entre cette solution et les autres hyperviseurs, nous avons recensé la disponibilité des différents compteurs,

	VMware ESX/ Vcenter 5.5U2	Hyper-V 2012	XenServer 6.X	KVM X.X
CPU Used	√	√	√	√
CPU Ready	√	√	√	x
CPU Wait	√	√	(?)	x
CPU Run	√	x	√	x
CPU Idle	√	x	√	x
CPU Costop	√	x	(?)	x
Memory Allocated	√	√	√	√
Memory Consumed	√	x	√	x
Memory Active	√	x	x	x
Memory Granted	√	x	x	x
Memory Overhead	√	x	x	x
Memory Shared	√	x	x	x
Memory Page Sharing	√	x	x	√
Memory Ballooning	√	√	√	√
Memory Compression	√	x	x	x
Memory Swapping	√	√	√	√
Disk Read	√	√	√	√
Disk Write	√	√	√	√
Disk Latency	√	√	√	x
Disk IOPS	√	√	√	x
Disk Aborted Commands	√	(?)	(?)	x
Net Transmitted	√	√	√	√
Net Received	√	√	√	√
Net Errors	√	x	√	x

TABLE 3.1 : Disponibilité des compteurs dans les différents hyperviseurs

que nous avons retenus lors nos études dans le cadre d'une analyse fine des machines virtuelles, au sein du tableau 3.1. De ce fait, l'étude de le disponibilité des compteurs a été faite pour les hyperviseurs Hyper-V 2012 de Microsoft, XenServer 6.X de Citrix ainsi que l'hyperviseur open-source KVM X.X. La première observation que nous pouvons faire concerne l'indisponibilité des compteurs liés à la mémoire. En effet, mis à par l'hyperviseur VMware, aucun autre hyperviseur ne dispose de compteurs permettant d'analyser précisément l'activité mémoire des machines virtuelles. A l'inverse les compteurs pour le suivi des activités disques et réseaux sont en grande partie disponibles pour tous les hyperviseurs, sauf pour l'hyperviseur KVM qui ne dispose pour le disque que des débits de lecture et d'écriture. Concernant les compteurs processeur, l'hyperviseur XenServer est, derrière l'hyperviseur VMware, le plus complet puisqu'il dispose de tous les compteurs permettant une analyse fine de l'activité processeur des machines virtuelles s'exécutant sur ce dernier. En revanche, l'hyperviseur Hyper-V 2012 dispose que d'une partie des compteurs (Utilisation, Ready, Wait) et l'hyperviseur KVM uniquement que le compteur permettant de connaître l'utilisation processeur. Cette étude croisée de la disponibilité des différents compteurs de performances permet de mettre en avant la lacune de l'hyperviseur open-source KVM.

Notre objectif étant de mesurer finement l'activité des machines virtuelles, l'indisponibilité des compteurs sous l'environnement KVM nous a conduit à développer un outil d'introspection non-intrusive de machines virtuelles. La contribution liée à cette introspection est détaillée dans la section suivante.

3.2 L'Introspection

L'introspection de machines virtuelles (VMI, Virtual Machine Introspection) est le processus d'inspection de l'état du système d'exploitation d'une machine virtuelle (OS invité), depuis le système hôte (hyperviseur). La VMI est proposée pour la première fois en 2003 dans les travaux de Garfinkel et Rosenblum [GR+03]. La VMI a été adoptée dans de nombreuses applications de sécurité tels que les systèmes de détection d'intrusion (IDS) [GR+03, PCSL08], la recherche de processus malveillants [DRSL08, DGLZ+11, JWX07], analyses fines de la mémoire physique [DGPL11, HN08] et le monitoring de processus [SLCL09]. La surveillance des ressources systèmes consommées par les machines virtuelles sur un serveur est un élément clé pour garantir le bon fonctionnement d'une plate-forme IaaS.

3.2.1 Définition

Dans [GR+03], Garfinkel et al. définissent l'introspection comme une méthode permettant d'inspecter les machines virtuelles depuis le système hôte, afin de surveiller le comportement des systèmes d'exploitation invités ainsi que des applications s'exécutant au sein de ces dernières. Dans la plupart des techniques d'introspection, l'observation d'une machine virtuelle est réalisée depuis une autre machine virtuelle ou directement au niveau de l'hyperviseur. Il s'agit alors d'une introspection non intrusive. Cette non intrusivité est due principalement au fait, que dans de très nombreux contextes, essentiellement pour des raisons de sécurité et de maintenance, les clients ne souhaitent pas installer au sein de leurs machines virtuelles des composants logiciels (agents) non maîtrisés. Réaliser l'introspection depuis un autre environnement permet également de garantir le bon fonctionnement de l'outil d'introspection (service non stoppé par l'utilisateur et protégé contre d'éventuelles attaques). Au cours des dernières années, cette technique a fait l'objet de plusieurs contributions (notamment dans le domaine de la sécurité), ainsi que diverses méthodes et applications ont été proposées pour inspecter les machines virtuelles [BAL15]. Différentes approches sont utilisées dans ces contributions. Pour analyser la mémoire des machines virtuelles, certains nécessitent de recompiler le noyau Linux de la machine virtuelle afin d'y ajouter les options de déverminage. C'est le cas par exemple pour Livewire [GR+03] et Insight-VMI [SPE11]. D'autres approches consistent à modifier et analyser le code source du noyau Linux (Nitro [PSE11]) ou à parcourir les structures de données du noyau Linux (LibVMi [XLXJ12, Pay12]).

Dans le détail [WTM+14], la VMI consiste à analyser l'empreinte mémoire et le disque des machines virtuelles, depuis l'extérieur sans interférer sur leur fonctionnement. Ainsi, leur comportement interne est ensuite interpréter à partir des informations bas-niveau obtenues au niveau de l'hyperviseur.

3.2.2 Contexte

La couche VMM (pour Virtual Machine Monitor, section 2.2) est une couche logicielle responsable de la mise à disposition de ressources physiques et de l'isolation des machines virtuelles. L'isolation apportée par la VMM est une fonctionnalité essentielle pour la sécurité des machines virtuelles. Toutes les interactions entres ces dernières ainsi que l'accès aux ressources physiques sont gérées par la VMM. Celle-ci permet entres-autres, qu'une machine virtuelle n'accède pas à la mémoire d'une autre machine virtuelle, ou qu'un incident sur une machine virtuelle n'affectera pas le fonctionnement des autres machines virtuelles. Cependant, bien que la VMM apporte une couche de sécurité, la virtualisation apporte ses propres risques de sécurité, non pas principalement en raison de la technologie elle-même mais du fait que les serveurs virtualisés sont moins sécurisés que les serveurs physiques qu'ils remplacent. Les problèmes de sécurité auxquels les environnements virtualisés font face sont alors plus complexes à résoudre que pour des environnements physiques traditionnels. En effet, l'activité ne se concentre plus uniquement sur un seul environnement mais potentiellement plusieurs. Un système malveillant peut s'exécuter au sein d'une ou plusieurs machines virtuelles, compromettant le bon fonctionnement de l'infrastructure. Il est alors néces-

saire pour un administrateur de surveiller son infrastructure afin de s'assurer que l'environnement est sain et qu'il fonctionne comme prévu. Il est presque impossible de diagnostiquer un problème manuellement dans une infrastructure virtualisée puisque le dysfonctionnement provenant d'une machine virtuelle peut impacter les performances des autres machines virtuelles. Sans surveillance, l'administrateur ne peut pas avoir l'assurance que son environnement est sain.

La VMI exploite les capacités offertes par la VMM de trois façons [GR+03] :

- Isolation : l'isolation permet d'assurer le fonctionnement de l'outil réalisant l'introspection. En effet, si une machine virtuelle dispose d'un processus malveillant, ce dernier est cloisonné et ne peut accéder à d'autres machines virtuelles, empêchant alors d'altérer le fonctionnement de l'outil d'introspection (qui s'exécute en dehors de la machine virtuelle compromise)

- Inspection : la VMM dispose d'un accès complet aux systèmes d'exploitation invités afin de contrôler les accès mémoire, les registres processeur et les opérations d'Entrées/Sorties. Cette visibilité dont bénéficie la VMM permet l'inspection complète des systèmes d'exploitation invités, rendant un code malveillant difficilement indétectable.

- Interposition : la VMM surveille le fonctionnement des machines virtuelles en interceptant les requêtes telles que des instructions processeur. Cette fonctionnalité peut être utilisée par la VMI dans le but de détecter des modifications d'états non autorisées.

3.2.3 Les principaux frameworks basés sur l'introspection

Dans [GR+03], Garfinkel et al. présentent Livewire, le premier prototype de système de détection d'intrusions basé sur l'introspection. Livewire est fondé sur une version modifiée de VMware Workstation, outil permettant la virtualisation de systèmes d'exploitation. Il ne fonctionne donc que sur l'hyperviseur VMware. Son architecture est composée de trois éléments. Le premier composant, *VMM Interface*, est une interface fournissant un accès à la couche VMM (Virtual Machine Monitor) de l'hyperviseur VMware. Cette interface est notamment utilisée pour l'envoi de commandes d'inspection (examiner l'état des machines virtuelles, analyser le contenu de la mémoire) et d'administration (suspendre et relancer une machine virtuelle). Le second composant, *OS Interface Library*, est utilisé dans le but de fournir des informations sur la machine supervisée. Des informations concernant aussi bien l'environnement interne de la machine virtuelle (par exemple la liste des processus) qu'externe, tel que son état. Pour les informations externes, Livewire nécessite une version améliorée de l'outil *crash* permettant ainsi d'interpréter l'état de la machine exportée par la *VMM Interface*. Le dernier composant, *Policy Engine*, écrit en langage Python, interprète les informations envoyées par les deux autres composants et décide si le système est corrompu ou pas.

Shneider et al. propose dans [SPE11], Insight-VMI, un outil d'introspection de machines virtuelles écrit en langage C++. Cet outil analyse la mémoire physique. Ainsi, l'outil est capable de traiter des images de la mémoire ou d'analyser directement la mémoire physique du système invité à partir de n'importe quel hyperviseur lui fournissant un accès en lecture. Aucun agent n'est installé à l'intérieur des machines virtuelles à superviser. Insight-VMI fournit un shell pour analyser l'état d'une machine virtuelle, permettant ainsi la consultation des structures de données du noyau et des variables globales. Insight-VMI fournit également un moteur JavaScript, permettant l'exécution de scripts afin d'automatiser les tâches d'introspection. L'analyse de la mémoire des machines virtuelle nécessite la recompilation du noyau Linux afin d'y ajouter les options de déverminage. Les sources d'Insight-VMI propose des scripts dont le script make-debug-kpkg permettant de modifier les noyaux de machines virtuelles sous Debian et Ubuntu. Ce script ajoute les options souhaitées et lance la compilation. Cette opération a l'inconvénient d'être longue (plusieurs heures) et coûteuse en stockage (environ 20 Go). De plus, le fonctionnement d'Insight-VMI obligeant la modification du noyau, l'analyse de machines virtuelles sous Windows est donc non encore supportée.

Nitro [PSE11], est un framework compatble uniquement qu'avec l'hyperviseur KVM, à base de processeurs Intel 64 bits disposant du flag "vmx" (VT-x). De plus, il ne supporte que certaines versions du

noyau Linux. Actuellement, les versions officiellement compatibles sont 3.11 et 3.13. Il capture les appels systèmes produits par les machines virtuelles afin d'analyser leur comportement. Les appels systèmes sont des opérations de bas niveaux utilisés pour la communication entre l'espace utilisateur et l'espace noyau d'un système d'exploitation. A titre d'exemple, des appels systèmes sont utilisés lors de la manipulation de fichiers (open, read, write, close). Afin de capturer ces appels systèmes, Nitro nécessite un hyperviseur KVM modifié (chargement de deux kernel modules spécifiques).

VMwatcher [JWX07] supporte plusieurs hyperviseurs tels que KVM, XEN ou encore VMware. Il s'agit d'un système de détection d'intrusion utilisant les techniques d'introspection. Cet outil se focalise sur l'introspection mémoire et analyse les systèmes de fichiers dans le but de détecter des applications malveillantes. Il est compatible avec les systèmes invités Linux et Windows. Pour les systèmes Linux, de la même manière que LibVMi, VMwatcher nécessite le fichier System.map pour accéder aux symboles du noyau Linux. Pour les systèmes Windows, VMwatcher accède aux symboles du noyau en recherchant directement dans la mémoire physique.

La librairie LibVMi [XLXJ12, Pay12], écrite en langage C, a pour but de simplifier l'introspection de machines virtuelles en cours d'exécution. Cette librairie propose des fonctions pour analyser la mémoire des machines virtuelles à partir d'un hyperviseur XEN ou KVM. Il n'y a aucun outil à installer à l'intérieur des machines virtuelles observées. LibVMi nécessite des informations sur l'organisation de la mémoire des machines virtuelles à surveiller. Ces informations sont des adresses mémoires propres à chaque système d'exploitation. Le programme linux-offset-finder, disponible dans les sources de LibVMi, permet de récupérer facilement ces informations. Pour les machines virtuelles exécutant un système d'exploitation Linux, LibVMi a également besoin du fichier System.map, présent dans le répertoire /boot de la machine virtuelle. Ce fichier, créé lors de la compilation du noyau, fait le lien entre une adresse mémoire et un symbole donné (nom d'une variable, nom d'une fonction...).

3.2.4 Le fossé sémantique

Toutes les solutions d'introspection de machines virtuelles doivent résoudre le problème de "fossé sémantique" [CN01] dû au fait que ces solutions s'exécutent en dehors des machines virtuelles à monitorer. En effet, au niveau de l'hyperviseur, il est possible d'obtenir une vue de l'état des machines virtuelles à partir de données brutes. Ces données brutes permettent d'obtenir diverses informations tels que les registres processeur (tous les registres processeurs peuvent être lus par l'hyperviseur, ce dernier contrôlant entièrement la couche matérielle), les pages mémoire (l'ensemble de l'état de la mémoire des systèmes d'exploitation invités peuvent être observés. Cependant, les hyperviseurs ont seulement accès à des adresses physiques qui doivent être traduit en adresses virtuelles), les blocs des disques durs (sauf s'ils sont encryptés, les contenus des disques durs des machines virtuelles peuvent également être observés) ou encore les entrées/sorties (l'hyperviseur contrôle les entrées/sorties c'est à dire le trafic réseau et disque). A l'inverse, depuis l'intérieur des machines virtuelles, il est possible d'obtenir une vue comportant des informations sémantiques notamment la liste des processus s'exécutant au sein du système d'exploitation invité, divers compteurs de performance, les appels systèmes ou encore accéder au fichiers des systèmes d'exploitation. Il y a donc un fossé sémantique entre ce que nous observons au niveau de l'hyperviseur et ce que nous souhaitons obtenir [BAL15]. Il est ainsi formé par la grande différence entre les observations externes et internes et représente le principal défi dans l'élaboration d'un outil d'introspection.

Pour combler ce fossé sémantique et ainsi interpréter les données afin d'en extraire les informations sémantiques, une des clés est de disposer de connaissances dans le fonctionnement d'un système d'exploitation. En effet, le système d'exploitation invité dispose d'un ensemble de structures de données, par exemple pour la gestion des processus ou le système de fichiers. De ce fait, en rapprochant les informations bas-niveau obtenues depuis l'hyperviseur aux structures de données des systèmes d'exploitation, des informations sémantiques ainsi que leur état peuvent être déduits. En guise d'exemple, en analysant le contenu bas-niveau des pages mémoires d'une machine virtuelle et en utilisant les connaissances sur les structures

de données du système d'exploitation s'exécutant au sein de cette machine virtuelle, il est possible d'identifier les structures de données du noyau et par translation obtenir diverses informations sur les processus (nom, PID, etc...). De ce fait, pour réaliser ces rapprochements sémantiques il est indispensable d'utiliser la table des symboles du système d'exploitation de la machine virtuelle surveillée qui permet d'interpréter les données bas-niveau. Cette table fait le lien entre une adresse mémoire et un symbole donné. Cependant, cette approche dispose de certaines limites. La première et la plus contraignante d'entres-elles, concerne la supposition que les tables des symboles ne seront pas modifiées durant toute l'exécution des systèmes d'exploitation des machines virtuelles surveillées. Une hypothèse délicate dans la mesure où les systèmes d'exploitations sont souvent mis à jour. Ce qui dans certains cas peuvent être des mises à jour du noyau et par conséquent impacter la table des symboles. La seconde limitation étant que l'outil d'introspection doit disposer de toutes les tables de symboles des machines virtuelles monitorées.

3.3 Contribution

Cette section traite de notre contribution qui a portée sur le développement d'une extension à LibVMi pour capturer et surveiller, directement depuis l'hyperviseur, les processus s'exécutant au sein du système d'exploitation de la machine virtuelle. Cette contribution a été réalisée en partenariat avec Sylvie Laniepce d'Orange Labs Caen. L'extension développée permet d'une part de capturer l'activité processeur et mémoire de manière non intrusive, c'est-à-dire sans installer un agent au sein de la machine virtuelle et d'autre part de détecter les processus ayant une activité mémoire abusive ou anormale. Cette analyse fine des consommations peut être utilisée dans plusieurs contextes tels que la sécurité, la tolérance aux pannes, les fuites mémoires ou encore l'optimisation de la consommation énergétique.

Située au niveau de l'hyperviseur, notre sonde logicielle ne dispose que de l'empreinte mémoire "brute" des machines virtuelles, c'est à dire, l'ensemble des pages mémoires utilisées par la machine virtuelle. Pour nous permettre une interprétation des données brutes il est nécessaire à la fois de disposer d'un outil facilitant le parcours dans l'empreinte mémoire (LibVMI) mais aussi d'informations sémantiques permettant de relier les zones mémoires au code logiciel du système d'exploitation invité. L'outil LibVMI apporte quelques abstractions sémantiques, mais nécessite d'être étendu pour analyser les taux d'occupation mémoires des processus s'exécutant sur le système invité. Pour ce faire, notre approche consiste à s'appuyer sur les structures de gestion interne du système d'exploitation invité et de collecter les métriques appropriées de ces structures internes. Un premier travail fût d'étudier la gestion de la mémoire ainsi que les structures du noyau Linux, travaux présentés dans les sous-sections suivantes.

3.3.1 Etude de la gestion de la mémoire sous Linux

Un processus est une instance d'un programme en cours d'exécution. Lors de son lancement, le système d'exploitation lui réserve un espace d'adressage virtuel. Chaque processus dispose ainsi de son propre espace d'adressage, appelé "mémoire virtuelle". Chaque référence à la mémoire faite par un processus ne concerne pas directement la mémoire physique mais la mémoire virtuelle du processus. Cette dernière peut être plus grande que la mémoire physique, permettant l'exécution de programmes nécessitant beaucoup de mémoire. La mémoire virtuelle est découpée en plusieurs entités, appelées "pages" mémoire, de taille fixe. Généralement la taille des pages est de 4 Ko mais peut varier de 4 Ko à 16 Ko. Chaque page virtuelle est une suite d'adresses contiguës de la mémoire virtuelle. Une adresse virtuelle est un couple, formé d'un numéro de page ainsi que du déplacement à effectuer dans la page. La mémoire physique est également découpée en zones, appelée "cadres" de page. Ces cadres sont de même taille que les pages mémoire. Chaque cadre de page peut être vide ou contenir la copie d'une page mémoire d'un processus.

