الجمهـــوريـــة الجـزائـريـة الديـــمقراطيـة الشـعبية
République Algérienne Démocratique et Populaire
وزارة الـتـعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمـــي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة وهران
Université d'Oran

كلية العلوم الدقيقة والتطبيقية
Faculté des Sciences Exactes et Appliquées
قسم الإعلام الآلي
Département d'Informatique

MÉMOIRE DE FIN D'ETUDES

En vue de l'obtention du

DIPLÔME DE MASTER EN INFORMATIQUE

Option : *Théorie et Pratique des Grands Réseaux (TPGR)*

Présenté et soutenu le *18/06/2014* **par :**

YAGOUBI Djamel Edine

Gestion de la consommation d'énergie dans le Cloud Computing.

JURY

Mlle M. Bourenane	Présidente
Mme F. Z. Bellounar	Examinatrice
Mr N. Lachachi	Examinateur

Encadré par :
Mr BELALEM Ghalem
Co-Encadré par :
Mlle DAD Djouhra

Promotion 2013/2014
Code Master : TPGR-13/2014

Dédicace

Je dédie ce modeste travail à mes très chers

Remerciments

En préambule à ce mémoire je remercie ALLAH qui m'aide et me donne la patience et le courage pour accomplir ce modeste travail.

Je tiens à exprimer mes sincères remerciements et toute mon gratitude à mon encadreur le Professeur G. Belalem et à mon co-encadreur Mlle D. Djouhra pour avoir accepté d'encadrer ce mémoire. J'aimerai leur adresser mes plus vifs remerciements pour tous leurs dynamismes, conseils, disponibilités, suivis, et leurs compétences scientifiques que nous avons pu apprécier tout au long de cette année.

Mes vifs remerciements vont aussi au jury, l'honorable Présidente Mlle M. Bourenane et les honorables membres Mme F. Z. Bellounar et Mr N. Lachachi pour l'intérêt qu'ils m'ont porté à mes recherches, en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

Je tiens aussi à remercier les enseignants qui m'ont aidé et soutenu durant toutes mes éducations.

Mes derniers remerciements vont à ma famille et notamment à mes parents, mes amis, car sans leurs constants soutiens et leurs encouragements, je n'aurai pas pu mener à bien ce mémoire.

Merci infiniment encore à tous.

الملخّص

أصبحت الحوسبة السحابية في السنوات الأخيرة النموذج المهيمن في ساحة تكنولوجيا المعلومات. مبدأها هو تقديم خدمات لامركزية، مما يسمح للعملاء على دفع ما يحتاجون إليه فقط. الطلب المتزايد على هذا النوع من الخدمات يجلب مقدمي خدمة السحابة لزيادة حجم بنيتها التحتية إلى حد أن استهلاك الطاقة والتكاليف المرتبطة بها أصبح أمرا بالغ الأهمية.
يجب على مزود الخدمة السحابية تلبية مجموعة من مطالب المختلفة. لهذا السبب نحن مهتمون في هذه الوثيقة على إدارة استهلاك الطاقة في مجال الحوسبة السحابية. وقمنا بتقديم نهجنا الذي يقلل من استهلاك الطاقة وعدد انتهاكات اتفاقية مستوى الخدمة. ونتيجة لذلك تقليل من انبعاث الحراري.

الكلمات المفاتيح: الحوسبة السحابية، مركز البيانات، الاستثمار، آلة افتراضية، هجرة، الطاقة.

Résumé

L'informatique dans les nuages (Cloud Computing) est devenu durant les dernières années un paradigme dominant dans le paysage informatique. Son principe est de fournir des services décentralisés à la demande et de permettre aux clients de payer uniquement les services dont ils auront besoin. La demande croissante pour ce type de service amène les fournisseurs de service Clouds à augmenter la taille de leurs infrastructures à tel point que les consommations d'énergie ainsi que les coûts associés deviennent très importants.
Chaque fournisseur de service cloud doit répondre à des demandes différentes. Les gestionnaires d'infrastructure doivent héberger un ensemble de ces services. C'est pourquoi au cours de ce travail, nous nous sommes intéressés à la gestion de la consommation d'énergie dans le Cloud Computing, où nous avons présenté nos approches qui permettent de réduire la consommation d'énergie et le nombre de violations de SLA et par conséquent minimiser le dégagement de chaleur.

Mots clés: Cloud computing, Data center, Placement, Virtual machine, Migration, énergie.

Abstract

Cloud computing has become over the last years an important paradigm in the computing landscape. Its principle is to provide decentralized services and allows client to consume resources on a pay-as-you-go model. The growing demand for this type of service brings providers service clouds to increase the size of their infrastructure to the point that the energy consumption and associated costs become very important
Each cloud service provider has to provide to different types of requests. Infrastructure managers must accommodate a range of these services. That is why in this memory we are interested in the management of energy consumption in cloud computing, where we presented our approaches which allow to reduce energy consumption and the number of SLA violations and thus minimize the exotherm..

Keywords: Cloud computing, Data center, Placing, Virtual machine, migration, Energy.

Introduction générale

L'informatique dans les nuages (cloud computing en anglais) s'est imposée ces dernières années comme un paradigme majeur d'utilisation des infrastructures informatiques. Celui-ci répond à des besoins et demandes croissantes en terme de disponibilité et flexibilité. Le développement remarquable du Cloud Computing, ces dernières années, suscite de plus en plus l'intérêt des différents utilisateurs de l'Internet et de l'informatique qui cherchent à profiter au mieux des services et des applications disponibles en ligne à travers le Web en mode services à la demande et facturation à l'usage.

L'approche du Cloud Computing s'appuie principalement sur le concept de virtualisation. Ce concept est un ensemble de techniques permettant de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, isolés les uns des autres. Un Cloud est constitué d'un ensemble de machines virtuelles qui utilisent la même infrastructure physique.

La croissance des deux côtés de l'offre et de la demande rend le problème de la consommation d'énergie plus complexe, sophistiqué dans un environnement Cloud. Les demandes cumulées sur une seule machine virtuelle ou en général sur un seul Data Center mènent à la saturation. Ces derniers ne pourront plus satisfaire les demandes des utilisateurs. D'autre part les besoins en ressources des applications peuvent aller au-delà de ce qui est disponible dans un Cloud.

La consommation électrique des centres de données (data centers en anglais) dans le monde a été estimée pour 2010 entre 1.1% et 1.5% de la consommation électrique globale. Même si cela parait peu en proportion, cette consommation d'énergie représente plus de 200 milliards de kilowatt-heure [Koo08]. Cette consommation d'énergie est en croissance permanente au fil des années. Elle était estimée à 12% par an en 2007 par l'U.S Environmental Protection Agency

pour atteindre un coût de l'électricité utilisée par les serveurs des data centers estimé de 7.4 milliards de dollars [EPA07].

Dans ce travail, l'objectif visé est de proposer une stratégie basée sur la migration des machines virtuelles, en appliquant des méthodes et des algorithmes. Notre politique de migration des machines virtuelles vise à optimiser la consommation d'énergie.

Organisation du mémoire

Le présent mémoire est structuré autour de quatre principaux chapitres qui se résument comme suit :

Chapitre 1 : Dans le premier chapitre, nous présenterons le problème de la consommation d'énergie des data center et son impact énergétique.

Chapitre 2 : Le second chapitre présentera quelques différentes techniques d'optimisations d'énergie et les principaux travaux de recherches qui ont été proposés dans la littérature.

Chapitre 3 : Le troisième chapitre sera réservé à la description détaillée de la conception de la stratégie utilisée ainsi que des services que nous avons proposés. Cette conception se fera à l'aide de formules, d'algorithmes et de diagrammes UML.

Chapitre 4 : Ce dernier chapitre présentera les étapes de l'implémentation de l'approche proposée. Nous y détaillerons la réalisation de certaines fonctionnalités ainsi que l'étude d'évaluation de cette stratégie. Les résultats d'expérimentation seront interprétés.

Enfin, un ensemble de perspectives viendra clore notre travail.

Table des matières

Chapitre 1

Consommations d'énergie des data centers

Sommaire

1.1. Introduction

Les énergies fossiles (pétrole, gaz, charbon) ont façonné le monde d'aujourd'hui. Elles sont à l'origine de la révolution industrielle et du développement des transports et ont modelé l'organisation sociale et économique actuelle de l'humanité. Les stocks de pétrole, de gaz et de charbon seront vides d'ici quelques décennies et ne se reconstitueront pas avant plusieurs centaines de millions d'années [WEB11]. Bon gré ou malgré, nous n'avons donc pas d'autres choix que de tendre vers plus de sobriété énergétique, le temps de trouver des alternatives viables aux énergies fossiles. D'autant que l'exploitation intensive de ces ressources carbonées a considérablement augmenté l'effet de serre ces 50 dernières années entraînant un dérèglement climatique global. Ce dérèglement met en danger la survie de l'humanité, notamment en accélérant l'écroulement de la biodiversité.

Les problèmes liés aux émissions de dioxyde de carbone et à l'énergie font de plus en plus souvent la une des journaux internationaux. Les gouvernements, les associations à but non lucratif et les entreprises réalisent désormais des enquêtes régulières afin d'analyser leur émissions du gaz CO2. Leur objectif est de mesurer l'incidence de leurs activités sur le réchauffement climatique et d'élaborer des plans d'action visant à réduire leurs émissions de dioxyde de carbone.

1.2. Définition des data centers

Le data center est un terme anglais qui est employé par les professionnels de l'informatique. Un data center est un bâtiment spécialement conçu pour héberger des équipements informatiques. Ces bâtiments sont découpés en plusieurs salles qu'on appelle des salles blanches. Les grandes entreprises disposent souvent de plusieurs data center pour héberger leurs serveurs. Cependant, de nombreux centre de données sont mutualisés et ils hébergent des serveurs de différents clients.

1.3. L'impact énergétique des data centers

L'alimentation électrique est un aspect stratégique pour un cloud data center : sans électricité en quantité suffisante, un data center ne peut pas fonctionner. C'est pour cette raison que les data centers sont construits dans des zones où les infrastructures électriques sont importantes.

La consommation d'énergie des salles serveurs ne cesse d'augmenter pour répondre aux besoins notamment des services web. La consommation électrique des data centers a doublé en 5 ans et les études montrent que la tendance s'accélère.

La direction informatique se retrouve ainsi au centre de nouveaux enjeux : enjeux financiers comme source de réduction des coûts, enjeux environnementaux en participant à la réduction des émissions de CO_2.

La part de la facture électrique représente 10% du coût de fonctionnement d'un centre de données. Elle pourrait s'élever très rapidement à 50% en absence des changements [WEBb].

Selon une étude IDC, pour 1$ investit dans le hardware, 1$ d'énergie sera dépensé [WEBb].

Ces centres qui regroupent les serveurs où sont stockées les données informatiques, sont de véritables gouffres énergétiques. Selon Dalkia [1], un data center de 10.000 m^2 consomme autant d'énergie qu'une ville de 50.000 habitants.

D'après une étude de l'Université de Standford, les quelques 500.000 data centers existants dans le monde consomment environ 30 milliards de watts d'électricité par an, soit l'équivalent de la production de 30 centrales nucléaires [GLA12].

L'énergie dépensée par les centres de stockage des données informatiques est equivalente à 1,5% de la consommation électrique mondiale. Les data centers seraient également responsables de 2% des émissions mondiales de CO_2, soit autant que le trafic aérien mondial.

1. Dalkia est une entreprise spécialisée dans les services énergétiques, filiale de Veolia Environnement à 66% et d'Électricité de France à 34%. Son siège social est situé à Saint-André-lez-Lille, en France.

Le poids de la consommation d'énergie des data centers est encore plus important dans les pays développés. selon France Culture [2], en France les data centers consomment 9% de l'électricité du pays. Les statistiques montrent que le premier data center construit par Facebook en Europe consomme à lui seul 1% de l'énergie suédoise.

Selon un rapport de l'EPA [3], les data centers implantés aux Etats-Unis ont consommé 61 milliards de kilowatt/heures d'électricité en 2006. Ce chiffre représente 1,5 % de toute l'électricité consommée aux Etats-Unis et un coût de 4,5 milliards de dollars. Les data centers ont été identifiés comme l'un des consommateurs d'énergie qui connaît la croissance la plus rapide. L'EPA exige que le secteur public développe des stratégies pour améliorer l'efficacité énergétique de ses data centers et a fixé un objectif d'amélioration de 20 % [AGE07]. Les data centers du secteur privé, quant à eux, pourraient bien être contraints de respecter sous peu des limites d'émissions de CO_2.

La croissance du secteur étant très importante en raison du développement d'Internet sur appareils mobiles et de l'augmentation des besoins de stockage des entreprises. La demande énergétique liée au data centers va continuer à augmenter fortement.

1.4. Pourquoi les data centers consomment autant d'énergie ?

La consommation d'énergie conséquente des data centers s'explique par plusieurs raisons. D'une par, les serveurs de stockage ne peuvent pratiquement jamais être arrêtés pour ne pas perdre les données confiées par les clients des data centers. Ils fonctionnent donc 24 heures sur 24, tous les jours de l'année.

De plus, ces serveurs sont des appareils électroniques de grande taille qui produisent énormément de chaleur. Leur bon fonctionnement est assuré par des

2. France Culture est la chaîne de radio culturelle nationale publique française du groupe Radio France.

3. Environmental Protection Agency, Agence américaine de protection de l'environnement [AGE07]

systèmes de climatisation très énergivores.

Enfin, pour ne pas perdre de données lors d'éventuelles coupures d'électricité, les data centers sont équipés de générateurs de secours fonctionnant au fioul (émetteurs de CO_2), voire de batteries, qui contribuent à accentuer leur dépense énergétique.

1.5. Pourquoi réduire la consommation ?

Suivant la courbe de progression de la performance de calcul [KBSW09], la consommation d'énergie des data centers n'a cessé de croître ces dernières années.