Lors de l'exécution d'un processus, seules les pages mémoire nécessaires à son exécution résident dans la mémoire physique. Les autres pages sont stockées sur le disque dur. Les adresses virtuelles sont traduites en adresses physiques grâce à l'unité de gestion mémoire, dite MMU (Memory Management Unit). Il s'agit d'un composant matériel, intégré au processeur. Lorsque la MMU reçoit une adresse virtuelle, cette

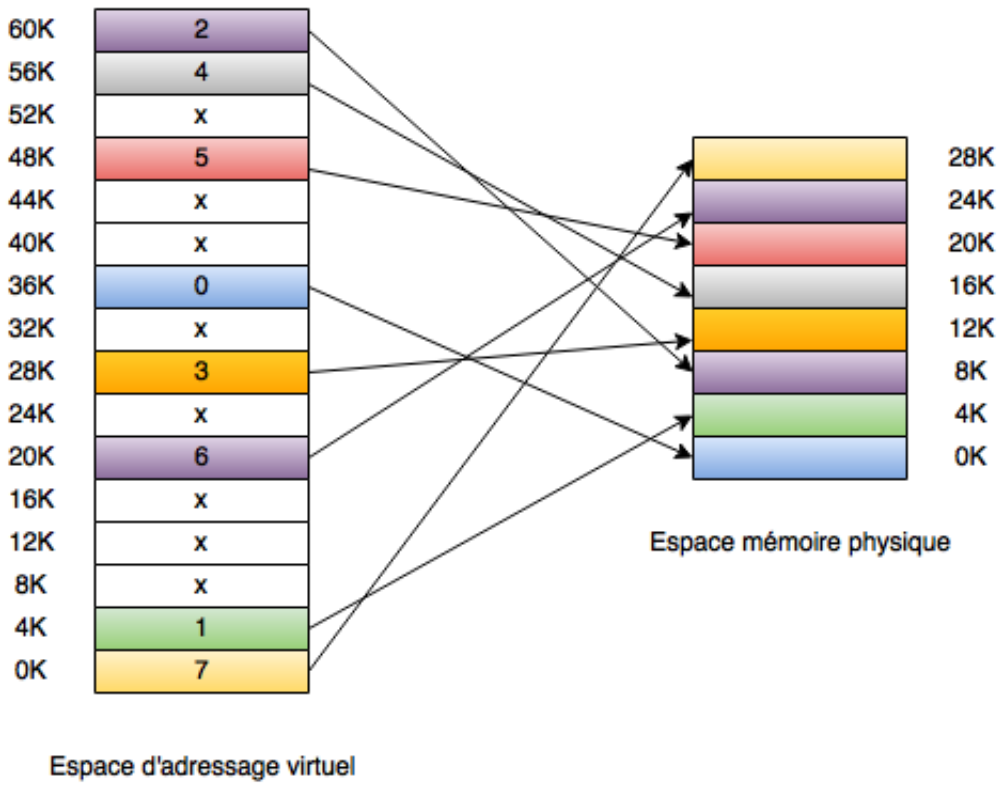


FIGURE 3.5 : Exemple d'adressage virtuel et physique

dernière examine la table des pages du processus afin d'en déterminer l'adresse physique. La table des pages du processus permet ainsi la correspondance entre pages virtuelles et pages physiques (cadres). La figure 3.5 permet d'illustrer les faits. Dans cet exemple, un programme dispose d'un adressage virtuelle de 64 Ko (16 pages de 4 Ko). La machine physique dispose de 32 Ko (8 cadres de 4 Ko) de mémoire vive. Le programme souhaite accèder à l'adresse virtuelle 28692 (7x4096 + 20). La MMU constate que cette adresse se situe dans la page mémoire 7 (page mémoire jaune, 28 Ko à 32 Ko) et la transformation donne la case 3 (cadre mémoire jaune, 12 Ko à 16 Ko) dans la mémoire physique. Ainsi à l'adresse virtuelle 28692 correspond l'adresse physique 12308 (4x4096 + 20). Si la page recherchée n'est pas disponible en mémoire mais sur le disque (sur l'exemple, une page mémoire notée d'un 'x'), la MMU déclenche une interruption dite "défaut de page". Le système d'exploitation doit alors libérer un cadre de page en le copiant sur le disque, le remplacer par la page mémoire correspondante et mettre à jour la table des pages du processus. La zone du disque permettant de stocker les pages mémoire ne résidant pas dans la mémoire physique se nomme "swap".

Un défaut de page est très "coûteux" (plusieurs centaines de milliers d'instructions)[HC92]. De ce fait, de nombreux travaux autour d'algorithmes de remplacements de pages mémoire [Den68] ont été réalisés dans l'objectif de réduire le nombre de défauts de page. L'algorithme utilisé dans les systèmes d'exploitation actuels est l'algorithme LRU (Least Recently Used)[OOW93].

Les compteurs mémoire

La taille de l'ensemble des cadres mémoire utilisés par un processus, c'est à dire l'ensemble des pages mémoire résidant dans la mémoire physique est appelée *Resident Set Size*. La taille de l'ensemble des pages mémoire utilisées par un processus est appelée *Virtual Set Size*. Ces compteurs sont disponibles à partir des commandes *top* ou *ps*. Ces dernières permettent de surveiller l'activité du système d'exploitation en obtenant la liste des processus exécutés ainsi que leur consommation processeur et mémoire. Cependant, ces commandes donnent un grand nombre d'informations diverses et variées. Nous ne rentrerons pas dans le détail de ces dernières dans cette thèse.

Compteur	Description
MemTotal	quantité de mémoire totale disponible pour le système d'exploitation (correspond à LowTotal + HighTotal)
MemFree	quantité de mémoire disponible (correspond à LowFree + HighFree)
MemAvailable	
Buffers	quantité de mémoire utilisée comme mémoires tampons (buffers)
Cached	quantité de mémoire utilisée en tant que cache (fichiers lus à partir du disque). Ne comprend pas SwapCached
Active	quantité de mémoire récemment utilisée (correspond à Active(anon) + Active(file))
Inactive	quantité de mémoire non utilisée récemment, pouvant être réutilisée par le système si besoin (correspond à Inactive(anon) + Inactive(file))
HighTotal	mémoire totale présente dans la zone ZONE_HIGHMEM
HighFree	mémoire disponible présente dans la zone ZONE_HIGHMEM
LowTotal	mémoire totale présente dans la zone zone ZONE_NORMAL
LowFree	mémoire disponible présente dans la zone ZONE_NORMAL
SwapTotal	Taille totale du swap (correspond à SwapFree + SwapCached)
SwapFree	Taille disponible du swap
SwapCached	quantité de mémoire lue depuis le swap mais toujours présente dans ce dernier
Dirty	quantité de mémoire en attente d'être écrite sur le disque
WriteBack	quantité de mémoire en cours d'écriture sur le disque
Committed_As	quantité de mémoire actuellement allouée. Il s'agit de somme de toute les mémoires allouées par les processus
CommitLimit	quantité de mémoire disponible pouvant être allouée.
AnonPages	quantité de mémoire allouée avec la fonction mmap() et l'argument MAP_ANONYMOUS, c'est à dire par projection anonyme
Active(anon)	quantité de mémoire "AnonPages" récemment utilisée
Inactive(anon)	quantité de mémoire "AnonPages" non utilisée récemment
Mapped	quantité de mémoire allouée avec la fonction mmap() par projection de fichiers (tels que les librairies) ou de périphériques
Active(file)	quantité de mémoire "Mapped" récemment utilisée
Inactive(file)	quantité de mémoire "Mapped" non utilisée récemment
PageTables	quantité de mémoire utilisée pour les tables de pages
AnonHugePages	Identique à AnonPages mais pour des HugePage
HugePages_Total	nombre de HugePage allouées. Une HugePage est une page de grande taille
HugePages_Free	nombre de HugePage disponibles
Hugepagesize	taille d'une HugePage
VmallocTotal	quantité de mémoire totale de l'espace d'adressage virtuel
VmallocUsed	quantité de mémoire utilisée dans l'espace d'adressage virtuel
VmallocChunk	taille du plus grand bloc contigu de mémoire virtuelle disponible dans l'espace d'adressage virtuel

TABLE 3.2 : Description des compteurs meminfo

En revanche, nous nous sommes focalisés sur les moyens de suivre précisément l'activité mémoire d'un système d'exploitation Linux. Pour celà, nous nous sommes concentrés plus particulièrement sur le répertoire */proc*.

Le répertoire */proc* permet d'obtenir des informations sur le système. Un répertoire est créé pour chaque processus s'exécutant sur le système, nommé par le PID (Process IDentifier) du processus et contenant les informations sur le processus en question. Le fichier */proc/meminfo* permet d'obtenir des informations détaillées sur l'utilisation de la mémoire. Ainsi, les métriques obtenues par la commande *cat /proc/meminfo*, pour un noyau 2.6.32-5 sont décrites dans le tableau 3.2. Les informations retournées par cette commande varient en fonction de l'architecture.

Nous avons discuté dans cette section de la gestion de la mémoire sous Linux ainsi que les procédés permettant de superviser finement l'activité mémoire du système d'exploitation ainsi que des processus. Cependant, il s'agit là d'introspection intrusive dans la mesure où la surveillance est réalisée directement depuis le système d'exploitation inspecté. Cette intrusivité a été utilisée dans nos travaux pour l'évaluation de notre extension à LibVMi 5.2. En effet une sonde intrusive a été implémentée dans l'objectif de mesurer les métriques obtenues par la commande *cat /proc/meminfo*. Les résultats obtenues ont été correlés à ceux obtenus depuis notre sonde.

3.3.2 Etude des structures de données pour la gestion des processus dans Linux

Comme toute application, le système d'exploitation dispose de données structurées pour le suivi et l'exécution des processus. Ces structures permettent de mémoriser des données statiques (nom du processus, pid etc.) et dynamiques (priorité, temps d'exécution etc.). Dans le cadre de notre contribution, nous nous sommes focalisés sur les structures internes du noyau Linux. Notre approche peut être adaptée à d'autres systèmes invités. Pour le noyau Linux, les principales structures utiles pour la collecte des métriques mémoires sont la structure task_struct et mm_struct. Ces structures sont présentées ci-après.

3.3.2.1 Structure task_struct

Dans le noyau Linux, chaque processus est représenté par une structure, nommée task_struct, dans laquelle il est possible de retrouver les informations relatives à chaque processus. Il s'agit du descripteur du processus. L'ensemble de ces descripteurs est contenu dans une table des processus, elle-même incluse dans l'espace mémoire du noyau. La structure task_struct est doublement chaînée, c'est-à-dire que chaque processus est lié au processus précédent (*struct task_struct *prev_task*) ainsi qu'au(x) processus suivant(s) (*struct task_struct *next_task*). La majorité des champs de la structure sont manipulables en utilisant le langage C (elle est déclarée dans *linux/sched.h*). Les champs de la structure task_struct qui nous intéressent dans notre contribution sont le *pid* correspondant à l'identifiant unique du processus, le champ *comm* correspondant au nom court du processus ainsi que le champ **mm* qui est un pointeur vers la structure mm_struct. La définition de la structure task_struct est dépendante de la version du noyau. Dans ces travaux, nous avons travaillé sur un noyau 2.6.32-5.

3.3.2.2 Structure mm_struct

Afin de mesurer l'empreinte mémoire des processus, nous nous sommes intéressés à la structure mm_struct. Il s'agit du descripteur mémoire d'un processus. La structure mm_struct sauvegarde toutes informations liées à l'espace d'adressage d'un processus. Elle comporte notamment le nombre de pages mémoires allouées au processus, résidant dans la mémoire physique (*unsigned long rss, Resident Set Size*), les adresses de début et de fin des segments mémoires relatifs à l'environnement d'exécution du processus (*unsigned long env_start, env_end*), ses arguments (*unsigned long arg_start ; unsigned long arg_end*), la liste triée des zones mémoires (VMA, Virtual Memory Area) appartenant au processus (*struct vm_area_struct *mmap*),

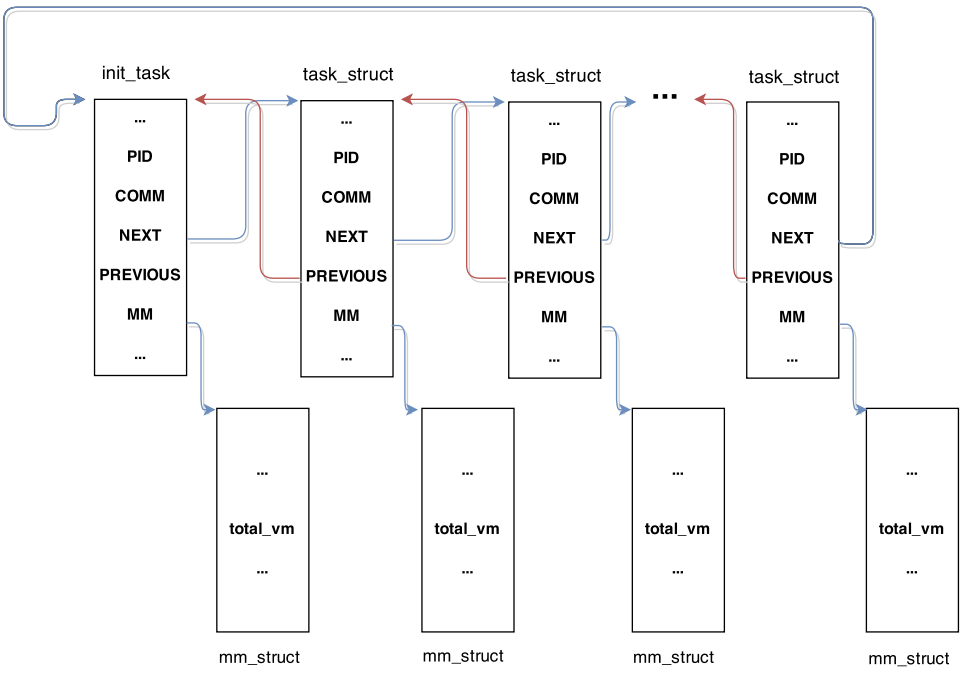


FIGURE 3.6 : Les structures task_struct et mm_struct

etc. Le compteur que nous utilisons dans notre contribution est le compteur *unsigned long total_vm*. Ce compteur permet de connaître la taille de la mémoire virtuelle du processus.

3.3.3 Une extension à LibVMI

Dans l'objectif d'analyser de manière non intrusive l'empreinte mémoire d'une machine virtuelle, nous avons développé une extension à la librairie LibVMi. LibVMi fournit une API permettant de parcourir les structures du noyau du système d'exploitation s'exécutant dans la machine virtuelle. Dans un premier temps, le travail consiste à parcourir l'ensemble des processus s'exécutant au sein de la machine virtuelle, donc présents au sein de la mémoire physique. Cette liste de processus s'obtient en parcourant la liste doublement chaînée *task_struct* (Figure 3.6). Pour cela, nous devons trouver l'adresse du point d'entrée, c'est à dire l'adresse de la structure *init_task*. Cette dernière correspond au premier élément de la liste des processus, du type *task_struct*. Son adresse est obtenue en recherchant le symbole "init_task" dans la table des symboles du fichier System.map. Pour rappel, ce fichier fait le lien entre une adresse mémoire et un symbole donné. La recherche d'adresse dans la table des symboles est réalisée par la méthode *addr_t vmi_translate_ksym2v(vmi_instance_t vmi, const char * symbol)*. En passant le symbole "init_task", cette méthode retourne l'adresse correspondante. Une fois cette adresse mémoire obtenue, nous pointons désormais sur la structure *task_struct*, correspondant au processus père, appelé "init". Dans le noyau Linux, les processus étant liés entre-eux, l'adresse du prochain processus est obtenue en pointant sur le champ *tasks* de chacune des structures *task_struct*. Le parcours de l'ensemble des processus se termine alors lorsque le pointeur se situe à nouveau sur la structure *task_struct* du processus "init". Pour pointer sur le champ *tasks*, il faut alors effectuer un déplacement dans la zone mémoire, entre l'adresse de début de la structure et l'adresse correspondant au champ *tasks*.

Ces déplacements ont été préalablement calculés à partir d'un module chargé dans le noyau des machines virtuelles à analyser. Ces déplacements sont propres à chaque machine virtuelle dans la mesure où l'organisation de la mémoire est dépendante du système d'exploitation et de la version du noyau. Ainsi, pour les besoins de cette extension, nous avons calculé les déplacements pour les champs *tasks*, *comm*, *pid*, *pgd* et *total_vm*.

Les déplacements obtenus pour le noyau 2.6.32-5 sont les suivants :

- comm = 0x224
- tasks = 0xf0
- mm = 0x10c
- pid = 0x12c
- total_vm = 0x60

En partant des informations précédentes, pour pointer sur le champ *tasks*, il faut faire un déplacement de '0xf0' depuis l'adresse d'entrée de la structure. A partir de cette adresse, nous pointons sur la structure suivante et pouvons récupérer les informations permettant d'identifier le processus comme son nom et son PID. Ces informations sont obtenues en effectuant à nouveau un déplacement dans la zone mémoire. En effet, le nom d'un processus est disponible via le champ *comm*, qui dans notre exemple, nécessite un déplacement de '0x224'. Ce déplacement correspond alors au décalage à effectuer entre l'adresse de la structure et l'adresse du champ *comm*. De la même manière, nous récupérons le PID du processus en effectuant un déplacement de '0x12c'. Une fois le processus identifié, nous réalisons un décalage vers la structure *mm_struct*. C'est au sein de cette structure que se trouve le champ *total_vm* correspondant à la taille de la mémoire virtuelle du processus. Pour chaque *task_struct*, nous effectuons un décalage permettant ainsi de récupérer l'adresse de la *mm_struct*. En pointant à cette adresse et en effectuant un nouveau décalage, nous sommes en mesure de récupérer la valeur du champ *total_vm* du processus.

Ainsi, en effectuant à nouveau un décalage au sein de la structure *task_struct*, nous récupérons l'adresse pointant vers le processus suivant et réalisons les mêmes opérations que précédemment pour les autres processus, jusqu'à pointer sur l'*init_task*. Pour chaque processus, nous récupérons alors son nom, son PID et la valeur du champ *total_vm*.

Une fois le parcours des structures mis en place, le second travail a consisté en l'historisation des données. Cette historisation permet de suivre l'activité de chacun des processus de la machine virtuelle dans l'objectif de détecter un processus malveillant, un processus dont l'activité est anormale. Ainsi, pour chaque processus, nous enregistrons la valeur obtenue par le champ *total_vm* de la structure *mm_struct* dans un fichier. Chaque processus dispose alors d'un fichier dans lequel sont stockées les dates de relevé et les mesures de consommations. Nous avons également développé un script R, permettant rapidement de créer les courbes de consommations pour chaque processus.

L'évaluation de cette contribution est réalisée dans le chapitre 5

3.4 Conclusion

Ce chapitre peut être vu à la fois comme un état de l'art technique, poussé et indispensable et une contribution sur l'environnement Linux . En effet, dans l'objectif d'analyser finement l'activité des machines virtuelles, une étude des compteurs de performances est nécessaire afin de comprendre leur signification et en déterminer la pertinence. Dans un premier temps, cette étude de compteurs de performances a été réalisée à partir de l'hyperviseur le plus utilisé dans les entreprises, l'hyperviseur ESX, développé par VMware. Cette dernière a ensuite été complétée par une étude de différents hyperviseurs, Hyper-V 2012 de Microsoft, XenServer de Citrix et l'hyperviseur open-source KVM. De part ce comparatif, nous avons observé l'indisponibilité de la majorité des compteurs de performances sous KVM. Cette observation s'est traduite par le développement d'un outil d'introspection non intrusive de machines virtuelles pour l'hyperviseur open-source KVM.

4

Analyses des ressources systèmes

Dans ce chapitre, nous discutons des différentes façons d'analyser les centres de données. Au niveau des machines virtuelles et des hyperviseurs, l'analyse avancée des compteurs de performances est primordiale dans la mesure où elle permet de détecter ou de prévenir d'éventuelles dégradations de performances. Au niveau du centre de données, l'analyse permet de détecter des machines virtuelles dont le comportement est différent des autres machines virtuelles s'exécutant au sein de ce centre de données. Nous les appelons machines virtuelles atypiques.

Sommaire

L'ANALYSE fine des ressources systèmes peut être utilisée dans plusieurs contextes tels que la sécurité, la tolérance aux pannes, les fuites mémoires ou encore l'optimisation de l'infrastructure et sa consommation énergétique. Dans ce chapitre, nous distinguons deux types d'analyses. La première consiste à analyser les compteurs de performances en vue de rechercher ou de détecter d'éventuelles dégradations de performances des machines virtuelles. Ces analyses reposent sur la surveillance de compteurs clés, étudiés dans le précédent chapitre de cette thèse. La seconde consiste à analyser les machines virtuelles au niveau "centre de données". Il ne s'agit plus d'étudier finement les métriques de performances mais d'analyser plus globalement le comportement des machines virtuelles. Ainsi, dans un premier temps, il s'agit de rechercher les machines virtuelles ayant des comportements pré-déterminés. Ce sont des machines dont l'activité répond à des critères identifiés. Ce type d'analyse permet par exemple, de connaître rapidement les machines

virtuelles "à risque", celles dont les consommations en ressources sont importantes. La seconde analyse recherche les machines virtuelles ayant des comportements atypiques. Il s'agit ici de rechercher les machines virtuelles ayant un profil d'activité différent des autres.

4.1 Analyse avancées des compteurs de performance des serveurs physiques et virtuels

Dans cette section, en s'appuyant sur les différentes études effectuées dans le chapitre 3, nous définissons diverses analyses critiques de type performance. Ces analyses concernent les principales ressources utilisées par les machines virtuelles : processeur, mémoire, disque et réseau.