Ainsi, les clusters ont augmenté leur nombre de serveurs en même temps que leur puissance. Ce qui nous amène à faire des prédictions pour le futur de la consommation des data centers, qui sont du même ordre que le mur de la miniaturisation des composants. En effet, la miniaturisation croissante des processeurs a dû ralentir le fait de la dissipation de la chaleur insuffisante, et a amené les constructeurs à changer leurs architectures en augmentant le nombre de processeurs ou de cœurs. Si on prévoit la consommation électrique des superordinateurs du top500 [WEBc] en supposant qu'elle croît de façon constante, on arrivera à des clusters d'une consommation de plusieurs centaines de mégawatts ! ce qui nécessite d'avoir un réacteur nucléaire pour alimenter ces supercalculateurs.

Il est donc nécessaire de réduire cette consommation par des moyens permettant à la fois de réduire l'énergie demandée et de garder un niveau acceptable de performance. Eteindre systématiquement la moitié des serveurs entrainant la réduction de la consommation d'énergie, ça n'est évidemment pas une solution fiable. Ajoutons à cela l'impact environnemental des data center. Décentraliser les opérations locales réduit la consommation d'énergie et l'émission du gaz CO_2, cela en délocalisant les applications vers des serveurs plus efficaces énergétiquement. Selon un rapport de Greenpeace [WEB10], la consommation d'énergie globale des data centers était responsable en 2007 de 116 tonnes de dioxyde de

carbone (MtCO2). Ce même rapport estime que ces mêmes émissions pourraient doubler d'ici 2020 pour atteindre 257 $MtCO_2$.

La consommation d'énergie des serveurs n'est pas en effet la seule source d'émission du gaz CO_2. Il faut aussi prendre en compte le cycle de vie complet d'un serveur, depuis le moment où le serveur est construit jusqu'à son démantèlement. Un serveur sera responsable de plus d'émission du gaz CO2 que sa consommation d'électricité [WEB11].

Enfin, le coût d'électricité de l'exploitation d'un data center peut rapidement devenir important, jusqu'à ce que la consommation d'électricité sur plusieurs années revienne au même prix qu'acheter ce même data center.

Que ce soit donc pour des raisons économiques, environnementales ou tout simplement pratiques, il est intéressant, voire dans certains cas nécessaire, de réduire la consommation énergétique des clouds. Cependant, comme nous allons le voir par la suite, cela ne peut pas se faire de n'importe quelle façon.

1.6. Le Power Usage Effectiveness (PUE)

1.6.1. Le PUE : c'est quoi ?

Le PUE est un indicateur mis au point par le Green Grid pour mesurer l'efficacité énergétique d'un data center. Il est calculé en divisant le total de l'énergie consommée par le data center par le total de l'énergie utilisée par les équipements informatiques (serveur, stockage, réseau). En moyenne les data center français ont un PUE de 2,5 ce qui signifie que pour 1 Watt consommé par l'équipement informatique, il en faut 2,5 Watt à l'entrée du data center.
Sans être un indicateur suffisant pour déterminer l'efficience d'un data center, il n'en demeure pas moins qu'un indicateur universellement reconnu [WEBa].

1.6.2. Faut-il choisir entre disponibilité du data center et PUE ?

Les équipements informatiques sont hébergés dans un data center disposant d'une infrastructure complexe (onduleurs, batteries, groupes électrogènes , climatisations, ...) destinée à maximiser le taux de disponibilité des équipements qu'il héberge. La norme TIER définie par l'UPTIME Institute [4] classifie les data centers de 1 à 4. Les data centers de type TIER 1 disposent d'une seule chaîne électrique alors que les data centers de type TIER 4 disposent de 2 chaînes électriques en redondance totale. Par conséquent, ces derniers assurent une disponibilité statistique de 99,995%, soit une indisponibilité de 0,5 heure par an contre 99,671% et 28,8 heures d'indisponibilité statistique annuelle pour ceux de type TIER 1 [WEBa].

A l'heure ou le fonctionnement de chaque entreprise est de plus en plus dépendant de l'informatique, la tendance actuelle est de privilégier des data centers avec le niveau de TIER le plus élevé. Sans rentrer dans le débat de la légitimité de cet indice, on constatera tout au moins que l'on attend un taux de disponibilité de plus en plus important [WEBa].

Par voie de conséquence, ce type de data center suppose une infrastructure plus importante qui augmente logiquement le PUE. Ceci est d'autant plus vrai qu'une architecture classique de type TIER 4 impliquant de répartir la charge entre les sources de production électrique. Ceci conduit les onduleurs à être chargés autour de 40% et donc moins efficients que dans leur plage de fonctionnement de prédilection située autour de 80%. Ces derniers ayant un moins bon rendement, ils contribuent à dégrader encore le PUE.

Il est donc plus facile d'avoir un bon PUE avec un faible niveau de TIER. On voit bien que cet indicateur n'est pas moins insuffisant malgré sa pertinence d'un point de vue écologique [WEBa].

La Figure 1.1, illustre La consommation électrique dans les data centers.

4. The Uptime Institute, association américaine, a défini en 1995 une norme permettant d'analyser les topologies mises en œuvre et de délivrer une certification. Cette norme décrit quatre topologies d'infrastructure de sites dont les niveaux croissants d'éléments et de chemins de distribution redondants servent de base de comparaison.

FIGURE 1.1 – *La consommation électrique dans les data centers*

1.7. CONSOMMATIONS D'ÉNERGIE AU NIVEAU DU SYSTÈMES DE REFROIDISSEMENT

Comme c'est illustrée dans la Figure 1.1, le refroidissement représente jusqu'à 50% de la consommation d'énergie totale des data centers et des salles serveurs. Il est donc essentiel de concevoir un refroidissement efficace pour les sites informatiques quelle que soit leur taille.

L'électricité consommée par les équipements informatiques est presque entièrement transformée en chaleur par effet *Joule*. Pour tenir à température constante le matériel, un système de refroidissement est nécessaire. Sur un ordinateur personnel, ce rôle est tenu par un ou deux ventilateurs. Dans une salle contenant plusieurs centaines d'équipements informatiques, l'installation d'une climatisation est nécessaire. Les nouveaux data centers déclarent souvent qu'ils utilisent des sources froides naturelles, généralement de l'air, en complément de la climatisation.

Il est possible de récupérer et de valoriser la chaleur produite par les climatisations. Cette chaleur pourrait être valorisée dans des réseaux de chaleurs

locaux dans une logique de filière énergétique locale. Pourtant, la valorisation de la chaleur dans un réseau de chauffage urbain est rarement mise en œuvre lors de la construction des data centers

1.8. Consommations d'énergie au niveau des serveur

Les serveurs sont les équipements les plus énergétiques dans les data centers, ils consomment environ 40% d'énergie totale des Data Centers. Ils constituent donc l'une des cibles prioritaires pour la mise en oeuvre des mesures d'économie d'énergie [SBW+].

Pour traiter les données au sein de certains Data Centres, les serveurs n'utilisent que 6% à 12% d'énergie qu'ils consomment [GLA12]. Un serveur qui n'utilise que 20% de sa capacité de calcul utilise déjà 70% de sa puissance électrique maximale. Par ailleurs, une pièce jointe liée à un ancien mail de trois ans doit être stockée sur un serveur qui consomme de l'énergie pour fonctionner même si on n'utilise plus la pièce jointe.