4.1.1 Analyse d'anomalies liées au processeur

4.1.1.1 Détection d'une surcharge au sein de la machine virtuelle

La détection d'une forte activité processeur au sein de la machine virtuelle se traduit par l'étude des différentes valeurs des compteurs "COSTOP", "READY", "WAIT" et "RUN". Comme expliqué précédemment, ces compteurs représentent divers temps dont la somme représente 100% du temps d'exécution de la machine virtuelle. Nous obtenons alors l'équation 4.1.

$$100\% = \%RUN + \%READY + \%COSTOP + \%WAIT. \tag{4.1}$$

A partir de l'équation 4.1, pour détecter qu'une machine virtuelle dispose d'une forte activité CPU, il faut se focaliser uniquement sur les compteurs "RUN", "READY" et "COSTOP". N'est pas considéré le "WAIT", dans la mesure où ce dernier correspond au temps d'attente d'une ressource ou une attente de travail. De ce fait nous obtenons l'équation 4.2, prenant en compte uniquement les compteurs responsables de l'activité CPU.

$$\%WAIT \approx 0\% \equiv \%RUN + \%READY + \%COSTOP \approx 100\% \tag{4.2}$$

Ainsi, dans le cadre de détection de problèmes de performances, il est pertinent de comparer la somme des compteurs non pas à 100% mais plutôt par rapport à des seuils. En effet, à 100% les machines virtuelles sont déjà à saturation. Il faut anticiper d'éventuels problèmes. De plus, la définition de plusieurs seuils permet de prioriser les machines virtuelles à surveiller. Pour cela, nous avons défini des seuils par une analyse croisée entre les préconisations fournies par VMware et notre expérience chez Easyvirt. Ainsi, un premier seuil, placé à 90%, qui est également le seuil de détection de machines virtuelles dites "busy", permet d'alerter sur la forte charge processeur des machines virtuelles. Ce seuil permet de mettre en avant les machines virtuelles qui risquent d'arriver à saturation. Un second seuil, placé à 95% permet d'ajouter un niveau de risque supplémentaire puisqu'il met en avant les machines virtuelles arrivant à saturation. Il est donc crucial de se focaliser sur ces machines virtuelles avant que celles-ci ne se dégradent en performances. Une machine virtuelle, bien que correctement dimensionnée, peut être en surcharge processeur. En effet, pour cela, il faut vérifier que la machine virtuelle ne dispose pas d'une limitation CPU. Ce dernier, si présent, limite la ressource CPU à la machine virtuelle et est responsable de la pleine charge de cette dernière. S'il n'y a pas de limitation CPU, la machine virtuelle a donc besoin de ressources supplémentaires. Il faut lui ajouter un ou plusieurs vCPU. Ceci est possible tant que la machine virtuelle n'est pas configurée avec le nombre de VCPU maximum. Le nombre de vCPU qui peut être attribué à une machine virtuelle est le nombre de cœur logique que possède l'hyperviseur sur lequel elle est hébergée. De plus, il faut faire

attention à ne pas trop surcharger l'hyperviseur. La somme des vCPU de l'ensemble des machines virtuelles sur ce dernier ne doit pas dépasser un ratio compris entre 3 et 5 vCPU [Low13] par cœur logique. En effet, au dessus de ce ratio, l'ordonnanceur de l'hyperviseur sera moins performant. Il passera beaucoup de temps à ordonnancer les différents vCPU des machines virtuelles, impactant alors les performances de ces dernières.

4.1.1.2 Détection d'une surcharge vCPU au niveau de l'hyperviseur

Chacun des vCPU des machines virtuelles est affecté à un cœur logique, nommé pCPU, de l'hôte. Tout au long de l'exécution d'une machine virtuelle, un même vCPU ne correspond pas à un même pCPU. L'allocation maximale de vCPU à une machine virtuelle correspond au nombre de pCPU de l'hôte. Cependant, cela ne signifie pas que le nombre total de vCPU ne peut être supérieur au nombre de pCPU. Il s'agit alors de sur-provisionnement. Au niveau de l'hyperviseur la détection d'un sur-proviosionnement important se réalise en analysant les compteurs "READY" et "MLMTD". Pour rappel le compteur "READY" correspond au temps qu'un ou plusieurs vCPU ont attendu avant d'être ordonnancés sur un ou plusieurs cœurs logiques par l'hyperviseur afin de continuer leur execution. Le compteur "MLMTD" retourne le temps passé par la machine virtuelle ayant atteint la limite maximale du processeur. La valeur du compteur "MLMTD" est incluse dans le compteur "READY". De ce fait, afin d'évaluer réellement la valeur du compteur "READY", il faut retrancher la valeur du "MLMTD". Ainsi on obtient une nouvelle mesure, nommée READY_REEL 4.3.

$$READY\_REEL = \%READY - \%MLMTD \tag{4.3}$$

Comme nous l'avons vu dans la section 3.1.1.2, plusieurs facteurs peuvent être responsables d'un important "READY". Pour améliorer les performances de la machine virtuelle, comme dans le cas précédent, il faut vérifier que cette dernière ne dispose pas d'une limitation CPU. De plus il faut également vérifier que la machine virtuelle ne dispose pas d'une affinité CPU. Pour rappel, l'affinité CPU consiste à mapper un vCPU sur un pCPU. De ce fait, l'hyperviseur, bien qu'il dispose de pCPU disponibles, va attendre la disponibilité du pCPU associé au vCPU. Cette attente est responsable de l'augmentation du "READY". Si les machines virtuelles n'ont ni limitation ni affinité et qu'elles disposent d'un "READY" élevé, il s'agit alors d'un hyperviseur sur-provisionné en vCPU. Afin de détecter d'éventuels problèmes de performances, il est nécessaire de comparer la valeur de 4.3 à différents seuils. Ces seuils ont été définis en fonction de différentes recherches et de nos retours d'expériences. De ce fait nous avons déterminé deux seuils. Le premier, de 10%, représente un seuil critique. C'est à partir de ce niveau que le ressenti utilisateur est présent. De ce fait les performances de la machine virtuelle dépassant ce seuil sont fortement dégradées. Cependant pour anticiper ces éventuels problèmes de performances, nous avons défini un niveau à 5%, correspondant alors à la moitié du seuil critique. Pour résoudre ces problèmes de "READY", la solution est de réduire le nombre de vCPU total sur le serveur ou de migrer les machines virtuelles ayant des valeurs critiques.

4.1.1.3 Détection d'une surcharge vCPU au niveau de la machine virtuelle

Lors de la création de machine virtuelle, il est difficile d'attribuer le bon nombre de vCPU. La tendance est l'attribution d'un nombre de vCPU trop élevé par rapport aux réels besoins de la dite machine. L'administrateur pense que le nombre de vCPU à un lien avec de bonnes performances de la machine. Ceci est une fausse idée. En effet, configurer des machines virtuelles avec plus de 1 vCPU peut conduire à des problèmes de performances. La détection d'une surcharge de vCPU se réalise en étudiant le compteur "COSTOP". En effet, ce dernier indique le temps d'attente dû à la contention d'ordonnancement des vCPU. Plus une machine virtuelle dispose de vCPU, plus cette dernière attendra l'accès au processeur. En règle générale, lors de la création d'une machine virtuelle, il faut configurer le nombre de vCPU en fonction des capacités des applications s'exécutant au sein de celle-ci et des probables charges de travail qu'elles peuvent générer. Si

l'application est multi-threadée mais que ca charge est faible, le nombre de vCPU à attribuer devrait être de 1 vCPU. Cette configuration permet d'éviter de sur-provisionner l'hyperviseur en vCPU. A l'inverse, si l'application est multi-threadée et la charge de travail attendue est élevée, il est préférable de configurer la machine virtuelle avec 2 vCPU ou plus (en fonction de l'application). Pour que la configuration soit optimale, il est nécessaire de surveiller l'activité des machines virtuelles afin de détecter d'éventuels problèmes de contention ou non et redimensionner au mieux le nombre de vCPU en conséquence. Ainsi, nous détectons une surcharge vCPU au niveau de la machine virtuelle lorsque la valeur du compteur "COSTOP" est positive. Au dessus de 1%, il s'agit d'une simple alerte, permettant d'anticiper d'éventuels problèmes de performance liés à l'ordonnancement. Au dessus de 5%, nous considérons que la surcharge est effective et qu'il est nécessaire de redimensionner la machine virtuelle en diminuant le nombre de vCPU.

Comme constaté dans la précédente section, détecter et corriger les anomalies liées au processeur n'est pas forcément simple sur un système virtualisé. Cela dépend à la fois de la machine virtuelle analysée, de son activité, de la machine physique qui l'héberge et également des autres machines virtuelles. Cette analyse est encore plus délicate pour la mémoire.

4.1.2 Analyse d'anomalies liées à la mémoire

4.1.2.1 Etude fine de la mémoire active

Cette métrique n'a pas été présentée précédement (section 3.1.2) dans la mesure où elle ne peut être représentée sur un schéma. Elle demande une étude particulière. Cette dernière permet d'estimer la quantité de mémoire physique cliente qui est activement utilisée, basée sur le nombre de pages mémoires récemment accédées. Pour redimensionner une machine virtuelle en mémoire, une possibilité serait de redimensionner l'allocation mémoire de la machine virtuelle au niveau du compteur *Memory Consumed*. En effet, ce compteur estime convenablement la consommation mémoire d'une machine virtuelle (elle prend en compte le partage mémoire). Cependant, bien que l'allocation mémoire de la machine virtuelle pourrait être drastiquement réduite, cette préconisation serait inutile, dû aux techniques d'optimisation de la mémoire. C'est le cas du ballooning (section 3.1.2.4), qui permet de récupérer de la mémoire disponible sur une machine virtuelle pour la donner à une autre machine virtuelle. La ballooning part du constat que seul un certain pourcentage de la mémoire est activement utilisé : la *Memory Active*. Il s'agit de la mémoire indispensable au bon fonctionnement de la machine virtuelle. La mémoire inutilisée (idle), différence entre la *Memory Consumed* et la *Memory Active* ne servant à rien, pourrait alors être libérée. Ainsi, le bon critère pour le redimensionnement serait alors d'analyser le comportement du compteur *Memory Active* et, si l'activité de ce dernier est stable, redimensionner l'allocation mémoire de la machine virtuelle juste au dessus de la valeur de ce compteur.
A partir des différentes traces de centres de données fournies par Easyvirt, nous avons remarqué que le ratio entre *Memory Consumed* et *Memory Active* est de l'ordre de 10 à 15. Cela veut dire que la mesure du compteur *Memory Consumed* est 10 à 15 fois plus importante que la mesure du compteur *Memory Active*. Ceci s'expliquant entre autres par le fait que la valeur du *Memory Consumed* prend en compte les différents caches du système d'exploitation. Ces différents caches ont été utilisés lors du démarrage de la VM mais ne sont plus activement utilisés par le système hôte. Pour confirmer l'utilisation du compteur *Memory Active* pour le redimensionnement des machines virtuelles, nous avons réalisé divers tests dont un qui consistait à compacter une douzaine de machines virtuelles sur un hôte disposant de caractéristiques matérielles modestes. Les machines virtuelles disposaient d'un vCPU et d'une allocation mémoire de 1 Go avec pour système d'exploitation Windows Server 2012 R2. Ces machines virtuelles étaient exécutées sur une machine physique ayant les caractèristiques matérielles suivantes : Intel(R) Core(TM)2 Duo CPU T5750 @2.00GHz, 4 Go de RAM. Nous utilisons ensuite ces machines virtuelles pour l'utilisation de services de base, tel que le service SNMP (Simple Network Management Protocol). Il s'avère que toutes les machines virtuelles s'exécutaient sans soucis de performances. Chacune d'entres-elles répondant parfaitement aux requêtes SNMP. En revanche, il est évident que si plusieurs machines virtuelles devaient redémarrer, leur

phase de redémarrage ne se réaliserait pas sans dégradations de performances, voir échouerait.

4.1.2.2 Détection de sur-provisionnement mémoire au niveau de l'hyperviseur

Un hyperviseur est en sur-provisionnement mémoire lorsque la somme des mémoires allouées aux machines virtuelles hébergées en cours d'exécution sur ce dernier est supérieure à la capacité mémoire de l'hôte. Le sur-provisionnement mémoire est géré par l'hyperviseur grâce aux différentes techniques d'optimisations mémoire. De plus, dans la majorité des cas, les machines virtuelles ne consomment pas la totalité de la mémoire qui leur est allouée. Partant de ce principe, l'hyperviseur est donc capable de gérer une quantité de mémoire allouée supérieure à sa capacité réelle. Cependant, plusieurs études [BGTV13] et nos retours d'expériences montrent qu'à partir d'un certain ratio (équation 4.4), les performances des machines virtuelles se dégradent. Les techniques d'optimisations utilisées par l'hyperviseur ne sont plus suffisantes. Nous recommandons alors que le ratio de sur-provisionnement ne dépasse pas 160%.

$$SUR\_PROVISIONNEMNT = \frac{\sum ALLOCATION\_VM}{CAPACITE\_HOTE} \tag{4.4}$$

De plus, dans [BGTV13], Banerjee et al. démontrent que ces techniques ont un certain coût en terme de ressources et donc impactent les performances. L'analyse fine du sur-provisionnement se réalise donc en surveillant le fonctionnement de ces procédés. Le Transparent Page Sharing est une technique mise en place par défaut au sein de l'hyperviseur, en sur-provisionnement ou non, afin de partager au maximum les pages mémoire identiques. Il n'y a donc aucune supervision particulière à effectuer. En revanche, le ballooning n'est utilisé qu'en cas de sur-provisionnemement. Dès lors que l'hyperviseur réclame de la mémoire à une machine virtuelle, il est nécessaire de suivre ce comportement afin d'anticiper un sur-provisionnement mémoire trop important. Cela se traduit par le suivi de la quantité de mémoire réclamée par l'hôte (cible) ainsi que la mémoire "donnée" par la machine virtuelle. Si la cible est supérieure à zéro, cela signifie que l'hôte ne dispose plus de mémoire physique et demande à la machine virtuelle de lui libérer de la mémoire. Afin d'éviter à l'hyperviseur d'utiliser cette technique, nous recommandons de redimensionner les machines virtuelles (voir section 4.1.2.1). Les différents redimensionnements permettront de baisser le taux de sur-dimensionnement mémoire de l'hôte, voir même ne plus être en sur-dimensionnement. Limiter l'utilisation du ballooning permet également d'améliorer les performances. En effet, lorsque l'hôte réclame de la mémoire physique, la machine virtuelle utilise des ressources processeur et disque pour swapper localement des pages mémoires. A l'inverse, lorsque le système invité tente d'accéder à une page mémoire qui a été swappée, un événement de type "page-fault" est généré, obligeant alors de ré-allouer la page mémoire. Cet événement génère un activité processeur, caractérisée par une augmentation du compteur "WAIT" correspondant au temps d'attente du processeur, le temps que la page mémoire soit à nouveau accessible. Cet événement génère également une activité disque dû à la lecture de la page mémoire. Pour éviter de fortes dégradations de performances, il est donc important de suivre l'utilisation du ballooning. Pour ce faire, nous analysons les compteurs correspondant à la taille de la cible (3.1.2.4) ainsi que la mémoire libérée (3.1.2.4) de chacune des machines virtuelles. Si la machine virtuelle est en train de libérer de la mémoire pour l'hyperviseur et que cette dernière est supérieure à 10% de sa mémoire consommée (3.1.2.2), la machine virtuelle risque des dégradations de performances. L'hyperviseur est sur-dimensionné en mémoire. Il faut alors redimensionner les machines virtuelles de sorte à laisser la mémoire auparavant allouée et non utilisée, disponible à l'hyperviseur.

4.1.2.3 Détection de contention mémoire au niveau de l'hyperviseur

Un hyperviseur est en contention mémoire lorsque la somme des mémoires actives aux machines virtuelles en cours d'exécution, hébergées sur ce dernier est supérieure à la capacité mémoire de l'hôte. Ainsi, le ratio défini dans l'équation 4.5, est critique s'il est égal ou supérieur à 1. Dans ce cas, l'hyperviseur va utiliser

les techniques de compression mémoire et/ou de swapping. Ces deux techniques sont plus agressives que le ballooning dans la mesure où elles sont plus coûteuses en ressources.

$$CONTENTION = \frac{\sum ACTIVE\_VM}{CAPACITE\_HOTE} \tag{4.5}$$

La compression est une technique essentiellement coûteuse en processeur. En effet, ce dernier est sollicité à chaque tentative de compression d'une page mémoire. Il l'est également lors de la décompression. De même que dans le cas du ballooning, lorsqu'une page mémoire compressée est accédée par une machine virtuelle, l'hyperviseur doit décompresser la page et en allouer avant d'autoriser l'accès à la page décompressée. Un événement de type "page-fault" est généré. Le temps de décompression et de ré-allocation de la page se traduit par une augmentation du compteur "WAIT" au sein de la machine virtuelle. Le swapping est la technique d'optimisation utilisée en derniers recours. En effet, c'est une technique très coûteuse dans la mesure où il y a de nombreux échanges avec le disque, impactant alors le stockage et les IO. Lorsqu'une machine virtuelle accède à une page mémoire swappée, le temps d'accès à cette dernière sera plus long, ce qui se traduit également par une augmentation du compteur WAIT. Dans l'objectif de détecter une contention mémoire au niveau de l'hyperviseur, nous suivons les différents compteurs expliqués dans les sections 3.1.2.5 et 3.1.2.6. Dès qu'un des compteurs à une valeur supérieure à zéro, cela signifie que l'hyperviseur utilise la technique de la compression ou du swapping.

4.1.3 Analyse d'anomalies liées au disque

4.1.3.1 Détection de latence au niveau de la machine virtuelle

Les faibles performances disque des machines virtuelles sont généralement le résultat d'une latence importante. Il est donc primordial de surveiller finement les compteurs liés à la latence disque des machines virtuelles. Au niveau des machines virtuelles, le compteur à surveiller est le compteur nommé GAVG (3.1.3.3) (Guest Average Latency). Il correspond à la latence totale vue par le système d'exploitation invité. Afin de prévenir d'une éventuelle latence, nous avons mis en place deux seuils d'alerte. Le premier seuil est défini à 30 millisecondes, seuil à partir duquel la latence a un impact sur les performances de la machine virtuelle, pouvant résulter d'un ralentissement ressenti par l'utilisateur. Nous définissons alors ce seuil, comme un seuil critique puisqu'à ce niveau, la latence influe sur la bonne exécution de la machine virtuelle. Afin de prévenir de cette dégradation de performance et dans l'objectif de ne pas atteindre le seuil critique, nous avons défini un second seuil, défini à 20 millisecondes. Dans un environnement idéal, la latence devrait être de 0. Cependant, dans pratiquement tous les centres de données que nous avons supervisés, il arrive parfois que les machines virtuelles aient de la latence. Ainsi, définir une alerte à partir d'un seuil défini à 0, conduirait à détecter des faux-positifs. Nous avons ainsi fait le choix de définir le second seuil, non pas à zéro mais à 20 millisecondes.

4.1.3.2 Détection de commandes échouées au niveau de la machine virtuelle

Comme nous l'avons vu dans la section 3.1.3.2, chaque périphérique de stockage est capable de gérer un certain nombre d'opérations. De ce fait, pour savoir si les échanges entre la machine virtuelle et son disque se sont bien passés, il est nécessaire de surveiller si toutes les commandes se sont bien exécutées. De ce fait, nous mesurons le compteur permettant de récupérer le nombre de commandes échouées (3.1.3.2). A partir d'une commande échouée nous considérons qu'il y a un potentiel problème, non critique mais sur lequel il faut faire attention. Ainsi, nous avons défini un premier seuil à 0. Cependant, pour les mêmes raisons que dans la sous-section précédente, il se peut qu'une seule commande ait échoué, pouvant conduire à la détection de faux-positifs. Pour certains administrateurs, détecter toutes les machines virtuelles ayant eu au moins une commande échouée serait intéressant. Pour d'autres, cela pourrait éventuellement conduire à la détection d'un nombre trop important de machines virtuelles. De sorte à filtrer ces dernières, nous avons

défini un second seuil. Au dessus de 5 commandes échouées, nous considérons qu'il s'agit d'un problème critique. En effet, un nombre important de commandes échouées a pour conséquence une sollicitation trop fréquente de l'espace de stockage de la machine virtuelle, impactant alors fortement ses performances. En effet, la machine virtuelle enverra la commande tant que la commande échouera.

4.1.4 Analyse d'anomalies liées au réseau

4.1.4.1 Détection de paquets perdus / erronés au niveau de la machine virtuelle

De la même manière que pour le disque, nous surveillons les paquets perdus ou erronés, aussi bien en émission qu'en réception de la machine virtuelle. Détecter des paquets perdus ou erronés est primordial pour un administrateur dans la mesure où cette détection permet de suivre les performances réseau de la machine virtuelle mais également vérifier que cette dernière répond bien aux exigences (Qualité de Service) qui peuvent être potentiellement signées entre l'hébergeur et son client. Pour identifier les machines virtuelles ayant des paquets perdus et/ou erronés, nous avons défini les mêmes seuils que dans la sous-section 4.1.3.2.