24,7 milliards de dollars sont dépensés chaque année pour le matériel, la maintenance, la gestion, l'alimentation énergétique et le refroidissement des serveurs non utilisés. Cela représente environ le coût des 13 années du programme Apollo [5][Kar09].

L'énergie consommée pour faire fonctionner ces serveurs non utilisés compense les émissions du gaz CO_2 de 6,5 millions de véhicules.

Un serveur à haute efficacité énergétique comporte des composants performants d'un point de vue énergétique, tels que les blocs d'alimentation, les processeurs, les disques durs, la mémoire et les connecteurs d'extension. À cette fin, des cartes mères pour serveur, plus efficaces que celles des générations précédentes sont également en cours de développement. L'idée générale est de minimiser les pertes liées à la transformation de courant sur la carte mère. Ces pertes sont induites par l'écart entre la puissance d'entrée (sortie du bloc d'alimentation) et la puissance au niveau de chacun des composants [SBW+].

5. Coûts du programme Apollo : 25,4 milliards de dollars.

Composant	Puissance max (W)
Processeur	80
Mémoire vive	36
Disques	12
Connecteurs d'extension	50
Carte mère	25
Ventilateurs	10
Alimentation	38

Table 1.1 – *Exemples de puissance absorbée par différents composants*

Le tableau 1.1 indique la consommation énergétique approximative de différents composants matériels et montre que les processeurs, les blocs d'alimentation, la mémoire vive et les bus de connecteurs d'extension sont ceux qui absorbent le plus de puissance.

1.9. Conclusion

La problématique est donc la suivante : comment réduire la consommation d'énergie des serveurs sans pour autant trop impacter la qualité de service et les performances des applications qui s'exécutent sur le cloud ? Il existe de nombreux moyens pour réduire la facture d'électricité, et un bon nombre de techniques peuvent se faire sans impacter la qualité de service. Cependant, en acceptant une dégradation faible de performance, il est possible de réduire encore plus la puissance consommée par l'infrastructure.

Dans le prochain chapitre, nous allons présenté quelques techniques nécessaire pour réduire la consommation d'énergie dans les Data centers des Clouds.

Chapitre 2

Comment réduire l'impact énergétique des data centers ?

Sommaire

2.1. Introduction

Comme tout autre système distribué, les datas centers des clouds font face à une demande croissante en énergie. Cette énergie ne provoque pas seulement la diminution des bénéfices des fournisseurs mais aussi émette une grande quantité de dioxyde de carbone. Afin d'agir sur cette consommation, les travaux de recherches dans la littérature présentent un ensemble de techniques bien différents pour résoudre ce problème. Ces moyens sont à différentes échelles mais chacun a un impact sur cette consommation et des effets souvent négatifs sur les performances du système. Ces techniques sont appliquées soit à un niveau bas comme le processeur, soit à un niveau plus élevé de cluster ou d'infrastructure au niveau machine. Ces techniques peuvent être aussi bien logiciels comme la virtualisation que matériels comme le DVFS (voir la section 2.2).

Nous allons présenter dans ce chapitre quelques différentes techniques d'optimisations d'énergie et les principaux travaux de recherches qui ont été proposés dans la littérature.

2.2. Dynamic Voltage and Frequency Scaling

Le DVFS est un mécanisme mis en place par les constructeurs de puces. Il permet de réduire ou d'augmenter dynamiquement la fréquence d'un composant ainsi que son voltage. Cela permet de diminuer la fréquence d'un processeur lorsqu'il est sous utilisé, ou au contraire, d'augmenter sa fréquence au dessus de sa fréquence maximale lorsqu'il a besoin d'être plus performant. Ce mécanisme se fait en réglant un couple de voltage/fréquence (appelés "Performance state" PState) dont les valeurs dépendront du processeur utilisé.

Les différents couples voltage/fréquence sont gérés par le système de la machine. C'est le cas dans les systèmes où sont implémentés des gestionnaires de fréquence CPU qu'on appelle gouverneur. Le but de ces gouverneurs est de gérer la fréquence et le voltage du processeur en fonction d'une orientation

définie par l'utilisateur et en fonction de certains paramètres systèmes. Trois gouverneurs différents sont implémentés par défaut : **performance**, **powersave** et **ondemand**. Le gouverneur **performance** réglera la fréquence au maximum, le **powersave** règlera la fréquence au minimum. Le gouverneur **ondemand** quand à lui règlera la fréquence au maximum dès l'arrivée des tâches à effectuer, il réduit petit à petit cette fréquence quand la charge baisse.

Néanmoins, toute action a un coût et le DVFS n'y échappe pas et peut ralentir une application. Changer la fréquence prend du temps (de l'ordre de dizaine de microsecondes [BB00]). Ce temps passe sans calcul. Par conséquent, des changements courants de fréquence vont avoir un impact très négatif sur les performances du système. Cet impact induira évidemment à un mauvais temps d'exécution de l'application (si cette application termine), ce qui donne une mauvaise consommation d'énergie. Il est important de noter qu'un autre inconvénient de la technique DVFS existe. Il peut arriver qu'une application consomme plus en tournant à une fréquence plus basse. Prenons par exemple une application qui se termine en 10 secondes avec un processeur qui consomme 100 W à une fréquence maximale. Nous aurons donc une consommation d'énergie de 1000 Joules. Admettons maintenant que cette même application consomme une fréquence minimale de 50W mais prend un temps de calcul de 30 secondes. Nous aurons alors une consommation d'énergie de l'application de 1500 Joules. Par contre, si cette application se termine en 15 secondes à fréquence minimale, nous aurons une consommation de 750 Joules.

Une vitesse d'exécution réduite n'implique pas forcément une réduction de la consommation d'énergie. Ce qui signifie que la réduction de la puissance n'implique pas non plus une réduction de la consommation d'énergie. Un bon exemple est décrit dans [SRWM10], où les auteurs présentent et implémentent un algorithme de scheduling basé sur des événements pour le gouverneur de processeur. Pour ce faire, les auteurs ont implémenté un scheduler qui reçoit des événements en fonction de la charge de travail. Ainsi, si la charge était basse au dessus d'un certain seuil, l'événement "chargé" va être envoyé et le gouverneur va ainsi changer la fréquence du processeur.

La comparaison est faite notamment avec les gouverneurs implémentés dans

les noyaux Linux qui sont : performance (avec une fréquence maximale) et powersave (avec une fréquence minimale). On remarque dans les résultats présentés par les auteurs que le gouverneur performance exécute le programme en 1h03 avec une puissance de 310W, cela résulte une énergie de 329 Wh. On remarque aussi que le gouverneur powersave exécute ce même programme en 3h09 à une puissance de 284W résultant une énergie de 895Wh. L'algorithme proposé dans cet article exécute ce même programme en 1h05 avec une puissance moyenne de 298W avec une énergie de 324Wh. Cela montre bien qu'il est nécessaire de bien utiliser la gestion de fréquence des processeurs. Une fréquence plus faible n'économisera pas forcément de l'énergie et le bon choix sera parfois de favoriser la performance pour économiser l'énergie consommée avec un temps d'exécution plus court.