4.1.4.2 Reconfiguration de la carte virtuelle de la machine virtuelle

Lors de la création d'une machine virtuelle, l'interface réseau par défaut est l'interface E1000. Il s'agit d'une version émulée de la carte réseau physique 82545EM développée par Intel, dont les pilotes sont disponibles par défaut au sein de très nombreux systèmes d'exploitation. Cependant, il ne s'agit pas d'une carte optimisée pour la virtualisation. Alors que la E1000 est une carte 1 Gb, la VMXNET3 est une carte 10 Gb, entièrement virtualisée. Il faut installer les outils supplémentaires pour que cette carte soit reconnue par les systèmes invités.

4.2 Analyse comportementale des centres de données

Dans la section précédente, nous avons analysé et défini des règles de suivi de gestion d'une machine virtuelle. Dans cette section, nous nous intérressons à l'analyse comportementale du centre de données, en réalisant une analyse sur l'ensemble des machines virtuelles de ce dernier. La première analyse consiste à repérer des comportements typiques, basés sur des règles classiques. La seconde analyse s'appuie sur le regroupement de machines virtuelles, en partant du principe qu'une machine virtuelle puisse ne pas avoir le même comportement que les autres.

4.2.1 Etat de l'Art

Un de nos objectifs de notre travail est de déterminer les machines virtuelles ayant un comportement pré-déterminé ou atypique hébergées dans les centres de données. Concernant la détection de comportements pré-déterminés, il s'agit d'analyser l'activité des machines virtuelles (principalement les ressources processeur et mémoire) et repérer plusieurs ensembles de machines virtuelles avec des comportements prédéfinis fixés. Concernant la détection de comportements atypiques, il s'agit d'identifier les machines virtuelles dont les consommations de ressources diffèrent des autres machines virtuelles. Nous recherchons donc des cas isolés.

Concernant la détection de comportements pré-déterminés, plusieurs solutions ont été proposées telles que VROPS de VMware, DCScope d'Easyvirt ou encore VMTurbo. Ces solutions proposent le même ensemble de comportements pré-déterminés. Cependant, ces solutions ont plusieurs inconvénients. Le premier inconvénient de ces solutions est qu'il est difficile voir impossible de modifier les paramètres de chacun des comportements. Par exemple dans VROPS, une machine virtuelle est définie comme 'idle' si sa consommation processeur ne dépasse jamais 10%. Définir un unique pourcentage, fixe, afin d'analyser les machines

virtuelles hétérogènes s'exécutant au sein d'un centre de données dans lequel différentes machines physiques cohabitent est dénué de sens. Le second inconvénient est le fait que les solutions précédemment citées n'autorisent pas la gestion du bruit dans l'analyse des données. Il s'agit d'un élément essentiel lorsqu'un outil doit analyser des millions de données, ce qui est généralement notre cas.

Concernant les comportements atypiques, aucune solution basée sur les comportements pré-déterminés ne peut être utilisée. Ainsi, dans nos travaux, un comportement est dit 'typique' s'il est partagé par plusieurs machines virtuelles. Notre objectif est donc de regrouper par similarité des ensembles de machines virtuelles puis de déterminer celles ne pouvant être regroupées avec les autres. Cette approche classique d'analyse de données s'appuie sur une technique bien connue : le *Clustering*. Le Clustering consiste à regrouper un ensemble de points, caractérisés par plusieurs dimensions, en partitions (ou clusters) de points similaires. La similarité est exprimée par l'utilisation d'une mesure de distance entre les points. Pour partitionner un groupe de points, deux méthodes de regroupement existent. La première consiste à associer un point à un unique groupe. La seconde, permet de donner à un point, un critère d'appartenance à un groupe. Dans cette deuxième approche, un point peut donc appartenir à un ou plusieurs groupes. Ainsi, un point peut par exemple appartenir à un groupe 'g1' avec un degré d'appartenance de 0.5 et appartenir à un groupe 'g2' avec un degré d'appartenance de 0.2. Si cette approche peut être intéressante dans certains contextes, notre objectif étant de déterminer les machines virtuelles ayant un comportement atypique, il nous est donc nécessaire qu'un point appartienne à un unique groupe. Nous avons alors étudié les méthodes permettant d'associer un point à un unique groupe. Ces méthodes de partitionnement peuvent être classées selon quatre classes, définies suivant la fonction d'appartenance : une mesure de distance, la densité des points, une grille ou une approche hiérarchique.

L'algorithme le plus usité, K-MEANS[M+67], propose de diviser un ensemble de points en k partitions afin d'obtenir une similarité satisfaisante pour l'ensemble des K partitions. Cette approche itérative cherche à déterminer K centroïdes. Un centroïde est un point de l'espace de point définissant le centre d'une partition. Le choix initial des K centroïde est aléatoire. L'algorithme place les points dans le groupe le plus proche (le centroïde le plus proche) puis recalcule le centre de gravité de chacun des groupes, redéfinissant ainsi la position des K centroïdes. L'algorithme itère en replaçant les points par rapport à ces nouveaux centroïdes et se termine lorsque aucun des points ne change de groupe. L'algorithme possède plusieurs défauts. Le premier étant qu'il est sensible au bruit. Le second qu'il ne trouve pas nécessairement la configuration optimale et le dernier, qui est le plus bloquant dans nos travaux, est qu'il est nécessaire de définir le nombre K de groupes.

Pour améliorer la résistance au bruit, une amélioration de K-MEANS a été proposée dans PAM [KR09]. Son principe consiste à utiliser les points à analyser comme centroïde. Le choix initial de ces K points (appelé médoïde) est également aléatoire. Le reste de l'algorithme est proche de celui de K-MEANS si ce n'est que le nouveau barycentre doit être un point des données. Cette recherche de nouveaux médoïdes reste très coûteuse. De ce fait, il est rarement utilisé pour des espaces de point à classifier de taille important. Cette amélioration nécessite toutefois de définir à l'avance le nombre de partition voulu (nombre K).

Les algorithmes basés sur la densité, tels que DBSCAN [EKSX96] cherchent à définir des partitions basées sur la densité des points. Un point est considéré voisin d'un autre point s'il est à une distance inférieure à une valeur fixée. Un point est considéré dense si le nombre de ses voisins dépasse un certain seuil. L'algorithme cherche à regrouper des points denses entre eux et ainsi fusionner des partions. Initialement, le nombre de partition est égale aux nombres de points denses dans l'espace de données. Deux partions peuvent fusionner si un point P de la première partition est connectable à un point Q de l'autre partition, i.e. qu'il existe une suite de points entre P et Q dont la distance entre eux est inférieure au seuil. Cette approche itérative construit donc les partitions par agglomération en vue d'obtenir les plus grandes partitions possibles. L'avantage de cette approche est qu'elle est relativement efficace en temps de calcul et ne demande pas de fixer le nombre de partitions à l'avance comme dans les approches précédentes. En revanche, le seuil de densité reste difficile à déterminer.

Les algorithmes basés sur une grille tels que BANG [SE97], sont similaires aux algorithmes basés sur la densité. Ils cherchent à diviser l'espace en cellules, formant ainsi une grille, puis de fusionner les cellules

Sociétés	DC01	DC02	DC03	DC04
# Machines physiques	10	8	7	8
# Machines virtuelles	123	122	83	151
Taille BD (en Mo)	9	25	18	61
# Entrées de consommation	1753743	4282260	2855559	9197709

Sociétés	DC05	DC06	DC07	DC08
# Machines physiques	3	8	16	53
# Machines virtuelles	309	88	605	842
Taille BD (en Mo)	107	17	82	590
# Entrées de consommation	14325798	3892219	11633702	50803305

TABLE 4.1 : Infrastructures de centres de données

proches entre-elles. Chaque cellule de niveau i est divisée en plusieurs cellules plus petites de niveau i+1. L'algorithme effectue le regroupement des cellules denses de manière hiérarchique, en partant de la cellule racine et en fusionnant successivement les cellules denses voisines. Les cellules dense fusionnent si la différence de densité ne dépasse pas un certain seuil. Comme pour les algorithmes basés sur la densité, le seuil est donc difficile à déterminer.

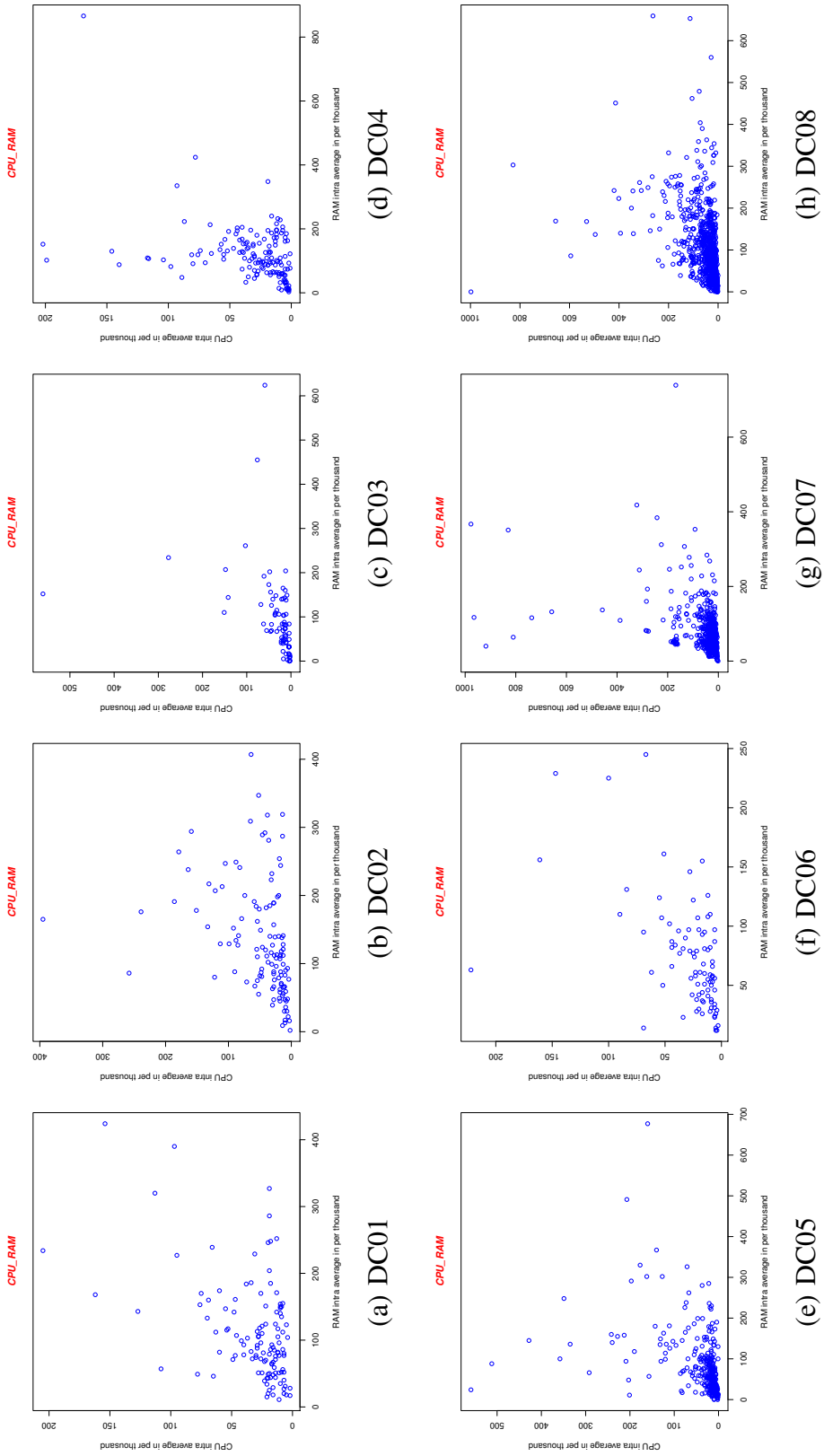
Les algorithmes hiérarchiques cherchent soit à regrouper des points jusqu'à obtenir des partions les plus denses (approche par agglomération)(e.g. AGNES (AGglomerative NESting)[KR09], soit à l'inverse, à diviser en partitions plus petites (approche par division)(e.g. DIANA (DIvide ANAlysis)[KR09] à partir d'un groupe regroupant l'ensemble des points. L'approche par agglomération cherche à réunir deux groupes les plus proches afin de les regrouper en un seul. Il itère jusqu'à ce que N (pouvant être égal à 1) groupes (défini à l'avance) soient trouvés. L'approche par division consiste à diviser les partitions selon un critère de dispersion des objets. L'algorithme itère jusqu'à ce que N groupes soient trouvés (N pouvant être le nombre de points initiale du problème). Pour ces deux algorithmes, la distance entre deux groupes peut être calculée comme : soit la distance minimale entre tous les points des deux groupes (single-link), soit la distance maximale entre tous les points des deux groupes (complete-link) soit la moyenne des distances entre tous les points des deux groupes (average-link). Le choix du seuil reste un élément difficile à paramétrer.

4.2.2 Contributions

Nos contribution peuvent être classées suivant trois sections : traces collectées, comportements pré-déterminés personnalisables, identification de machines virtuelles à comportement atypique.

4.2.2.1 Traces récoltées

Dans le cadre des activités de la société EasyVirt, j'ai été amené à réaliser de nombreux audits énergétiques. Ces audits ont consisté en la mise en place de sondes logicielles dans les centres de données des clients, puis à l'historisation des ces données dans une base de données MySql. Les sondes logicielles collectent les consommations de type ressource des serveurs physiques ou virtuels depuis la solution de virtualisation VMware®vCenter Server™. EasyVirt possède les traces d'une cinquantaine de clients. Parmi celles-ci, nous en avons sélectionné une dizaine, dont les informations sont représentées dans le tableau 4.1. Pour des raisons de confidentialité, les entreprises dans lesquelles les données ont été collectées ne peuvent être citées. De même, nous ne pouvons diffuser les données brutes colletées. En revanche, les résultats des analyses sont diffusables.



(a) DC01 (b) DC02 (c) DC03 (d) DC04

(e) DC05 (f) DC06 (g) DC07 (h) DC08

FIGURE 4.1 : Profils des centres de données

Notre prototype supervise l'activité des centres de données à partir d'une sonde logicielle, développée en JAVA, se connectant directement au Vcenter. A partir d'un compte en lecture seule uniquement, la solution récupère diverses informations. Dans un premier temps, la solution construit l'architecture Datacenter/Cluster/Server/machines virtuelles puis remonte les informations statiques des serveurs physiques. Ainsi, nous récupérons des données liées aux caractéristiques matérielles telles que le modèle du serveur, le modèle du processeur, sa fréquence, son nombre de cœurs ou encore sa quantité de mémoire. Ensuite, la solution récupère les informations de machines virtuelles liées à leur configuration tels que le nombre de vCPU, son allocation mémoire, sa quantité de mémoire réservées, l'état des VMware tools et la taille du disque virtuel.

Suite à la récupération de ces informations, la solution supervise ensuite les serveurs physiques et les machines virtuelles de sorte à suivre leurs consommations ainsi que le cycle de vie des machines virtuelles. Diverses ressources sont ainsi monitorées : processeur, mémoire, réseau et disque. L'ensemble des compteurs de performances remontées par DCScope sont principalement expliqués dans le chapitre 3.

La récupération des informations se réalise toutes les minutes, sans impacter les performances du VMware® VCenter. L'évaluation de l'impact de notre sonde sur l'activité de la machine hébergeant le VCenter est réalisée dans la section 5.1. Les données récupérées par cette sonde logicielle sont stockées dans une base de données MySQL. Ne sont pas stockées dans une seule table, toutes les données de consommations pour des raisons de performances. Nous avons fait ce choix dans la mesure où les temps de réponses sont dépendants du nombre d'entrées dans la table contenant les informations. En effet, en effectuant une requête 'select' dans la table, MySQL parcourt toute la table afin de vérifier si chaque entrée de la dite table répond aux conditions (clause 'where'). Plus la taille de la table est importante, plus le temps de réponse de la requête est long. Le tableau 4.1 présente la taille de la base de données ainsi que le nombre d'enregistrements de données de consommations pour les huit centres de données sélectionnées.

Dans l'objectif de maîtriser l'espace disque utilisée par notre base de données, nous avons développé différents scripts qui permettent automatiquement et périodiquement d'analyser, de consolider et de purger les données récoltées. Ainsi, les informations stockées dans chaque table de consommation sont consolidées dans une nouvelle table. Les consommations pour chaque élément (serveurs physiques et machines virtuelles) sont ainsi résumées en une seule entrée dans la nouvelle table. Ceci permettant une éventuelle purge de la base de données en supprimant les tables de consommations déjà résumées de sorte à libérer de l'espace disque.

Dans cette section, nous nous sommes focalisés sur les consommations CPU et mémoire. En fait, ces deux critères sont les plus pertinents pour notre étude et sont les ressources de contention dans un centre de données. Les figures 4.1(a) à 4.1(h) présentent une représentation graphique du comportement d'un ensemble de machines virtuelles. Sur ces graphiques, nous prenons pour point, la moyenne CPU et mémoire de chaque machines virtuelles sur la période d'analyse. En abscisse le taux d'utilisation CPU et en ordonnée le taux d'utilisation mémoire.

Les graphiques sont petits, mais nous pouvons observer une similitude d'activités entre les profils des différents centres de données. En effet, nous remarquons la présence d'un très grand ensemble de machines virtuelles près de l'origine. Ces ensembles sont représentés par des masses denses. Autour de ces masses denses, plusieurs machines virtuelles ont une activité complètement différente. Il s'agit de machines virtuelles ayant un comportement atypique. La détection de comportement atypique est présentée dans la section 4.2.2.3.

4.2.2.2 Comportements pré-déterminés

En vue d'aider l'administrateur à détecter des comportements et ou des configurations anormales, la solution DCScope développée par Easyvirt, est capable d'analyser sur une période de temps donné, les machines virtuelles ayant un comportement pré-déterminé. Il y a en tout six comportements :

- *idle* : Si sur une période de temps donné, 100% des valeurs de consommations processeur et mémoire sont inférieurs à Y%. Le taux d'activité (Y) étant directement tributaire de la capacité processeur et

mémoire du serveur sur lequel les machines virtuelles sont hébergées.

- *lazy* : Si sur une période de temps donné, une ressource R (processeur ou mémoire) a des pics de consommations supérieurs à 30% pendant moins de 10% du temps et est considérée comme 'idle' le reste du temps.

- *undersized* : Si sur une période de temps donné, 100% des valeurs de consommations processeur et mémoire sont supérieurs à 70%.

- *busy* : la définition d'une machine virtuelle 'busy' est identique à celle d'une machine virtuelle 'undersized', si ce n'est que le taux d'activité est supérieur à 90%.

- *oversized* : Si sur une période de temps donné, 100% des valeurs de consommations processeur et mémoire sont inférieurs à 30%.

- *ghost* (off) : Si à la fin de la période d'analyse, la machine virtuelle est éteinte.

Cependant, afin d'obtenir une détection plus robuste que des outils comme VROPS de VMware, nous avons modifié les comportements pré-déterminés en ajoutant une notion de bruit. En effet, dans les définitions précédentes, la détection des machines virtuelles est stricte. 100% de points doivent être en dessous ou au dessus d'un certain seuil, en fonction du comportement. Mais, une machine virtuelle dispose toujours d'une activité (pics d'activité) même si cette dernière n'est plus utilisée. Cette activité résulte de l'activité du système d'exploitation (vérification, installation des mises à jour) ou de l'activité temporaire d'une application (recherche anti-virus par exemple). De ce fait, nous avons modifié les définitions des comportements pré-déterminés afin de prendre en charge cette activité particulière, que nous appelons bruits. Ainsi, pour chaque comportement, nous faisons varier le pourcentage de points suivant quatre seuils : 100%, 99.99%, 99.9% et 99%. Ce facteur de 10 est lié à des seuils classiques de centres de données de hautes disponibilités.

Nous détaillons dans le chapitre 5 l'interêt de ces choix.

4.2.2.3 Comportements atypiques

L'état de l'art nous a amené à développer notre propre algorithme de partitionnement multi-critères et multi-ressources répondant à nos besoins, à savoir, rechercher des machines virtuelles ayant un comportement atypique, insensible aux bruits. Notre algorithme, basé sur une mesure de distance, est multi-ressources (CPU, RAM, E/S disque et E/S réseau). Il est également multi-critères dans la mesure où chaque ressource analysée peut elle-même être divisée en critères. Ici, nous définissons un critère comme étant une donnée statistique. Les données statistiques disponibles étant le premier quartile (25% des points sont en dessous de ce seuil), la médiane (autant de points en dessous qu'au dessus de ce seuil), la moyenne, le troisième quartile (75% des points sont en dessous de ce seuil), la valeur minimale et la valeur maximale, pour chacune des ressources. Ainsi, il est possible de chercher à regrouper les machines virtuelles en fonction d'un ou plusieurs critères.