Les auteurs dans [HF09] tentent de résoudre le problème d'affectation de la fréquence la plus efficace en énergie en utilisant la charge de travail par intervalle. Cela est fait pour allouer la bonne fréquence et le bon voltage aux tâches. Ils ont proposé un gouverneur visant à réduire la consommation d'énergie en minimisant les pertes de performance, acceptant ainsi des pertes pour avoir une meilleure économie d'énergie. Pour cela, les auteurs caractérisent la charge de travail pour chaque nœud. Ils définissent ensuite le gouverneur efficace en énergie ecod basé sur les moments où le CPU attend le résultat d'activités hors processeur. Cela caractérise la charge en se basant sur des mesures sans avoir besoin d'aucune connaissance des applications qui s'exécutent sur le cluster. L'algorithme est ensuite évalué par l'exécution des benchmarks NAS Parallel Benchmark [BBB+91]. Les auteurs montrent qu'une économie d'énergie allant jusqu'à 11% peut être effectuée par rapport au gouverneur **ondemand** avec une faible perte de performance (5.1% d'augmentation du temps de calcul contre 7.9% pour **ondemand**).

Un autre inconvénient de taille pour le DVFS est l'impact de la fréquence du DVFS sur le fonctionnement du processeur. En effet, des études montrent que l'utilisation du DVFS a un impact négatif sur la fiabilité du processeur [ZMM04]. Les auteurs montrent analytiquement que réduire la fréquence à voltage fixe tend à réduire la performance (c'est-à-dire la probabilité de finir une application

correctement avant échéance lorsqu'il y a des erreurs). De plus, réduire le voltage à des fréquences réduites augmente le taux d'erreurs dans les processeurs.

L'approche de type DVFS n'est pas seulement limitée aux processeurs. C'est une approche qui peut être appliquée aussi à d'autres types de matériels. Par exemple, nous pouvons utiliser une approche similaire au niveau des cartes réseaux. Celle-ci s'appelle l'Adaptative Link Rate (ALR) [GCNS08]. Le principe est le même : augmenter ou baisser la vitesse de la carte (son débit, faire fonctionner une carte Ethernet 1Gbps à 100Mbps ou moins) en fonction des besoins afin de limiter la consommation d'énergie.

2.3. Allumage et extinction des machines

Il existe un moyen simple pour réduire la consommation d'énergie d'une machine. Il suffit de l'éteindre ! Par éteindre, nous entendons une extinction totale de la machine pour amener sa consommation à 0W. Cela est possible par des appareils permettant de "désactiver" les prises de courant [KOWN10]. Il faut bien faire la distinction entre une extinction et une mise en hibernation. Ce sont deux états similaires mais qui ne consomment pas la même puissance. L'extinction de la machine désigne l'arrêt total de tous ses composants en réduisant la consommation de puissance à un niveau proche de 0W. Nous désignerons par hibernation l'arrêt du système à l'exception de certains composants utilisés pour la mise en veille du système. Comme exemple d'hibernation, nous citons la carte réseau par l'utilisation du *wake-on-lan* (technique consistant à réveiller une machine à distance par le réseau), le *suspend-to-RAM* ou le *suspend-to-Disk*.

La Figure 2.1 est un exemple de diagramme des différents états d'une machine Linux. Elle présente les coûts en temps et en puissance électrique nécessaire pour passer d'un état à un autre. Les valeurs présentées sont tirées de [DSI10].

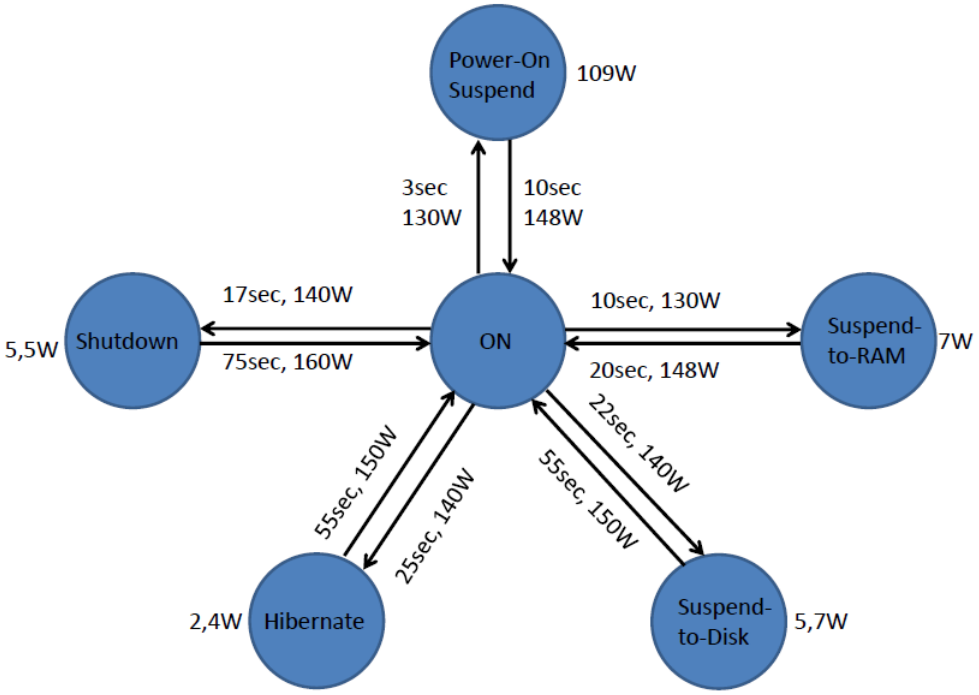

FIGURE 2.1 – *Exemple de transition entre états pour une machine Linux*

Ainsi, il est possible de développer des approches utilisant cette technique afin de réduire la consommation d'énergie. Dans [NCS09], une approche utilisant un processus de décision de Markov est utilisée. Cette approche permet de gérer les tâches, l'état des hôtes (allumé ou éteint) et elle permet aussi d'atteindre l'état optimal pour la consommation d'énergie tout en préservant la qualité de service (temps d'attente des tâches).

Cependant, pour réduire la consommation énergétique d'un serveur à 0W, nous devons prendre en compte plusieurs facteurs.

Premièrement, nous devons être capables d'éteindre et de rallumer à distance les machines. Mais dans les clusters actuels, nous n'avons pas tout le temps cette capacité car les machines ne sont pas faites pour être éteintes. Deuxièmement, nous devons prendre en compte le temps pris pour éteindre et rallumer les différentes machines. Ce temps est équivalent au temps pendant lequel les machines ne soient pas utilisées et consommeront de l'énergie. Il faut donc bien

prendre en compte le temps d'allumage pour ne pas en avoir besoin en temps réel. Cela induit à une dégradation dans la qualité de service de la tâche. Enfin, il faut prendre en compte le coût induit par la consommation des ressources lors de l'allumage et l'extinction. C'est le cas lorsqu'on allume une machine. En plus du démarrage normal du système d'exploitation, cette machine effectue une série de tests matériels et logiciels. Cela prend donc du temps mais aussi coûte de l'énergie dans la mesure où la machine fait des calcule. Ces temps et consommations d'énergie varient bien sûr en fonction de la machine, du système et des vérifications matérielles qui doivent être faites. Pour le temps d'allumage, cela peut aller jusqu'à la vingtaine de secondes dans [LO10] à un peu plus d'une minute comme dans [DSI10], voire même de plusieurs dizaines de minutes comme dans [HHN08].