La distance utilisée dans notre algorithme est calculée suivant un "taux de ressemblance", paramétrable, permettant de définir les bornes minimales et maximales des intervalles des valeurs statistiques. Ainsi, deux machines virtuelles ont une forte similarité si pour un ou plusieurs critères donnés, elles appartiennent aux mêmes intervalles. L'approche de la recherche du meilleur groupe est détaillée dans l'algorithme 17. Notre algorithme recherche récursivement le groupe comportant le plus de machines virtuelles en fonction du taux de ressemblance et des critères choisis et ce jusqu'à ce que toutes les machines virtuelles appartiennent à un groupe. Une machine virtuelle a un comportement atypique si cette dernière est seule dans son groupe ou si le nombre de machines virtuelles dans le groupe est inférieur ou égale à une borne arbitraire.

Algorithme 1 : recherche du meilleur groupe

Input : listVM : une liste de machines virtuelles, listCriterias : une liste de critères, VMgroupe : le groupe de machines virtuelles courant

Result : le groupe de machines virtuelles ayant le maximum de machines virtuelles pour les critères choisis

```
1  c1 = on prend le premier critère de listCriterias;
2  bestgroup = un nouveau groupe de machines virtuelles vide;
3  while valeur maximale de l'intervalle n'est pas atteinte do
4      tmpgroup = on cherche un groupe de machines virtuelles ayant une similarité proche pour le
       critère c1 et le groupe VMgroupe –> appel à la méthode getVMS;
5      if tmpgroup.size > VMgroupe.size then
6          if listCriterias n'est pas vide then
7              tmpgroup = searchGroup(listVM, listCriteria, tmpgroup);
8              if tmpgroup.size > bestgroup.size then
9                  bestgroup = tmpgroup;
10             end
11         else
12             bestgroup = tmpgroup;
13         end
14     end
15     on incrémente les intervalles;
16 end
17 return bestGroup ;
```

4.3 Conclusion

Dans ce chapitre nous nous sommes focalisés sur l'analyse des serveurs physiques et virtuels.

Dans un premier temps, nous avons mis en application nos différentes études sur les métriques de performances afin de proposer des analyses fines de l'activité des serveurs et des machines virtuelles sur les quatres ressources, processeur, mémoire , disque et réseau. Ces analyses permettent de rechercher ou de détecter d'éventuelles dégradations de performances des machines virtuelles. Il est important pour un administrateur qu'aucune machine de son infrastructure soit "à risque", aussi bien pour l'utilisateur de cette machine mais également dans l'objectif d'éviter que cette dernière impacte l'activité du serveur physique sur lequel elle est hébergée et indirectement, impacter les autres machines virtuelles.

Dans un second temps, nous avons discuté de l'analyse des comportements pré-déterminés et atypiques permettant à l'administrateur d'identifier des machines virtuelles sans activités (comportements *idle,lazy*), qui peuvent potentiellement être arrêtées. En arrêtant voir supprimant ces dernières, l'administrateur peut de ce fait libérer des ressources au sein du centre de données et donc éviter le rachat de nouveaux équipements. Cette analyse lui permet également d'identifier les machines virtuelles dont les capacités des ressources doivent être augmentées (*buzy, undersized*) ou au contraire diminuées (*oversized*). En redimensionnant les machines virtuelles, les gains en ressources peuvent être importants. En effet, le redimensionnement peut se faire sur 2 ressources : la ressource processeur, via l'affectation de VPCU (virtual CPU) et la ressource mémoire.

5

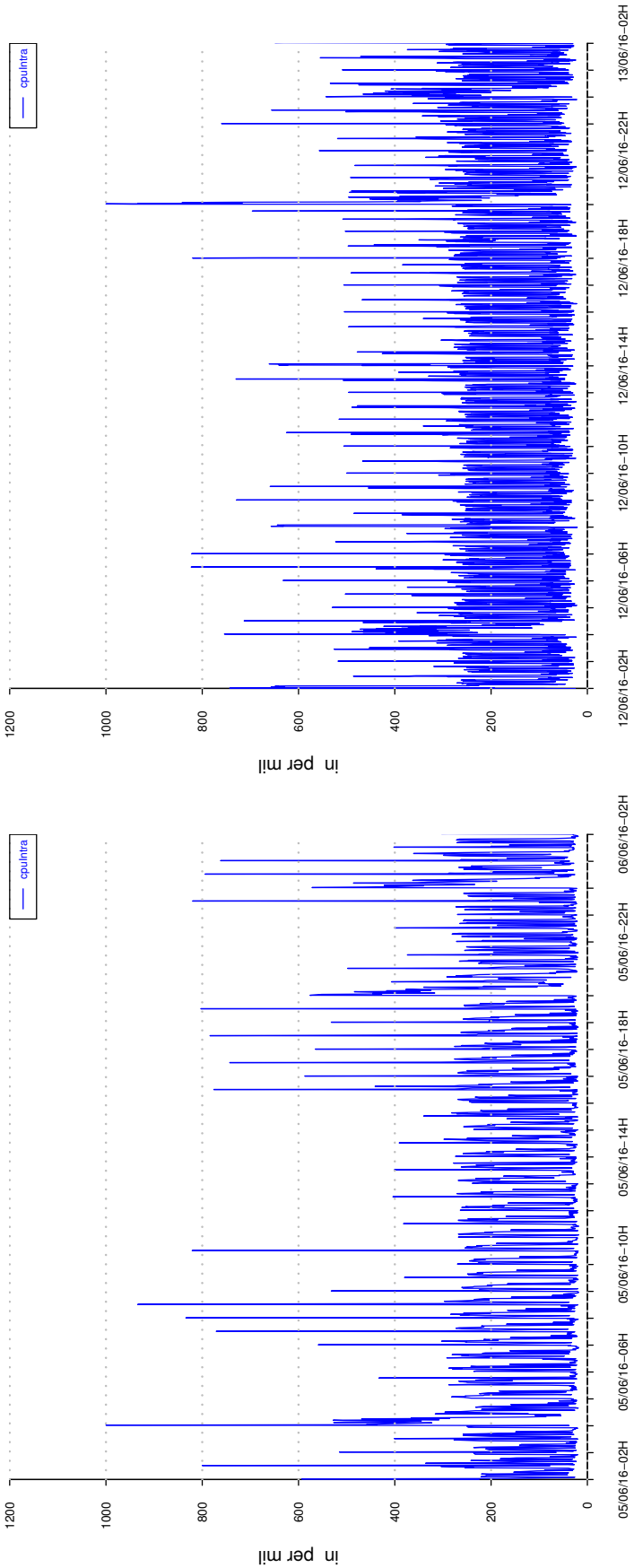
Expérimentations

Dans ce chapitre nous évaluons les différentes contributions apportées dans cette thèse. La première partie de ce chapitre concerne l'évaluation de notre sonde VMware. Nous mesurons l'impact de notre sonde logicielle et abordons le choix de la durée de l'échantillonage permettant un bon compromis entre impact de notre sonde logicielle et pertinence des analyses. Dans la deuxième partie de ce chapitre, nous réalisons l'évaluation de notre outil d'introspection de machines virtuelles sous KVM. Nous y montrons la capacité à récolter des métriques, plus finement que celles qui peuvent être récupérées par défaut. La troisième partie concerne l'évaluation de l'analyse comportementale des machines virtuelles.

Sommaire

5.1 Evaluation de la sonde VMware

La sonde VMware est une sonde logicielle, développée en JAVA, qui se connecte directement sur le VMware® VCenter. Cette sonde, à partir de laquelle nous récupérons les métriques de performances, doit également faire l'objet d'une évaluation. En effet, comme tout code logiciel, notre sonde a besoin d'être évaluée afin de surveiller son comportement et vérifier qu'elle n'impacte pas les bonnes performances du VCenter. Ainsi dans la première partie de cette section, nous évaluons l'impact de notre sonde logicielle sur l'activité processeur du système d'exploitation. Dans la seconde partie, nous présentons nos résultats sur l'impact de la durée de l'échantillonage sur les analyses comportementales des machines virtuelles.



(a) Echantillonage à 60 secondes

(b) Echantillonage à 20 secondes

FIGURE 5.1 : Impact de la sonde VMware sur l'activité processeur

Consommation CPU	<100	<200	<300	<400	<500	<600	<700	<800	<900	<1000
Echantillonage à 20s	47,4	8,9	34,4	5,2	1,8	0,7	0,9	0,4	0,2	0,1
Echantillonage à 60s	55,2	12,1	26,3	2,3	1,8	1,1	0	0,6	0,4	0,2

TABLE 5.1 : Répartition des données de consommation processeur

5.1.1 Impact de la sonde sur le VMware® VCenter

Dans l'objectif de mesurer l'impact de notre sonde sur le VCenter, nous avons mesuré l'activité processeur du système d'exploitation sur lequel il s'exécute, en deux temps. Le VCenter est exécuté au sein d'une machine virtuelle disposant de 2 vCPU et de 8 Go de mémoire et un environnement SUSE Entreprise 11 64 bits. Ce dernier gère 10 ESX et 300 machines virtuelles. Dans un premier temps, nous avons déployé la sonde VMware avec une récupération des métriques effectuée toutes les 60 secondes. L'activité processeur du système d'exploitation pendant cette phase est représentée sur la figure 5.1(a). Dans un second temps, nous avons modifié l'échantillonage de la sonde à 20 secondes. L'activité correspondante est représentée sur la figure 5.1(b). Nous n'avons pas réduit l'échantillonage en dessous des 20 secondes dans la mesure où il s'agit de l'intervalle de temps utilisé par le VCenter pour mettre à jour les métriques de performances des machines virtuelles. Nous avons choisi de représenter l'activité processeur sur une journée, un dimanche, de sorte à mieux distinguer les pics d'activités liés au changement de l'échantillonage. En effet, le dimanche, l'activité de l'infrastructure virtualisée est réduite et généralement liée à des tâches automatisées. Ainsi nous écartons, tout "bruit" qui peut être lié à une activité particulière. Sur les courbes 5.1(a) et 5.1(b), nous observons trois périodes de fortes activités, générées par des tâches automatisées (entre 04H et 05H, entre 20H et 21H et entre 00H et 01H). Pour étudier l'impact de notre sonde, nous nous focalisons sur les pics d'activités en dehors de ces périodes de charges.

Ainsi, nous constatons une similarité entre les profils, dont l'activité processeur varie de 50 à 250 pour mille. Cette variation semble plus régulière sur la figure 5.1(b) dans la mesure où il y a 2,5 fois plus de points que sur la figure 5.1(a) (et non 3 fois plus dû aux autres actions effectuées après récupération des données). Nous distinguons également des pics d'activité processeur au dessus de 400 pour mille, plus nombreux pour un échantillonage à 20 secondes que pour un échantillonage à 60 secondes. De ce fait, nous pouvons penser que la réduction de l'échantillonnage impacte les performances du système d'exploitation exécutant le VCenter.

Cependant, pour approfondir cette étude, nous avons travaillé sur la répartition des points et notamment calculé le pourcentage de points par intervalle de 100 pour mille. Les résultats obtenus sont décrits dans le tableau 5.1. Graphiquement, nous pensions que le nombre de points au dessus de 400 pour mille était plus important pour un échantillonage à 20 secondes. Or dans les deux cas, il y a 4,1% de points au dessus de 400. C'est en se focalisant sur les 95,9% restants que nous observons l'impact de la réduction de l'échantillonage. En effet, pour une durée de 20 secondes, il n'y a que 56,3% de points en dessous de 200 pour mille contre 67,3% pour une durée de 60 secondes. Cette différence de 11% se retrouve dans les pourcentages de points compris entre 200 pour mille et 400 pour mille. L'impact de la sonde sur le VCenter est alors une légère augmentation de la consommation processeur du système d'exploitation.

L'impact étant une augmentation de la consommation processeur, elle est donc dépendante des ressources matérielles sur lequel le VCenter est installé. Dans cette évaluation, l'impact est une augmentation de 10% de points dont la consommation est comprise entre 20% et 40%. Sur un système plus modeste, l'impact pourrait être plus important et aurait pour conséquence de dégrader les performances du VCenter. Il est alors essentiel de définir un échantillonage juste, permettant de mesurer une infrastructure virtualisée sans dégrader les performances du VCenter, tout en garantissant la pertinence des mesures dans l'objectif d'analyser finement les seveurs physiques et les machines virtuelles. C'est pourquoi, nous avons décidé de récupérer les métriques de performances toutes les 60 secondes. Dans la section suivante, nous nous

Comportements	Total	Run	Idle	Lazy	Oversized	Undersized	Off	Busy
Précision à 100%	279	263	34	28	230	1	16	1
Précision à 50%	279	263	34	28	230	2	16	1
Précision à 33%	279	263	34	28	230	2	16	1

TABLE 5.2 : Analyses comportementales de machines virtuelles en fonction de la précision

focalisons sur l'impact de la durée de l'échantillonage sur les analyses comportementales des machines virtuelles.

5.1.2 Impact de durée de l'échantillonage sur l'analyse comportementale

Dans la section précédente, nous avons évalué l'impact de l'échantillonage sur l'activité processeur du système d'exploitation exécutant le VCenter. Nous avons montré que la réduction de ce dernier avait un léger impact. Dans cette section, nous avons ciblé notre étude sur l'analyse comportementale de machines virtuelles suivant 3 précisions de mesures des données, 100%, 50% et 33%. Pour cela, nous avons travaillé sur une base de données dont l'échantillonage est de 20 secondes. A partir de cette base de données, nous l'avons modifiée de sorte à supprimer des entrées dans les tables stockant les métriques de performances des machines virtuelles. Le tableau 5.2 présente les résultats ainsi obtenus. La première ligne du tableau présente les résultats pour une précision des données à 100%, ce qui signifie sur toutes les données de la base de données, soit trois mesures par minute. La seconde ligne présente les résultats pour une précision à 50% (deux mesures par minute) et la dernière ligne les résulats pour une précision à 33% (une mesure par minute).

Nous remarquons que quelque soit la précision, nous obtenons le même nombre de machines virtuelles pour chacun des comportements sauf pour le nombre de machines virtuelles dites "Undersized". Pour rappel, une machine virtuelle est "undersized" si sur une période de temps donnée, 100% des valeurs de consommations processeur et mémoire sont supérieures à 70%. Nous avons dans cette évaluation une différence d'une machine virtuelle. Cette dernière est "undersized" à 99%. Elle ne dispose pas de ce comportement pour une précision de mesure à 100% étant donne le fait qu'au moins 1% de ses points sont inférieurs à 70%. En revanche, pour une précision de 50% et 30%, la machine virtuelle est détectée. Cela siginifie que la machine virtuelle dispose de pics d'activité au dessus de 70% très régulièrement.

Réduire la durée de l'échantillonage n'influe donc pas sur l'analyse fine des machines virtuelles. En effet, les résultats ont montré que les comportements des machines virtuelles sont identiques quelque soit la précision. Le choix de récupérer les métriques de performances toutes les 60 secondes est donc un très bon compromis entre l'impact de notre sonde sur le VMware® VCenter et la précision de l'analyse comportementale.

Dans la section suivante, nous nous intérressons à l'évaluation de la sonde Linux.

5.2 Evaluation de la sonde Linux

Dans le but de valider la pertinence de la valeur du compteur *total_vm* remontée depuis LibVMi, nous avons observé, suivant plusieurs méthodes, la consommation mémoire d'un processus. La charge théorique, c'est-à-dire la charge mémoire faite par un outil réalisant basiquement des allocations (malloc) et libérations (free) mémoire (nous appellerons par la suite cet outil 'mallocme'). Ce générateur d'activités est utilisé afin de simuler une activité mémoire au sein de la machine virtuelle sur laquelle il est exécuté. Il permet de

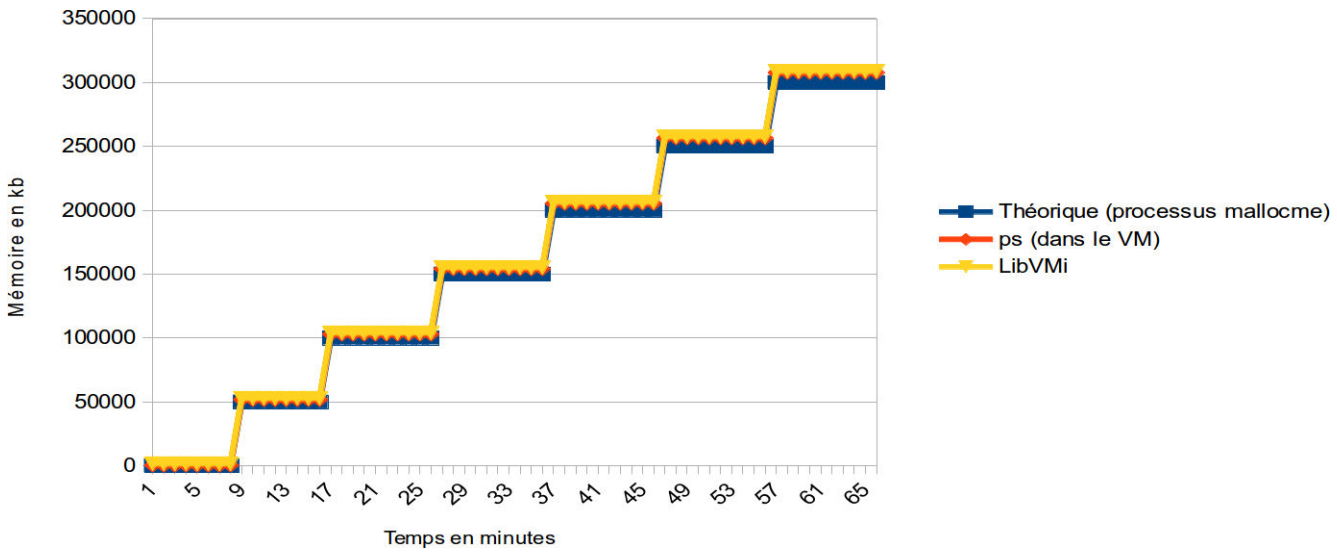


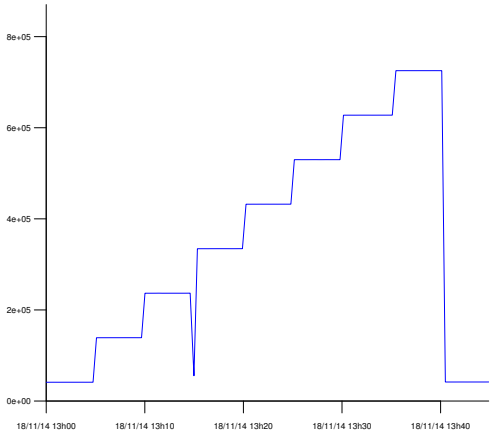
FIGURE 5.2 : Consommations mémoire (en ko)

scénariser des charges de consommations, en y détaillant les différentes charges souhaitées ainsi que leur temps de présence. Un exemple de scénario peut être une charge de 100 Mo pendant 5 minutes puis une charge de 300 Mo pendant 60 secondes. La consommation mémoire du processus est mesurée à la fois à partir du système d'exploitation de la machine virtuelle à l'aide de la commande ps et à la fois mesurée depuis l'hyperviseur via notre sonde logicielle. Pour cela, nous avons effectué nos expérimentations sur une infrastructure KVM. L'hyperviseur dispose d'une version Ubuntu 14.04 LTS 64 bits avec le noyau 3.13.0.14, d'une version de Libvirt 1.2.2 et de QEMU emulator 2.0.0. Les machines virtuelles déployées disposent d'un environnement Debian 6.0.7 32 bits avec un noyau 2.6.32-5, 1 VCPU et 1 Go de mémoire allouée.

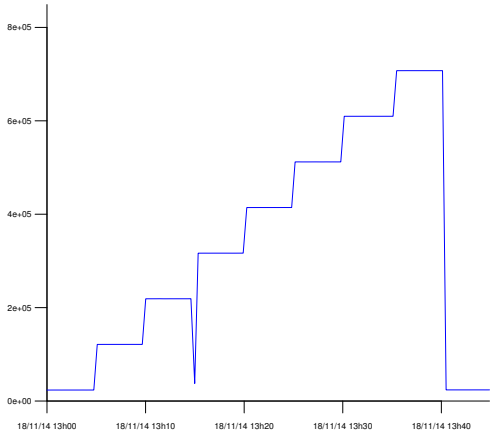
Les différentes courbes de la figure 5.2 montrent la même charge mémoire mesurée à l'intérieur de la machine virtuelle et depuis l'hyperviseur. En bleu, la charge théorique réalisée par le processus mallocme. En rouge, la consommation mémoire du processus mallocme, observée depuis la machine virtuelle, à l'aide de la commande ps. La consommation mémoire mesurée par cette commande correspond au VSZ (Virtual Set Size), représentant la taille totale de la mémoire virtuelle utilisée par le processus. En jaune, la consommation mémoire du processus mallocme, calculée à partir du compteur total_vm mesurée depuis LibVMi. Comme le champ total_vm mesuré depuis LibVMi retourne le nombre de pages mémoire utilisées par le processus, il suffit de multiplier cette valeur par la taille d'une page mémoire (4 Ko dans notre cas). Sur la figure 5.2, nous observons que les trois courbes se superposent parfaitement. Les courbes mettent en évidence la corrélation entre les mesures effectuées à partir de l'hyperviseur via notre sonde logicielle et celles effectuées à l'intérieur de la machine virtuelle via ps.

Dans l'objectif d'approfondir nos travaux autour de l'introspection mémoire, nous avons également corrélé les consommations mémoires mesurées depuis notre sonde, avec les compteurs mémoires obtenus via la commande cat /proc/meminfo. Les informations retournées par cette commande varient en fonction de l'architecture. Pour connaître les compteurs pertinents, nous avons créé un script, meminfo.sh, qui à un intervalle de temps paramétrable sauvegarde la valeur de chaque compteur. De ce fait, afin de déterminer quels sont les compteurs représentatifs, nous avons exécuté le script mallocme ainsi que le script meminfo.sh dans la machine virtuelle pendant plusieurs minutes. Grâce aux courbes, nous pouvons rapidement observer les compteurs évoluant en fonction de l'activité mémoire générée dans la machine virtuelle. Les compteurs représentatifs de l'activité mémoire d'un processus sont les compteurs *Active_anon* (quantité de mémoire récemment utilisée, non associée au système de fichiers), *Active* (la quantité de mémoire récemment utilisée), *AnonPages* (quantité de mémoire allouée par la fonction mmap()), *Commited_AS* (quantité de mémoire maximale utilisable par le système) et *PageTables* (quantité de mémoire dédiée aux tables de pages).

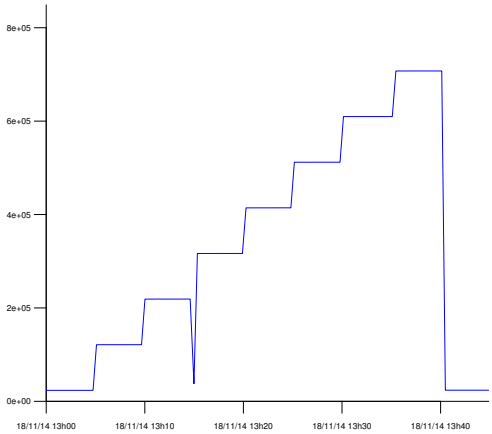
Les figures 5.3(a) à 5.3(e) permettent d'observer l'évolution de la valeur mesurée (en kilobytes) pour



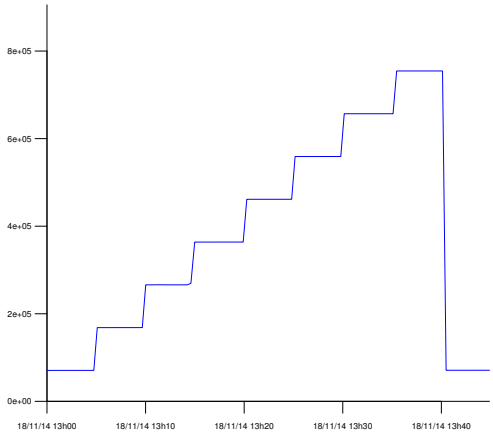
(a) Active



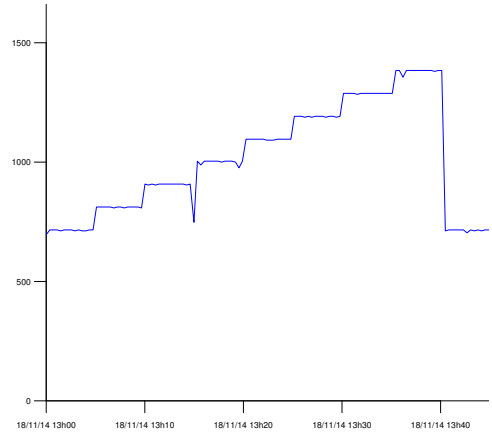
(b) Active_anon



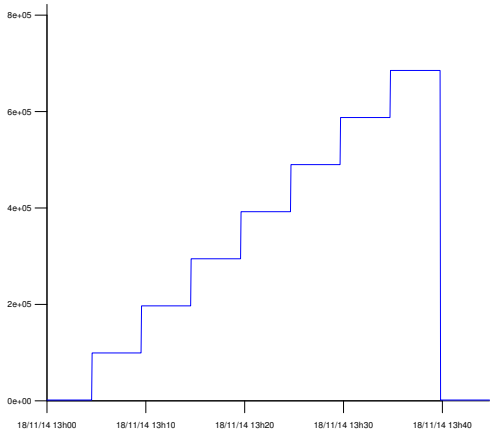
(c) AnonPages



(d) Committed_AS



(e) PageTables



(f) LibVMi (total_vm)

FIGURE 5.3 : Evolutions des compteurs de la commande meminfo et total_vm

les 5 compteurs. Nous remarquons les paliers, résultats des différentes allocations successives. En parallèle, nous avons également mesuré depuis l'hyperviseur, la consommation mémoire du processus mallocme afin de vérifier si la mesure effectuée par notre sonde est cohérente avec les résultats obtenus dans la machine virtuelle. Ainsi, la figure 5.3(f) correspond à la consommation mémoire du processus mallocme récupérée par notre sonde depuis l'hyperviseur. Nous observons que les profils d'activités sont identiques entre les compteurs (mis à part le compteur*PageTables* qui a une sémantique différente). De par ces expérimentations, nous sommes en mesure d'affirmer que la remontée du champ *total_vm* disponible au sein de la structure *mm_struct* est pertinente pour analyser l'activité mémoire d'un processus dans l'objectif de détecter un comportement anormal.

Dans la section suivante, nous nous intérressons à l'évaluation de notre outil d'analyses comportementales de machines virtuelles.

5.3 Evaluation de l'analyse comportementale

Dans la section 4.2.2.2, nous avons discuté de la détection de machines virtuelles à comportements prédéterminés. Il s'agit de détecter des environnements répondant à des critères prédéfinis. De sorte à rendre cette détection plus robuste, nous avons fait le choix de modifier les définitions des comportements prédéterminés en y incluant la notion de "bruits". Nos choix sont évalués dans la section suivante 5.3.1.

Dans la section 4.2.2.3, nous avons décrit nos recherches concernant la détection de machines virtuelles à comportement atypique. Cela consiste à repérer des machines virtuelles dont le comportement diffère des autres machines virtuelles de l'infrastructure supervisée. Pour permettre leur détection, nous avons développé un algorithme regroupant les machines virtuelles par similarité. Les machines virtuelles n'appartenant pas à un groupe sont alors identifiées comme atypiques. La section 5.3.2 présente les résultats de notre algorithme, comparés à ceux de l'algorithme K-MEANS.

5.3.1 Recherche de comportements pré-déterminés

La machine sur laquelle les évaluations ont été réalisées dispose d'un processeur Intel(R) Core(TM) i7-2760QM CPU @ 2.40GHz et de 4 Go de RAM. Le système d'exploitation utilisé est Ubuntu 13.10 64 bits, du noyau 3.11.0-26-generic, d'Oracle Java 1.7.0_45 et de MySQL server 5.5.37. Les évaluations ont été effectuées directement sur les données brutes, non résumées. Les temps de calculs dépendent du nombre d'entrées dans la base de données. Pour informations, le temps de calcul moyen sur une petite base de données (DC01) est de 2.40 secondes et de 72.96 secondes sur une grande base de données (DC08).

Les tableaux 5.4, 5.3 et 5.5 correspondent aux pourcentages de machines virtuelles ayant un comportement pré-déterminés, pour les huit centres de données sélectionnées dans cette thèse. L'analyse est réalisée par créneau d'une heure sur une plage de temps de l'ordre d'une semaine à plusieurs mois. Le tableau 5.4 représente les résultats obtenus pour les machines virtuelles ayant un comportement 'busy', 'undersized' ou 'oversized'. L'investigation de machines virtuelles busy est importante dans la mesure où ce sont des machines ayant une très forte activité. Nous ne savons pas si cette activité est normale ou pas mais la détection permet d'alerter l'administrateur sur l'état de ces machines virtuelles (point de vigilance). Il s'agit de la même chose pour les machines virtuelles 'undersized'. Les machines virtuelles ayant ce comportement sont des machines avec une forte activité (supérieure à 70%). Ces machines sous-dimensionnées peuvent être surchargées, ce qui peut impacter leur performance. De cette façon, l'identification de machines virtuelles 'undersized' permet d'alerter l'administrateur avant d'éventuelles dégradations de performances. Il peut ainsi modifier les ressources allouées à ces machines virtuelles en leur attribuant davantage de VCPU et/ou de mémoire. A l'inverse, la recherche de machines virtuelles 'oversized' permet à l'administrateur de gérer au mieux les ressources allouées aux machines virtuelles et ainsi économiser des ressources systèmes.

Le tableau 5.3 montre les résultats obtenus pour le comportement 'idle'. Pour rappel, une machine virtuelle à un comportement 'idle' si cette dernière n'a pas d'activité. C'est-à-dire que 100% de ces va-

Noise filtering (%)	**100**			**99.99**			**99.9**			**99**		
Inactivity threshold (%)	10	5	2	10	5	2	10	5	2	10	5	2
DC01	0.8	0	0	1.6	0.8	0	1.6	0.8	0	4.9	1.6	0
DC02	0	0	0	0	0	0	0.8	0	0	5.7	4.1	0
DC03	4.8	2.4	0	6.1	4.8	0	9.6	7.2	3.6	19.2	13.2	4.8
DC04	0	0	0	0	0	0	1.9	0.7	0	9.2	2.6	0
DC05	1.6	0.7	0.4	2.9	0.7	0.9	6.4	1.3	0.9	19.7	8.4	0.9
DC06	1.1	0	0	1.1	0	0	1.1	0	0	2.2	1.1	0
DC07	1.0	0.5	0.2	1.2	0.5	0.2	3.6	0.5	0.2	16.1	2.9	0.5
DC08	1.6	0.4	0	2.9	0.4	0	5.1	2.1	0	15.7	6.1	0.8

TABLE 5.3 : Pourcentages de machines virtuelles idle

	Buzy				**Oversized**				**Undersized**			
Noise filtering (%)	100	99.99	99.9	99	100	99.99	99.9	99	100	99.99	99.9	99
DC01	0	0	0	0	12.1	13.8	40.6	76.4	0	0	0	0
DC02	0	0	0	0	5.7	9.8	30.1	77.1	0	0	0	0
DC03	0	0	0	0	18.1	21.6	38.5	84.3	0	0	0	0
DC04	0	0	0	0	11.2	18.5	41.7	68.8	0.6	0.6	0.6	0.6
DC05	0	0	0	0	27.8	34.9	52.1	83.1	0.3	0.3	0.3	0.3
DC06	0	0	0	0	6.8	9.1	36.3	76.3	0	0	0	0
DC07	0	0	0	0	22.9	25.3	42.8	91.1	0	0	0.1	0.3
DC08	0.1	0.2	0.2	0.2	13.1	19.4	31.1	75.7	0.2	0.2	0.2	0.2

TABLE 5.4 : Pourcentages de machines virtuelles busy, oversized et undersized

Noise filtering (%)	**100**			**99.99**			**99.9**			**99**		
Inactivity threshold (%)	10	5	2	10	5	2	10	5	2	10	5	2
DC01	21.9	7.3	0	22.7	7.3	0	22.7	7.3	0	24.4	7.3	0
DC02	17.2	0.8	0	17.2	0.8	0	17.2	0.8	0	18.1	1.6	0
DC03	26.5	9.6	1.2	26.5	9.6	1.2	26.5	9.6	1.2	27.7	9.6	1.2
DC04	17.9	3.3	1.3	17.9	3.3	1.3	17.9	3.3	1.3	18.5	3.9	1.9
DC05	18.8	7.4	1.3	18.8	7.4	1.3	18.8	7.4	1.3	19.7	8.7	1.3
DC06	40.9	15.7	3.4	40.9	15.7	3.4	40.9	15.7	3.4	43.2	15.7	3.4
DC07	39.9	12.6	0.7	39.9	12.6	0.7	39.9	12.6	0.7	41.5	13.2	0.8
DC08	23.8	8.2	0.9	23.8	8.2	0.9	23.9	8.3	1.0	25.1	8.8	1.1

TABLE 5.5 : Pourcentages de machines virtuelles lazy

leurs sont en dessous d'un certain seuil (2%, 5% et 10%). L'identification de machines virtuelles ayant ce comportement permet à l'administrateur de cibler les machines qui peuvent être éteintes voir supprimées. L'objectif est de libérer des ressources systèmes et optimiser l'espace de stockage.

Le tableau 5.5 indique les pourcentages de machines virtuelles ayant un comportement 'lazy'. Ce dernier est un comportement particulier. En effet, une machine virtuelle est dite 'lazy' si elle est 'idle' au moins 90% du temps et dispose de pics d'activité supérieurs à 30% sur les 10% du temps restant. Ici, nous recherchons des machines virtuelles disposant d'une activité processeur ou mémoire non négligeable durant un court instant (moins de 10% du temps). Ces machines virtuelles ressemblent à des machines virtuelles 'idle' mais leurs profils nécessitent une analyse plus approfondie par l'administrateur.

En observant les résultats des différents tableaux, nous montrons l'importance de prendre en considération la notion de bruit dans la recherche de comportements. Pour certains comportements tels que 'busy' ou 'undersized', le pourcentage de machines virtuelles est quasiment identique entre 100% et 99%. Ce résultat est cohérent dans la mesure où il est rare de trouver une ou plusieurs machines virtuelles surchargées dans les centres de données. A l'inverse, le résultat est différent pour les machines virtuelles 'oversized'. Le pourcentage moyen obtenu pour 99% est 5 fois supérieur au pourcentage moyen obtenu à 100%. Similairement, le pourcentage moyen de machines virtuelles 'idle' à 10%, 100% du temps est de 1.36% alors qu'il est de 11.58% si nous prenons en compte le bruit, à 99%. Ce qui représente un ratio non négligeable de x8.

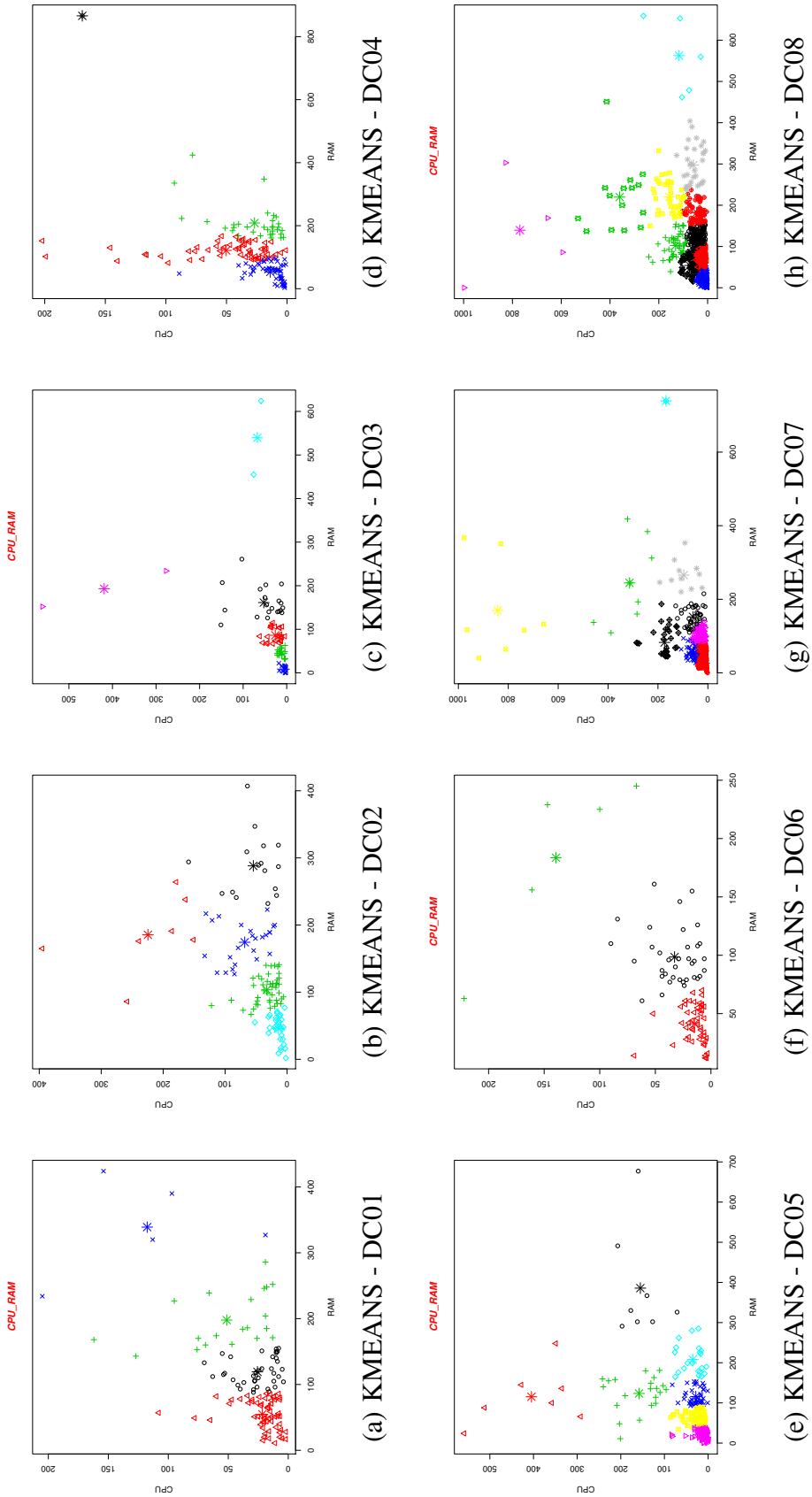
La notion de performance n'est finalement pas un critère complètement objectif. Cette subjectivité est prise en compte en proposant à l'administrateur les différents niveaux de bruits.

5.3.2 Recherche de comportements atypiques

Afin de valider notre algorithme, nous avons analysé les résultats de partitionnement des méthodes présentées dans l'état de l'art sur des données réelles collectées par l'outil DCScope dans une cinquantaine de centres de données. Pour des raisons de confidentialité, les entreprises ne peuvent être citées et les données brutes colletées ne sont pas diffusables. Par souci de compacité, nous avons choisi ici de nous focaliser sur huit centres de données.

R est un langage de programmation permettant de traiter, organiser et représenter rapidement d'importants volumes de données. R dispose également d'une grande variété de fonctions dans le domaine du Clustering. Dans nos expérimentations, nous avons utilisé la fonction 'boxplot' (afin d'obtenir, à partir des données de consommations stockées en BD, le premier quartile, la médiane, la moyenne, le troisième quartile, la valeur minimum et la valeur maximum pour chaque machine virtuelle) et la fonction kmeans pour comparer les résultats de ce dernier avec notre algorithme. De plus, toutes les représentations graphiques présentes dans cette évaluation ont été réalisées avec R.