Toutefois, un effet indésirable peut apparaître lorsqu'on sollicite trop l'allumage et l'extinction de matériels. Cela peut causer une augmentation du taux de malfonctionnement de la machine. Bien que rien n'indique qu'au niveau machine le taux de malfonctionnement des hôtes ait un lien avec le nombre de démarrage/extinction [HHN08]. Ce lien peut exister de la même manière que pour le processeur et le DVFS comme c'est vu dans la section précédente [ZMM04].

L'allumage et l'extinction des hôtes physiques est donc une technique importante et extrêmement puissante quand il s'agit de la réduction de la consommation d'énergie d'un système informatique. Ce moyen est à utiliser avec précaution dans la mesure ou une mauvaise prédiction peut causer des allumages/extinctions très fréquents impliquant ainsi un gaspillage de ressources et d'électricité. Il faut toujours que l'énergie économisée pendant qu'une machine reste éteinte soit plus importante que celle consommée pendant l'allumage et l'extinction. C'est à dire augmenter la consommation d'énergie tout en dégradant la qualité de service.

2.4. Allocation

L'allocation des tâches est aussi un moyen pour gérer la consommation d'énergie d'un système. En plaçant de façon intelligente les tâches, nous pouvons utiliser les infrastructures les moins gourmandes en électricité.

C'est le cas des techniques dites de consolidation. Le but étant de regrouper la charge de travail au même endroit pour pouvoir réduire la consommation d'énergie. Consolider les tâches sur un nombre réduit de machines peut avoir de graves effets. Outre le fait qu'on puisse tenter de consolider sans succès, il faut aussi prendre en compte les interactions que peuvent avoir les tâches entre elles lorsqu'elles s'exécutent sur une même machine. Dans [SKZ08], les auteurs montrent qu'en présentant la consommation de ressources d'une tâche, la consommation de ressources et l'énergie consommée par chaque transaction augmentent comparativement lorsqu'elle est seule. De plus, l'efficacité énergétique d'une transaction tend à augmenter aussi si la tâche s'exécute toute seule sur la machine. Les auteurs montrent ensuite qu'on peut trouver une zone optimale à laquelle chaque tâche s'exécutant sur la machine consommera le moins d'énergie. Ils proposent ensuite un algorithme de consolidation basé sur le profil des interactions des tâches pour déterminer la consolidation optimale ou proche de l'optimale, ainsi que pour caractériser quels types de charges de travail doivent être combinées.

L'allocation peut aussi être utilisée en tant que technique dans la mesure où une bonne caractérisation des différentes ressources matérielles peut faire une différence importante. Tous les serveurs ne consomment pas la même puissance pour exécuter la même application et donc l'utilisation de l'allocation des tâches s'avère donc importante. Différentes techniques peuvent être observées pour prendre en compte les différentes consommations à différents niveau, que ce soit au niveau du processeur ou au niveau de la machine elle-même.

C'est le cas par exemple de Dupont et al. dans [DGH+], où les auteurs utilisent l'allocation et la réallocation des machines virtuelles pour mettre en oeuvre un framework flexible et efficace en énergie (ensemble de composants logiciels). Pour ce faire, les auteurs se basent sur la gestion des clusters par la programmation par

contraintes sur Entropy [HLM+09] afin d'être flexible en exprimant les différents objectifs pour les algorithmes. Cela donne la possibilité d'ajouter ou d'enlever des contraintes pour exprimer les contraintes SLA sans changer les algorithmes. Les auteurs valident ensuite leur framework par simulation et expérimentations. Ils montrent que dans un environnement cloud de test, leur approche permet de réduire jusqu'à 18% d'émissions CO_2.

2.5. Virtualisation

Au fur et à mesure que les systèmes deviennent de plus en plus complexes et les architectures de plus en plus diverses, il faut rendre transparent certains services. Il devient ainsi nécessaire d'avoir un accès commun à des ressources différentes pour faciliter le développement. C'est ici qu'intervient ce qu'on appelle la technique de virtualisation. Cette technique permet de rendre l'accès identique à plusieurs architectures. Par exemple, on peut avoir un accès virtualisé à des processeurs différents quel que soit leur constructeur ou leur méthode d'accès. La virtualisation peut être appliquée à plusieurs niveaux. Nous pouvons avoir la virtualisation du stockage où l'accès aux ressources de stockage sera transparent quel que soit l'endroit où se trouvent effectivement les disques durs et quel que soit leur type (type normal, SSD, mémoire flash...) [SKM08].

Nous avons aussi la virtualisation des machines virtuelles (VMs). Cela consiste à donner un accès complet à une infrastructure virtuelle qui apparait à l'utilisateur comme une machine complète (voir Figure 2.2). Cette technologie sert à donner à deux clients différents un accès chacun à une machine complète sur une même machine physique (PM). Cela permet d'avoir une meilleure sécurité dans le sens où même si on donne un accès administrateur à un utilisateur sur une machine virtuelle, ce dernier ne pourra pas compromettre l'intégrité physique de la machine hébergeant la machine virtuelle [BDF+03].

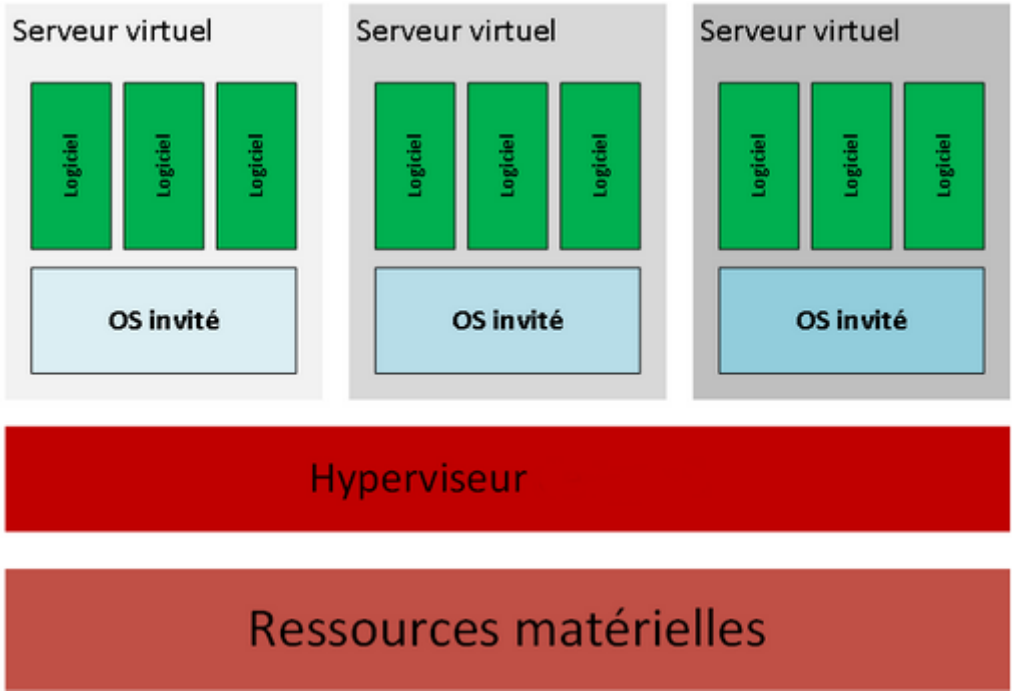

FIGURE 2.2 – *Diagramme virtualisation avec un hyperviseur*

Les économies d'énergie réalisables grâce à la virtualisation sont de l'ordre de plus de 70% à 80%. Les économies réelles prélevées des projets de virtualisation dépendent du degré de virtualisation ainsi que du niveau de déploiement des nouvelles applications et fonctionnalités[WEB13].

Les machines virtuelles ont l'avantage de permettre une séparation logique des ressources de la machine. Par exemple, on peut allouer deux machines virtuelles sur une même hôte en allouant par exemple 70% du CPU à la première machine virtuelle et les 30% restant à l'autre. Il apparaît donc que la gestion des tâches est facilitée par les technologies de virtualisation, notamment au niveau du pro-visionnement des tâches. Dans [HSN10], les auteurs utilisent le fait que les machines virtuelles ne consomment pas tout ce qui leur est attribué pour sur-provisionner les machines physiques. Cela pour maximiser leur utilisation effective et économiser 20% de ressources.

La technique de virtualisation possède quelques inconvénients. Elle cause une baisse de performances. La gestion transparente des ressources est gérée

par un intergiciel qui engendre des coûts de calculs et de gestion des ressources. Ainsi, un des inconvénients majeurs de la virtualisation est le surcoût associé à la virtualisation [Dre08]. Le surcoût est notamment dû au fait que d'une part l'accès aux ressources matérielles (GPU, disques durs, etc) est plus lent que l'intergiciel de virtualisation. Cet intergiciel doit faire le lien entre le système hôte et la machine physique. Il est aussi dû au fait que les défauts de la page mémoire sont significativement plus coûteux. Ce surcoût dépend d'un certain nombre de paramètres tels que : le programme et l'architecture matérielle qui est virtualisée. Et peut atteindre des valeurs de ralentissement lorsque le cache est trop petit (17% pour une architecture Intel et 38% pour une architecture AMD) [Dre08].

La virtualisation offre une fonctionnalité nommée migration qui permet de migrer des machines virtuelles d'un nœud physique à un autre. Le but de cette technique est d'améliorer le temps de réponse et mieux gérer la consommation d'énergie.

2.6. Migration des machines virtuelles

Les technologies actuelles de virtualisation permettent de migrer une machine virtuelle d'une machine physique à une autre. Cela peut se faire de deux manières. La première est une méthode qui consiste à mettre en pause la machine, la migrer et la reprendre une fois la migration terminée. La deuxième manière permet une migration dite à chaud. Il est possible de migrer une machine virtuelle presque sans perturber son fonctionnement, c'est-à-dire sans arrêter son fonctionnement mais avec une interruption de service minimale. Elle peut aller de 60ms pour un serveur de jeu à faible latence à 210ms pour un serveur avec un workload de type SPECweb99 [WEB00]. Ce type de migration permet d'avoir une continuité du service avec juste une perturbation minimale au niveau réseau lors de la dernière partie de la migration [CFH+05].

Il existe plusieurs implémentations de la migration "live" de machines virtuelles. L'approche traditionnelle qui est implémentée dans Xen ou KVM par exemple

est dite en pré-copie. Le processus de migration commence par une phase de pré-copie de la mémoire de travail d'une machine virtuelle, puis va itérativement transférer à nouveau les pages qui ont été utilisées par la machine virtuelle pendant le transfert précédent. Ceci jusqu'à l'obtention d'un noyau de pages mémoire suffisamment petit ou jusqu'à ce qu'un nombre prédéfini d'itération ait été effectué. La machine virtuelle est ensuite arrêtée et le noyau de mémoire final restant à transférer est envoyé à la machine virtuelle destination qui est démarrée à son tour marquant la fin de la migration.

Enfin, cette migration possède un coût non négligeable, qui entraîne une consommation de ressources sur l'hôte source et destination à la fois, ce qui augmentera la consommation d'énergie pendant la migration. De plus, les dernières phases de copie de la mémoire lors de la migration, juste avant de terminer celle-ci peuvent entraîner sur certaines machines une dégradation de la qualité de service. Par exemple, dans [VBVB09], les auteurs étudient le coût de la migration dans les clouds, avec des applications de type services web. Il apparaît que lors de la migration, le temps de réponse peut dans certains cas monter jusqu'à 3 secondes, induisant ainsi des violations de SLA[1].

Le coût de la migration peut être pris en compte dans la prise de décision afin de ne pas faire de ré-allocations de machines virtuelles et par conséquence ne pas perdre les performances. C'est le cas de l'architecture de gestion de machines virtuelles de GreenCloud proposé dans [LWL+09].

Il existe plusieurs techniques de migration des machines virtuelles, nous pouvons citer : la politique "Single Threshold" et la politique "Double Threshold".

2.6.1. La politique "Single Threshold" (ST)

La politique ST est basée sur l'idée de créer un seuil d'utilisation supérieur et de garder l'utilisation totale du CPU de toutes les machines physique en dessous de ce seuil.

1. Le service level agreement (SLA) est un document qui définit la qualité de service requise entre un prestataire et un client.

2.6.2. La politique "Double Threshold"

L'idée est de fixer deux seuils, un seuil supérieur et un seuil inférieur et de garder l'utilisation totale du CPU de toutes les machines physiques entre ces seuils. L'objective de cette politique est de réduire l'utilisation du processeur en dessous du seuil d'utilisation supérieur si le seuil supérieur est violé. Sinon si le seuil inférieur est violé, toutes les machines virtuelles doivent être migrées de cette machine physique et la machine physique doit être mise en état hors tension (etteinte). Nous allons définir maintenant les trois politiques basées sur la politique "Double Threshold".

2.6.2.1. La politique "Minimization of migrations" (MM)

La politique MM sélectionne le nombre minimum de machines virtuelles nécessaires pour une migration.

2.6.2.2. La politique "highest potential growth" (HPG)

Lorsque le seuil supérieur est violée, HPG migre les machines virtuelles qui ont la plus faible utilisation de la CPU par rapport à la capacité de CPU définie par les paramètres des VMs. Cela pour minimiser l'augmentation potentielle de l'utilisation de l'hôte et et pour empêcher une violation SLA.

2.6.2.3. La politique "The random choice" (RC)

RC repose sur une sélection aléatoire d'un certain nombre de machines virtuelles nécessaire pour diminuer l'utilisation du processeur en dessous d'un seuil d'utilisation supérieur.

2.7. Conclusion

Dans ce chapitre, nous avons présenté quelques techniques d'optimisation d'énergie des data centers, en utilisant certains concepts tels que la virtualisation, la migration et la technique DVFS. Certaines techniques sont au niveau composant et d'autres sont au niveau infrastructure. Il convient donc de choisir les bonnes techniques pour arriver aux meilleurs résultats d'économies d'énergie et une bonne optimisation. Cela avec une perte minimale de performance et de qualité de service.