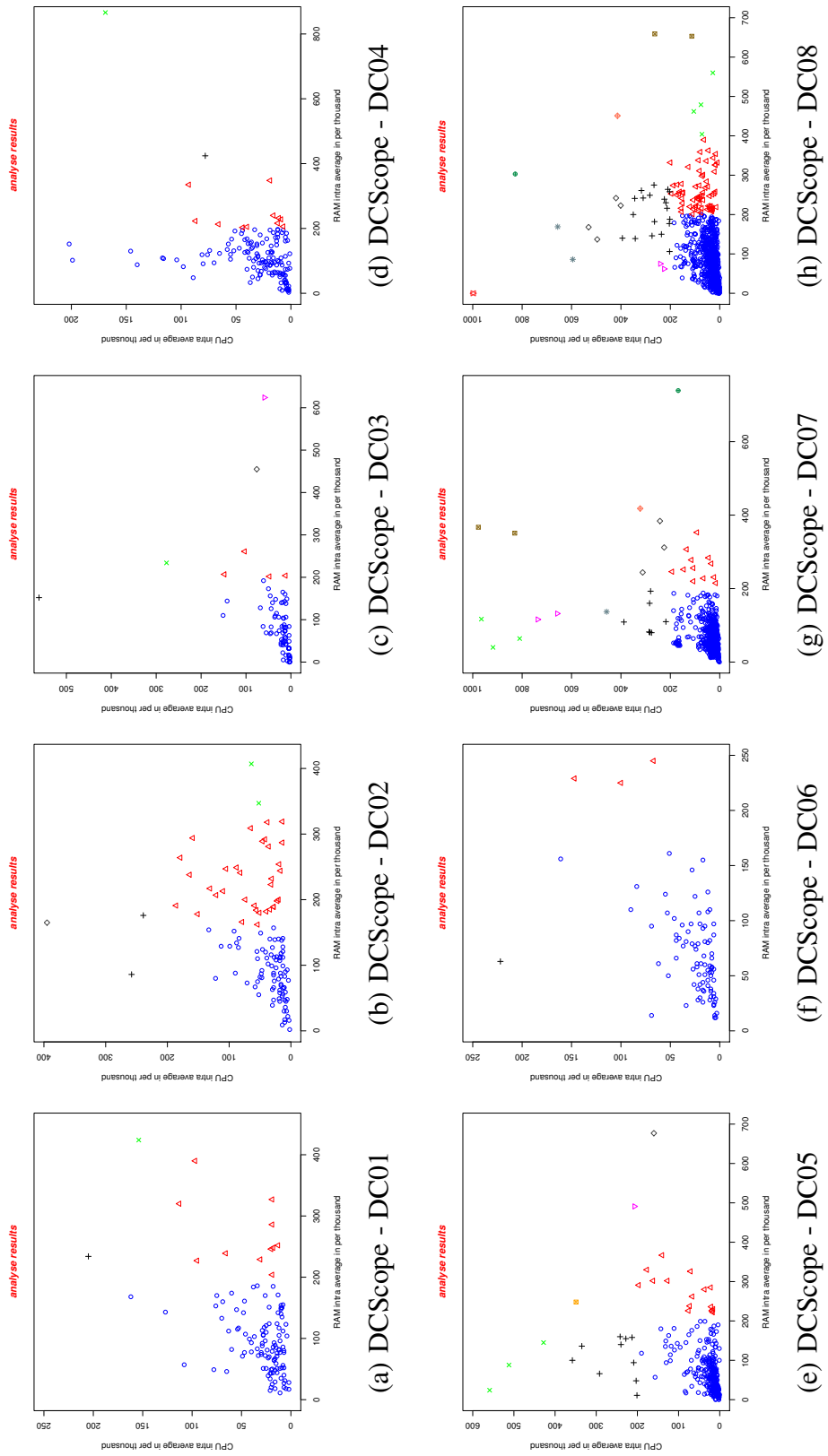
Pour comparer l'algorithme K-MEANS avec notre algorithme, nous avons déterminé le nombre de groupe (k) par une première itération de notre algorithme. Dans ces exemples, nous prenons des points sur un espace à deux dimensions : la moyenne CPU et mémoire de chaque machine virtuelle sur la période d'analyse. L'abscisse correspond au taux d'utilisation CPU et l'ordonnée au taux d'utilisation mémoire. Sur ces graphiques, chaque groupe de machines virtuelles est représenté par une couleur et un symbole. Les figures 5.5(a), 5.5(b), 5.5(c), 5.5(d), 5.5(e), 5.5(f),5.5(g) et 5.5(h) détaillent le partitionnement réalisé par notre approche sur chaque centre de données. Nous obtenons un groupe composé d'un nombre important de machines virtuelles et de plus petits groupes composés de 1 à 3 machines virtuelles. Les figures 5.4(a), 5.4(b), 5.4(c), 5.4(d), 5.4(e), 5.4(f), 5.4(g) et 5.4(h) représentent les résultats obtenus en utilisant l'algorithme de partitionnement K-means. Chaque centre de données est découpé en un nombre important de petits groupes, assez homogènes en nombre de points (machines virtuelles) ce qui est finalement peu exploitable dans notre cas.



(a) KMEANS - DC01 (b) KMEANS - DC02 (c) KMEANS - DC03 (d) KMEANS - DC04

(e) KMEANS - DC05 (f) KMEANS - DC06 (g) KMEANS - DC07 (h) KMEANS - DC08

FIGURE 5.4 : Partitionnement K-MEANS



(a) DCScope - DC01 (b) DCScope - DC02 (c) DCScope - DC03 (d) DCScope - DC04

(e) DCScope - DC05 (f) DCScope - DC06 (g) DCScope - DC07 (h) DCScope - DC08

FIGURE 5.5 : Partitionnement DCScope

Pour obtenir ces résultats, les données brutes stockées en base de données sont dans un premier temps exportées en CSV puis traitées avec R pour calculer les données statistiques de chaque machine virtuelle. Ces données statistiques sont enregistrées dans un fichier pour être réutilisées ultérieurement et éviter de multiplier les accès a la base de données. Les informations statistiques de chaque machine virtuelle sont utilisées par notre algorithme afin de construire le modèle de consommation des machines virtuelles puis ce dernier partitionne en fonction des critères sélectionnés. Les groupes obtenus par notre algorithme sont ensuite graphiquement représentés avec R. Concernant les graphiques obtenus pour K-MEANS, ces derniers sont le résultat de la fonction 'kmeans' implémentée dans R. Les temps de calculs pour traiter les données (hors export en CSV) et représenter graphiquement les résultats de partitionnement de notre algorithme et de K-MEANS sont de l'ordre de la seconde (1.7 seconde en moyenne).

Dans chaque cas, nous identifions 2 ou 3 machines virtuelles à comportement atypique. Nous rappelons que nous définissons une machine virtuelle 'atypique', une machine virtuelle qui dispose d'un profil différent des autres. Graphiquement, ce sont les machines virtuelles représentées par un point séparé des autres. Pour illustrer ce fait, prenons par exemple les figures 5.5(b),5.5(c),5.5(d),5.5(e) et 5.5(h). Sur ces dernières nous remarquons un point fortement isolé des autres points. Ces points ont pour coordonnées (165,395) pour DC02, (152,561) pour DC03, (866,169) pour DC04, (677,160) pour DC05 et (0,998) pour DC08. Ici, il s'agit de machines virtuelles à comportement atypique puisqu'elles n'appartiennent à aucun groupe. En comparaison à l'algorithme K-means, focalisons nous sur les figures 5.4(b), 5.4(c),5.4(d),5.4(e) et 5.4(h). En observant DC04, l'algorithme K-means retourne le même résultat que notre algorithme. Ce qui signifie qu'il a isolé le même point (la même machine virtuelle). En revanche, pour DC02, DC03, DC05 et DC08, K-means a inclus dans des groupes, des points que notre algorithme a isolés. En observant ces groupes attentivement, ce sont des points qui ont des données de consommation très différentes. Par exemple pour DC05, la machine virtuelle considérée par notre algorithme comme atypique (coordonnées 677,160) est incluse dans une groupe de machines virtuelles dont la consommation CPU varie de 300 à 700 pour mille. Nous retrouvons le même comportement pour DC02 et DC03. Dans DC02, la machine virtuelle considérée comme atypique (coordonnées 152,561) est regroupée avec une autre machine virtuelle (coordonnées 234,277). Ces différences s'expliquent par le fait que l'algorithme K-means est sensible aux bruits. Il cherche à consolider les points isolés avec d'autres points, bien que ces points soient distants. Cette sensibilité est due au nombre K. De plus, K-means coupe en plusieurs groupes les masses de points denses (en bleu et rouge dans nos résultats). Cette division est inutile parce que les machines virtuelles présentes dans ces groupes disposent de profils similaires. Ces machines doivent donc appartenir au même groupe. Ce comportement est particulièrement visible sur les figures 5.5(b)-5.4(b) et 5.5(h)-5.4(h).

Les observations présentées dans cette section montre l'efficacité de notre algorithme dans la recherche de machines virtuelles à comportement atypique. Un algorithme insensible aux bruits, multi-ressources, multi-critères permettant d'établir des groupes de machines virtuelles à données similaires.

Une machine virtuelle atypique n'est pas forcément une machine virtuelle "à problème". Leur détection permet à l'administrateur de les identifier rapidement et éventuellement agir sur ces dernières.

5.3.3 Evaluation de l'outil d'analyses comportementales

Dans l'objectif d'évaluer l'outil d'analyses comportementales, nous avons mesuré plusieurs temps de traitement pour huit centres de données représentatifs, c'est-à-dire de tailles différentes, composés de 6 à 57 serveurs physiques et de 113 à 880 machines virtuelles. Pour réaliser ces mesures, nous avons dans un premier temps importé les bases de données sur un serveur physique. Ce dernier est un serveur Dell PowerEdge R420, composé d'un processeur Intel(R) Xeon(R) CPU E5-2407 @ 2.20GHz et de 32 Go de mémoire. L'environnement utilisé est Ubuntu 12.10 64 bits, noyau 3.8.0-29 et MySQL server 5.5.35. Une fois les bases de données importées, nous avons mis en place un environnement d'évaluations permettant l'exécution de différents scripts, composant notre outil, sur chacune des bases de données. Les diverses informations nécessaires à l'évaluation sont enregistrées dans des fichiers de log. Nous avons ainsi mesuré le

	Serveurs	VMS	Run	Off	Idle	Busy	Oversized	Undersized	Lazy
DC1	41	517	440	77	15	0	36	2	22
DC2	10	355	229	96	114	0	272	0	79
DC3	6	113	73	40	9	0	57	0	3
DC4	11	287	271	16	27	0	241	0	34
DC5	57	880	712	168	92	2	377	2	92
DC6	6	561	406	155	44	0	302	1	34
DC7	13	232	207	25	2	0	7	0	2
DC8	6	183	119	64	5	0	46	0	6

TABLE 5.6 : Analyses comportementales de machines virtuelles pour 8 centres de données

	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
Durée monitoring (en H)	3290	5471	6788	10579	1612	16845	5186	6164
Taille BD (en Mo)	21576	29143	8879	31112	11632	83300	12726	11454
Temps pour consolidation table de consommations (en ms)	3203	2238	329	1018	5056	2375	947	1427
Temps pour un jour (en ms)	11360	8013	6592	10346	12357	17544	7345	6859
Temps pour deux jours (en ms)	12365	9130	6656	10611	14758	18437	7576	6850
Temps pour une semaine (en ms)	17029	10405	7111	13179	28619	23418	9215	7344
Temps pour un mois (en ms)	40500	19379	7702	23104	94977	46600	17111	9838

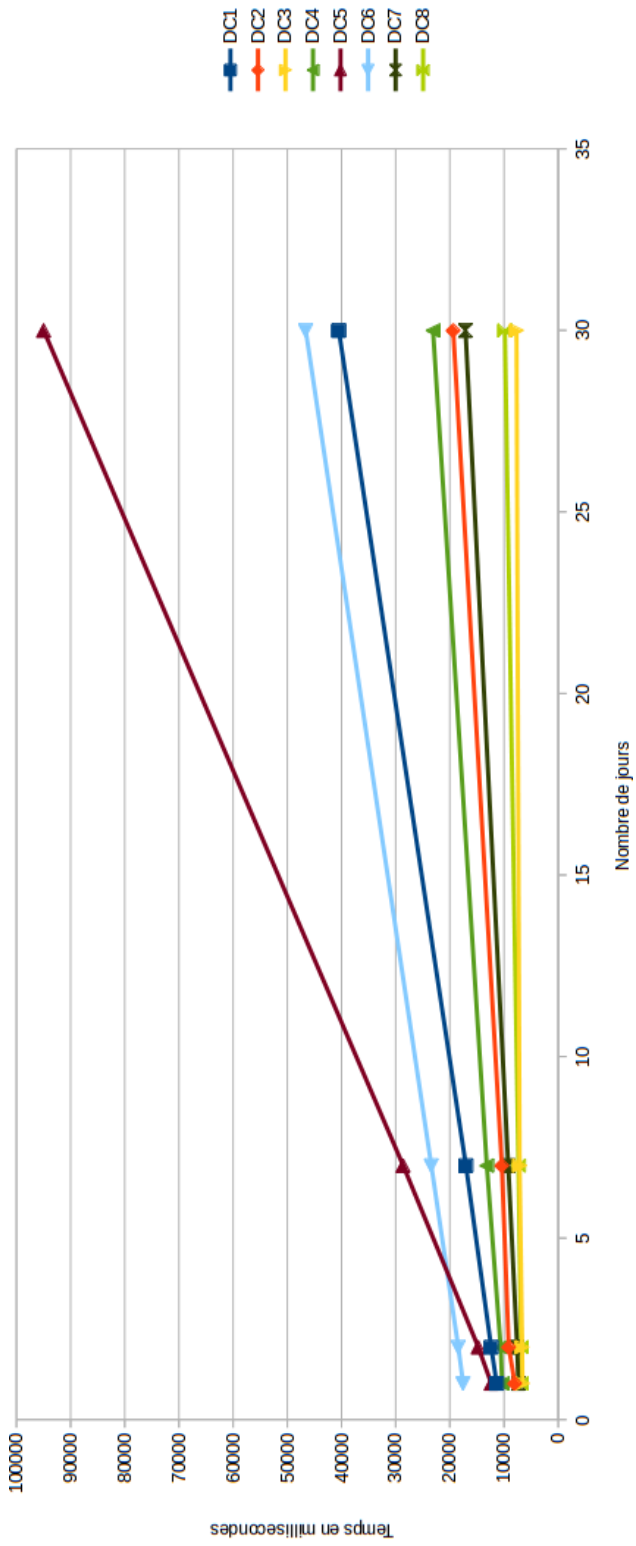
TABLE 5.7 : Analyses des temps de calculs pour 8 centres de données

temps mis pour consolider une table contenant les consommations des serveurs physiques et des machines virtuelles ainsi que mesuré le temps de calculs des comportements des machines virtuelles. Les résultats obtenus sont renseignés dans les tableaux 5.6 et 5.7.

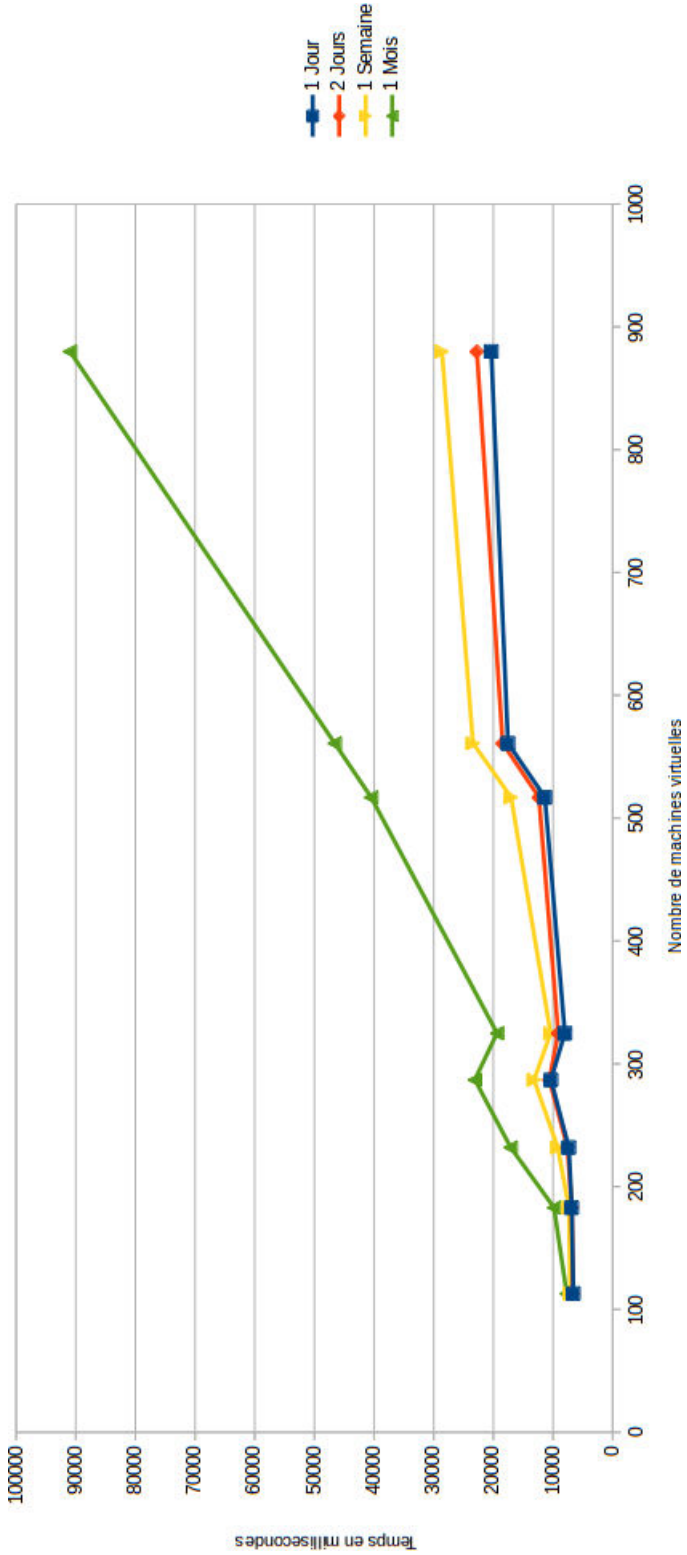
Le tableau 5.6 présente le nombre de serveurs physiques et de machines virtuelles pour chaque centre de données ainsi que les résultats obtenus pour les recherches de machines virtuelles à comportement prédéterminés (idle, busy, overzised, undersized, lazy). Le tableau 5.7 indique les temps moyens mesurés pour le calcul de ces comportements.

Pour obtenir ces résultats, nous avons fait varier la période analysée de sorte à calculer sur une journée, deux journées, une semaine et un mois. Pour chaque centre de données, nous avons selectionné toute l'infrastructure. En croisant les informations des tableaux 5.6 et 5.7, nous observons que le temps de calcul est dépendant à la fois de la période analysée mais également du nombre d'élements (machines physiques et virtuelles) à analyser. Les courbes 5.6(a) et 5.6(b) illustrent ce fait.

Ainsi, sur la courbe 5.6(a) nous observons la proportionalité entre le nombre de jours analysés et le temps de calcul pour les huit centres de données sélectionnés dans cette évaluation. Sur la courbe 5.6(b), nous constatons également la proportionalité entre le nombre de machines virtuelles analysées et les temps de calcul. Seul le centre de données DC02 (les points correspondant à 325 machines virtuelles) possède des temps de calculs légèrement à la baisse par rapport à la tendance.



(a) Temps de calculs en fonction de la période



(b) Temps de calculs en fonction du nombre de machines virtuelles

III

Conclusion

6

Conclusion et Perspectives

La surveillance des ressources systèmes consommées par les machines virtuelles sur un serveur physique est un élément clé pour garantir le bon fonctionnement d'une plateforme IaaS. Les travaux effectués dans le cadre de cette thèse ont pour objectif de développer un système d'analyse avancée et d'optimisation d'infrastructures Cloud, allant de l'introspection à l'analyse comportementale des machines virtuelles.

6.1 Conclusion

Afin d'atteindre notre objectif, la première partie de cette thèse aborde les thèmes du Cloud Computing et de la Virtualisation. Dans un premier temps, une présentation du Cloud Computing est réalisée. Nous le définissons et présentons les différents niveaux de services qu'offre le Cloud pour se focaliser ensuite au niveau Iaas (Infrastructure as a Service), niveau sur lequel porte les contributions de cette thèse. Dans un second temps, nous nous focalisons sur la virtualisation, la technologie clé sur laquelle repose le Cloud Computing. Cette partie permet de décrire le fonctionnement des hyperviseurs et d'introduire les machines virtuelles.

La deuxième partie de cette thèse présente les différentes contributions apportées :

- **Études des métriques de performances :** Dans le chapitre 3, nous présentons dans une première section l'étude des métriques de performances, réalisée sous VMware®. Nous avons fait ce choix dans le mesure où c'est le système d'IaaS le plus répandu dans les entreprises. Ce dernier dispose d'un nombre conséquent de compteurs et une gestion très sophistiquée de la mémoire (environ 50 compteurs rien que pour cette dernière). Il est donc indispensable de bien comprendre la différence entre ces compteurs afin de mieux les utiliser. Ainsi, nous avons dans une première section, détaillé les différents états processeur dans lequel peut se trouver un vCPU (RUN, WAIT, READY ou COSTOP). Dans une seconde section, nous avons expliqué finement les compteurs mémoire, pertinent dans l'analyse de l'activité mémoire des machines virtuelles. Dans les deux dernières sections, nous nous sommes focalisés sur les compteurs disque et réseau. En effet, les bonnes performances d'une machine virtuelle ne se résument pas uniquement en l'étude des ressources processeur et mémoire. L'accès au disque de façon optimale est également un enjeu important. Pour superviser les machines virtuelles, nous avons développé une sonde logicielle, qui se connecte au VCenter et qui récupère les informations toutes les minutes. Dans le chapitre 5 section 5.1, nous avons évalué cette sonde logicielle et montré que cette dernière n'impacte que faiblement l'activité du système d'exploitation hébergeant le VCenter. De plus, nous montrons que la récupération des données toutes les minutes (au lieu de 20 secondes) n'interfère pas sur la précision des analyses comportementales.

Dans la seconde section de ce chapitre, nous discutons de l'introspection de machines virtuelles, technique qui consiste à superviser le comportement des machines virtuelles. Nous nous sommes focalisés dans cette section sur l'hyperviseur KVM. Sur KVM, il est difficile d'obtenir précisément l'activité processeur et mémoire des machines virtuelles, surtout la mémoire. Nous avons alors étudié différentes librairies d'introspection et retenu LibVMi [XLXJ12, Pay12].

Dans la troisième section de ce chapitre, nous présentons notre contribution, liée à l'introspection. Une étude sur les structures du noyau Linux et sur les différents compteurs mémoire ont été réalisées. Suite à ces études, nous proposons une sonde logicielle non intrusive permettant de superviser la consommation des ressources systèmes des processus s'exécutant dans une machine virtuelle depuis l'hyperviseur. Cette solution, destinée aux administrateurs de parcs informatiques virtualisés, permet ainsi de détecter les processus défaillants ou atypiques au sein d'une machine virtuelle. Dans le chapitre 5 section 5.2, nous évaluons notre sonde logicielle non intrusive en démontrant la pertinence du choix du compteur mémoire utilisé par comparaison à d'autres compteurs mémoire récupérés par intrusion.

- **Développement d'un outil d'analyses comportementales :** Dans le chapitre 4, nous présentons les travaux réalisés autour des différentes façons d'analyser un centre de données. Dans cette thèse, nous proposons deux niveaux d'analyses. Ainsi, dans une première section, nous détaillons les diverses analyses effectuées au niveau des serveurs physiques et virtuels. Il s'agit à ce niveau d'étudier les métriques de performances permettant de prévenir ou de détecter d'éventuelles anomalies liées aux ressources processeur, mémoire réseau et disque. Cette section décrit les métriques à surveiller ainsi que les seuils à respecter pour assurer de façon optimale le bon fonctionnement d'une infrastructure virtualisée.

  Dans une seconde section, nous abordons l'analyse au niveau du centre de données. Il s'agit d'étudier le comportement des machines virtuelles. Nous y présentons nos contributions portant sur la détection de comportements pré-déterminés et atypique. Les comportements pré-déterminés ont été modifiés de sorte à leur rajouter de la robustesse. En effet, nous avons remarqué que même si elle ne fait rien, une machine virtuelle dispose toujours de pics d'activité. Ces pics influent sur la détection comportementale. De ce fait, nous avons ajouté une notion de bruit permettant de prendre en charge cette activité particulière. Les évaluations portant sur ces contributions, réalisées dans le chapitre 5 section 5.3.1 permettent de mettre en avant la nécessité de prendre en considération la notion de bruits dans la recherche de machines virtuelles à comportement pré-déterminé. Nous le démontrons par l'analyse de huit centres de données.

  La recherche de machines virtuelles à comportement atypique permet de se focaliser sur celles dont l'activité diffère des autres machines virtuelles du centre de données. Pour cela, nous avons développé notre propre algorithme de recherche et évalué dans le chapitre 5 section 5.3.2 l'efficacité de ce dernier face à l'algorithme K-MEANS [M+67].

6.2 Perspectives

Docker est un logiciel open-source basé sur LXC [lin] (pour LinuX Containers) permettant de déployer des applications dans des conteneurs logiciels. LXC permet de créer un environnement virtuel au niveau du système d'exploitation, en utilisant l'isolation. Il s'agit de cloisonner l'exécution des applications dans des "zones d'exécution". Chaque conteneur s'exécute de façon autonome et isolé les uns des autres. Alors qu'une machine virtuelle dispose de son propre système d'exploitation, le conteneur s'appuie sur le noyau du système d'exploitation sous-jacent et embarque uniquement les librairies et binaires nécessaires à l'application qu'il encapsule, allégeant ainsi ses besoins en ressources.

Différentes études [data, datb, rig] mettent en avant la progression de l'utilisation de Docker au sein des entreprises. Selon les études de Datadog [data, datb], entre septembre 2014 et août 2015, le nombre de serveurs hébergeant des conteneurs, supervisés par leur solution, a été multiplié par 5. Une autre étude [rig], de RightScale, met également en avant cette progression, indiquant que l'adoption de Docker par leurs clients a doublé, passant de 13% en 2015 à 27% en 2016.

Cette progression peut également s'expliquer par le fait que la virtualisation telle qu'employée actuellement est coûteuse, aussi bien en matériels que logiciels. En effet, les machines virtuelles disposent chacune de leur propre système d'exploitation, qui nécessite des ressources matérielles et dispose de coûts directs (licence du système d'exploitation, de l'anti-virus, etc) et indirects (maintenance, exploitation). De plus, ces systèmes d'exploitation sont de plus en plus gourmands en ressources (notamment mémoire et disque), diminuant ainsi le nombre de machines virtuelles pouvant être hébergées sur un serveur physique.

De ce fait, il serait intéressant de poursuivre nos travaux autour de la technologie Docker, selon plusieurs orientations.

6.2.1 Etude énergétique Docker/Virtualisation

Dans la littérature, nous trouvons peu de travaux autour de Docker et l'efficience énergétique [PDCB15, JM16]. Le premier travail serait d'étudier les gains énergétiques de la solution Docker par rapport à la virtualisation. Des gains probablement liés aux performances (une application exécutée dans un conteneur utilise moins de ressource processeur que dans une machine virtuelle) et certainement liés à l'empreinte mémoire. En effet, un conteneur nécessite moins de mémoire qu'une machine virtuelle pour exécuter son application, permettant une consolidation de conteneurs plus importante. Moins de serveurs physiques donc consommation énergétique réduite. Le second travail consisterait à définir le modèle énergétique d'une infrastructure à conteneurs.

6.2.2 Sonde de supervision

De la même façon que nous supervisons les machines virtuelles s'exécutant au sein d'infrastructures virtualisées VMWare et KVM, nous développerons une nouvelle sonde logicielle non intrusive capable de superviser les conteneurs.

6.2.3 Gestion des capacités

Nous modifierons nos algorithmes d'analyses de sorte à analyser l'activité des conteneurs et ainsi détecter des conteneurs à comportement pré-déterminé (idle, busy, lazy, etc...). De plus, l'analyse fine des métriques de performances des conteneurs permettra également le développement d'un outil de simulation de planification des capacités. Ainsi, un administrateur pourra simuler l'ajout/suppression de projets (conteneurs) et l'ajout ou la perte de serveurs (simulation de pannes). Cette outil sera en mesure de déduire soit les dates de saturation de l'infrastructure selon plusieurs ressources (processeur, mémoire, disque) soit déterminer la criticité de l'infrastructure en cas de panne pour la mise en place d'un PCA (Plan de Continuité d'Activité).

6.2.4 Plan de consolidation

Docker propose un outil permettant la consolidation de conteneur, nommé Swarm. Cependant, cette algorithme de placement est assez simple et ne dispose pas d'heuristiques liées à l'optimisation énergétique. A partir des données de supervision et de la modélisation énergétique, l'objectif serait de développer une heuristique déterminant le placement idéal d'un point de vue énergétique.

Bibliographie

[ABDDP13] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Cloud monitoring : A survey. *Computer Networks*, 57(9) :2093–2115, 2013.

[amaa] Amazon elastic compute cloud (ec2). http://aws.amazon.com/ec2/. 2015.

[amab] Amazon simple storage service (amazon s3). https://aws.amazon.com/fr/s3/. 2015.

[BAL15] Erick Bauman, Gbadebo Ayoade, and Zhiqiang Lin. A survey on hypervisor-based monitoring : Approaches, applications, and evolutions. *ACM Computing Surveys (CSUR)*, 48(1) :10, 2015.

[BBG10] Rajkumar Buyya, James Broberg, and Andrzej M Goscinski. *Cloud computing : principles and paradigms*, volume 87. John Wiley & Sons, 2010.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5) :164–177, 2003.

[BDGR97] Edouard Bugnion, Scott Devine, Kinshuk Govil, and Mendel Rosenblum. Disco : Running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems (TOCS)*, 15(4) :412–447, 1997.

[Bel66] Laszlo A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems journal*, 5(2) :78–101, 1966.

[BGTV13] Ishan Banerjee, Fei Guo, Kiran Tati, and Rajesh Venkatasubramanian. Memory overcommitment in the esx server. *VMware Technical Journal*, 2, 2013.

[BJW+10] Sina Bahram, Xuxian Jiang, Zhi Wang, Mike Grace, Jinku Li, Deepa Srinivasan, Junghwan Rhee, and Dongyan Xu. Dksm : Subverting virtual machine introspection for fun and profit. In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pages 82–91. IEEE, 2010.

[BL09] Tom Brey and Larry Lamers. Using virtualization to improve data center efficiency. *the green grid, whitepaper*, 19, 2009.

[BYV08] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing : Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee, 2008.

[CB05] Susanta Nanda Tzi-cker Chiueh and Stony Brook. A survey on virtualization technologies. *RPE Report*, pages 1–42, 2005.

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[CN01] Peter M Chen and Brian D Noble. When virtual is better than real [operating system relocation to virtual machines]. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 133–138. IEEE, 2001.

[data] 8 surprising facts about real docker adoption - 2015 - datadog. https://www.datadoghq.com/docker-adoption-2015/. 2016.

[datb] 8 surprising facts about real docker adoption - datadog. https://www.datadoghq.com/docker-adoption/. 2016.

[Den68] Peter J Denning. The working set model for program behavior. *Communications of the ACM*, 11(5) :323–333, 1968.

[DGLZ+11] Brendan Dolan-Gavitt, Tim Leek, Michael Zhivich, Jonathon Giffin, and Wenke Lee. Virtuoso : Narrowing the semantic gap in virtual machine introspection. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 297–312. IEEE, 2011.

[DGPL11] Brendan Dolan-Gavitt, Bryan Payne, and Wenke Lee. Leveraging forensic tools for virtual machine introspection. 2011.

[doc] Docker - build, ship, and run any app, anywhere. https://www.docker.com/. 2016.

[DRSL08] Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether : malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62. ACM, 2008.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[For10] Germain Forestier. *Connaissances et clustering collaboratif d'objets complexes multisources*. PhD thesis, Universite de Strasbourg, 2010.

[GB12] Saurabh Kumar Garg and Rajkumar Buyya. Green cloud computing and environmental sustainability. *Harnessing Green IT : Principles and Practices*, pages 315–340, 2012.

[Gol74] Robert P Goldberg. Survey of virtual machine research. *Computer*, 7(6) :34–45, 1974.

[GR+03] Tal Garfinkel, Mendel Rosenblum, et al. A virtual machine introspection based architecture for intrusion detection. In *NDSS*, volume 3, pages 191–206, 2003.

[HC92] Kieran Harty and David R Cheriton. *Application-controlled physical memory using external page-cache management*, volume 27. ACM, 1992.

[HLAP06] Wei Huang, Jiuxing Liu, Bulent Abali, and Dhabaleswar K Panda. A case for high performance computing with virtual machines. In *Proceedings of the 20th annual international conference on Supercomputing*, pages 125–134. ACM, 2006.

[HN08] Brian Hay and Kara Nance. Forensics examination of volatile system data using virtual introspection. *ACM SIGOPS Operating Systems Review*, 42(3) :74–82, 2008.

[JM16] Aymen Jlassi and Patrick Martineau. Benchmarking hadoop performance in the cloud-an in depth study of resource management and energy consumption. In *The 6th International Conference on Cloud Computing and Services Science*, 2016.

[JMF99] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3) :264–323, 1999.

[JWX07] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 128–138. ACM, 2007.

[KJM+14] Rakesh Kumar, Kanishk Jain, Hitesh Maharwal, Neha Jain, and Anjali Dadhich. Apache cloudstack : Open source infrastructure as a service cloud computing platform. In *International Journal of advancement in Engineering technology, Management and Applied Science*, 2014.

[KKL+07] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm : the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[KR09] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[KT15] Jonathan Koomey and Jon Taylor. New data supports finding that 30 percent of servers are 'comatose', indicating that nearly a third of capital in enterprise data centers is wasted. 2015.

[lin] Linux containers. https://linuxcontainers.org/. 2016.

[Lin13] Zhiqiang Lin. Toward guest os writable virtual machine introspection. *VMware Technical Journal*, 2(2) :9–14, 2013.

[Low13] Scott D Lowe. Best practices for oversubscription of cpu, memory and storage in vsphere virtual environments. *Technical Whitepaper, Dell*, 2013.

[LZ14] Gang Lu and Wen Hua Zeng. Cloud computing survey. In *Applied Mechanics and Materials*, volume 530, pages 650–661. Trans Tech Publ, 2014.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[Mar02] Debbie Marr. Hyper-threading technology in the netburst® microarchitecture. *14th Hot Chips*, 2002.

[Men05] Daniel A Menascé. Virtualization : Concepts, applications, and performance modeling. In *Int. CMG Conference*, pages 407–414, 2005.

[MG11] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[MK07] Dan Marinescu and Reinhold Kröger. State of the art in autonomic computing and virtualization. *Distributed Systems Lab, Wiesbaden University of Applied Sciences*, pages 1–24, 2007.

[MLM11] Dejan Milojičić, Ignacio M Llorente, and Ruben S Montero. Opennebula : A cloud management tool. *IEEE Internet Computing*, (2) :11–14, 2011.

[msv] Microsoft virtual pc. https://www.microsoft.com/fr-FR/download/details.aspx?id=3702. 2015.

[MT14] Asit More and Shashikala Tapaswi. Virtual machine introspection : towards bridging the semantic gap. *Journal of Cloud Computing*, 3(1) :1–14, 2014.

[Mun15] Muhammad Zeeshan Munir. *VMware vSphere Troubleshooting*. Packt Publishing Ltd, 2015.

[MW05] Al Muller and Seburn Wilson. Virtualization with vmware esx server. 2005.

[OOW93] Elizabeth J O'neil, Patrick E O'neil, and Gerhard Weikum. The lru-k page replacement algorithm for database disk buffering. *ACM SIGMOD Record*, 22(2) :297–306, 1993.

[ora] Oracle virtualbox. https://www.virtualbox.org/. 2015.

[Par11] Jonathan Parri. Introduction to cloud computing. *Introduction to Cloud Computing*, 2011.

[Pay12] Bryan D Payne. Simplifying virtual machine introspection using libvmi. *Sandia Report*, 2012.

[PCSL08] Bryan D Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares : An architecture for secure active monitoring using virtualization. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 233–247. IEEE, 2008.

[PDCB15] Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N Calheiros, and Rajkumar Buyya. A framework and algorithm for energy efficient container consolidation in cloud data centers. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 368–375. IEEE, 2015.

[PG74] Gerald J Popek and Robert P Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7) :412–421, 1974.

[PSE11] Jonas Pfoh, Christian Schneider, and Claudia Eckert. Nitro : Hardware-based system call tracing for virtual machines. In *Advances in Information and Computer Security*, pages 96–112. Springer, 2011.

[Res10] Applied Computer Research. Defining the data center market and data center market size. 2010.

[Res11] Applied Computer Research. Identifying it markets and market size by number of servers. 2011.

[RG05] Mendel Rosenblum and Tal Garfinkel. Virtual machine monitors : Current technology and future trends. *Computer*, 38(5) :39–47, 2005.

[rig] Cloud computing trends : 2016 state of the cloud survey. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey. 2016.

[SAE12] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack : toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3) :38–42, 2012.

[SDN14] SK Sowmya, P Deepika, and J Naren. Layers of cloud–iaas, paas, and saas : A survey. *International Journal of Computer Science and Information Technologies*, 5(3) :4477–4480, 2014.

[SE97] Erich Schikuta and Martin Erhart. The bang-clustering system : Grid-based data analysis. In *Advances in Intelligent Data Analysis Reasoning about Data*, pages 513–524. Springer, 1997.

[SFS06] Joel H Schopp, Keir Fraser, and Martine J Silbermann. Resizing memory with balloons and hotplug. In *Proceedings of the Linux Symposium*, volume 2, pages 313–319, 2006.

[SIB+14] Sahil Suneja, Canturk Isci, Vasanth Bala, Eyal de Lara, and Todd Mummert. Non-intrusive, out-of-band and out-of-the-box systems monitoring in the cloud. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, pages 249–261. ACM, 2014.

[SLCL09] Monirul I Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. Secure in-vm monitoring using hardware virtualization. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 477–487. ACM, 2009.

[SPE11] Christian Schneider, Jonas Pfoh, and Claudia Eckert. A universal semantic bridge for virtual machine introspection. In *Information Systems Security*, pages 370–373. Springer, 2011.

[TG05] Irina Chihaia Tuduce and Thomas R Gross. Adaptive main memory compression. In *USENIX Annual Technical Conference, General Track*, pages 237–250, 2005.

[vmwa] Troubleshooting storage performance in vsphere – part 1 – the basics. http://blogs.vmware.com/vsphere/2012/05/troubleshooting-storage-performance-in-vsphere-part-1-the-basics.html. 2012.

[vmwb] Vmware esxi. https://www.vmware.com/fr/products/esxi-and-esx/overview. 2016.

[vmwc] Vmware workstation / player. https://www.vmware.com/fr/products/workstation. 2015.

[VMw11] VMware. Understanding memory resource management in vmware vsphere 5.0. 2011.

[VMw13] VMware. The cpu scheduler in vmware vsphere 5.1. 2013.

[VV09] Anthony Velte and Toby Velte. *Microsoft virtualization with Hyper-V*. McGraw-Hill, Inc., 2009.

[Wal02] Carl A Waldspurger. Memory resource management in vmware esx server. *ACM SIGOPS Operating Systems Review*, 36(SI) :181–194, 2002.

[WCC+08] John Paul Walters, Vipin Chaudhary, Minsuk Cha, Salvatore Guercio Jr, and Steve Gallo. A comparison of virtualization technologies for hpc. In *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, pages 861–868. IEEE, 2008.

[WTM+14] Thu Yein Win, Huaglory Tianfield, Quentin Mair, Taimur Al Said, and Omer F Rana. Virtual machine introspection. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 405. ACM, 2014.

[XLXJ12] Haiquan Xiong, Zhiyong Liu, Weizhi Xu, and Shuai Jiao. Libvmi : A library for bridging the semantic gap between guest os and vmm. In *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, pages 549–556. IEEE, 2012.

[YAMV15] BO Yenké, AA Abba Ari, C Dibamou Mbeuyo, and DA Voundi. Virtual machine performance upon intensive computations. *GSTF Journal on Computing (JoC)*, 4(3), 2015.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18, 2010.

l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

MINES Nantes
Institut Mines-Télécom

Thèse de Doctorat

Frédéric DUMONT

Analyses et préconisations pour les centres de données virtualisés

Analysis and recommendations for virtualized datacenters

Résumé
Cette thèse présente deux contributions. La première contribution consiste en l'étude des métriques de performance permettant de superviser l'activité des serveurs physiques et des machines virtuelles s'exécutant sur les hyperviseurs VMware et KVM. Cette étude met en avant les compteurs clés et propose des analyses avancées dans l'objectif de détecter ou prévenir d'anomalies liées aux quatre ressources principales d'un centre de données : le processeur, la mémoire, le disque et le réseau. La seconde contribution porte sur un outil pour la détection de machines virtuelles à comportements pré-déterminés et/ou atypiques. La détection de ces machines virtuelles à plusieurs objectifs. Le premier, permettre d'optimiser l'utilisation des ressources matérielles en libérant des ressources par la suppression de machines virtuelles inutiles ou en les redimensionnant. Le second, optimiser le fonctionnement de l'infrastructure en détectant les machines sous-dimensionnées, surchargées ou ayant une activité différente des autres machines virtuelles de l'infrastructure.

Abstract
This thesis presents two contributions. The first contribution is the study of key performance indicators to monitor physical and virtual machines activity running on VMware and KVM hypervisors. This study highlights performance metrics and provides advanced analysis with the aim to prevent or detect abnormalities related to the four main resources of a datacenter: CPU, memory, disk and network. The second contribution relates to a tool for virtual machines with pre-determined and / or atypical behaviors detection. The detection of these virtual machines has several objectives. First, optimize the use of hardware resources by freeing up resources by removing unnecessary virtual machines or by resizing those oversized. Second, optimize infrastructure performance by detecting undersized or overworked virtual machines and those having an atypical activity.

Mots clés
Cloud Computing, virtualisation, centre de données, machine virtuelle, introspection, supervision, analyse, comportements, métriques de performance

Key Words
Cloud Computing, virtualization, datacenter, virtual machine, introspection, monitoring, analysis, virtual machine behaviour, key performance indicator

L'UNIVERSITÉ NANTES ANGERS LE MANS