Le chapitre prochain consiste à implémenter une nouvelle stratégie de migration des machines virtuelles, Et cela dans le but d'améliorer certaines métriques de performance telle que la consommation d'énergie.

Chapitre 3

Conception

Sommaire

3.1. Introduction

Dans les chapitres précédents, nous avons exploré quelques techniques qui ont été proposées dans la littérature et qui ont un impact sur la consommation d'énergie.

Notre objectif principal est de proposer et d'implémenter une nouvelle stratégie de migration des machines virtuelles, Et cela dans le but d'améliorer certaines métriques de performance telle que la consommation d'énergie sans dégrader les contraintes du contrat SLA.

Le présent chapitre permet d'expliquer notre démarche en décrivant ses différentes phases, les algorithmes nécessaires pour son fonctionnement ainsi que les différentes étapes formalisées à l'aide du langage UML.

3.2. Approche proposée

L'objectif de ce travail étant de proposer une stratégie basée sur la migration des machines virtuelles. Cette proposition permet de sélectionner le nombre minimum des machines virtuelles nécessaire à faire migrer à partir d'une machine physique. L'avantage de cette stratégie est qu'elle permet d'augmenter considérablement le taux d'utilisation et de réduire la demande en énergie. L'idée de base est de concentrer les traitements de données sur une machine physique et d'éteindre les autres.

3.3. Architecture du système

Dans cette partie, nous allons décrire le fonctionnement de notre modèle tout en formalisant et en utilisant une architecture qui sera la base de notre approche. Comme c'est illustrée dans la figure 3.1, cette architecture est décrite par l'ensemble des composants suivants :

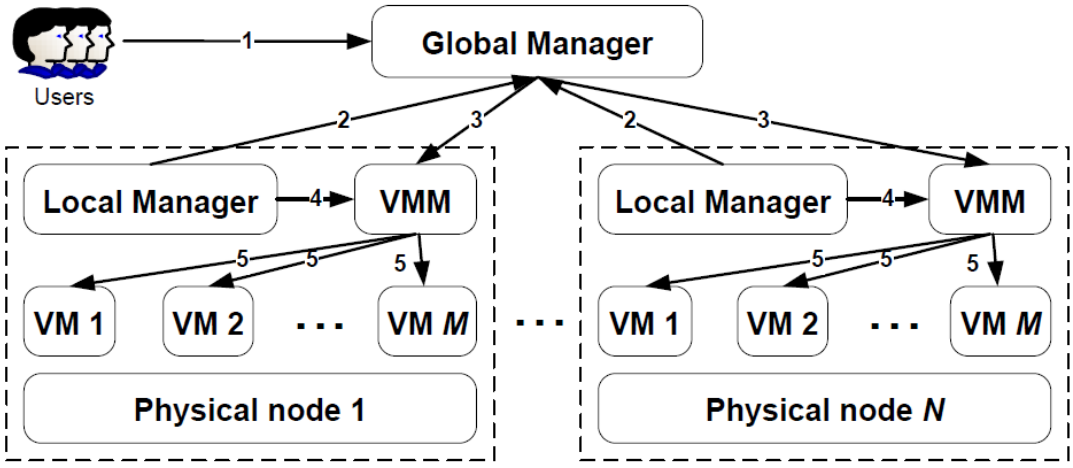

FIGURE 3.1 – *Architecture du système*

I) **Physical node :** Une machine physique est caractérisée par le nombre des VMs, la capacité, le nombre des requêtes arrivées, la vitesse du CPU et nombre de cores.

II) **VMs :** Une machine virtuelle consiste à créer plusieurs environnements d'exécution sur une seule machine physique. Elle fournit à chaque utilisateur un service selon la demande.

III) **VM Manager (VMM) :** Assure le suivi de la disponibilité des machines virtuelles et de leur utilisation des ressources. Il est chargé du provisionnement des nouvelles machines virtuelles ainsi que la réaffectation des VMs d'une machine physique à une autre afin d'adapter le placement.

IV) **Local managers :** Les gestionnaires locaux se trouvent sur chaque noeud comme un module du VMM. Leur objectif est le suivi continu d'utilisation du processeur d'un noeud, le redimensionnement de la VM en fonction de leurs besoins en ressources et la prise de décision (quand et où les machines virtuelles doivent être migrées à partir d'un noeud physique).

V) **Global manager :** Il réside sur un nœud maître et recueille des informations

auprès des gestionnaires locaux pour maintenir la vue globale de l'utilisation des ressources.

Les gestionnaires locaux se trouvent sur chaque nœud en tant que module du moniteur de machine virtuelle (VMM). Leur objectif est le suivi continu d'utilisation du processeur d'un nœud, le redimensionnement de la VM en fonction de leurs besoins en ressources et la prise de décision (quand et où les machines virtuelles doivent être migrées à partir du nœud d'accueil) (4). Ils permettent aussi de recevoir des requêtes à partir des clients (1). Le gestionnaire global réside sur un nœud maître et recueille des informations provenant des responsables locaux pour maintenir la vue d'ensemble de l'utilisation des ressources (2). Le gestionnaire global émet des commandes pour l'optimisation du placement VM (3). VMM effectue des redimensionnements réels et la migration des machines virtuelles ainsi que des changements dans les états d'alimentation des nœuds (5).

3.3.1. Calcul de la consommation d'énergie

Ils existent plusieurs méthodes pour calculer la consommation d'énergie, parmi lesquels nous avons utilisé la "méthode puissance par rapport à l'utilisation" : De nombreuses études, [FW07], [KKJ08] ont montré que la consommation d'énergie par les serveurs peut être décrite par une relation linéaire entre la consommation d'énergie et l'utilisation du processeur. Ces études confirment qu'une puissance moyenne consommée par un serveur inactif est de 70% de l'énergie consommée par rapport à un serveur pleinement utilisé et d'après [SP11] , si l'utilisation du processeur est supérieure à 30%, la valeur inférieure est toujours 0,3. Donc, ils ont défini la consommation d'énergie $P(u)$ par la Formule 3.1 :

$$P(u) = Pmax * (0.7 + 0.3u) \tag{3.1}$$

Tel que : $Pmax = 250$w pour des serveurs modernes. La constante 0.7 est la puissance moyenne consommée par un serveur inactif quand le serveur est

pleinement utilisé. u : est l'utilisation du processeur. Comme l'utilisation de CPU peut changer avec le temps en raison de la variabilité de la charge de travail, il s'agit d'une fonction du temps : $u(t)$. Par conséquent, pour définir la consommation d'énergie totale par un serveur, nous utilisons l'équation 3.2 .

$$E_S = \int_t P(u(t))dt \tag{3.2}$$

Donc la consommation d'énergie par rapport à un data center est calculée par la Formule 3.3 [LH12] :

$$E_D = \frac{\sum_{j=1}^{m} P(u)_j}{m} \tag{3.3}$$

m : nombre de machines physiques dans un data center.

3.3.2. Coût de la migration à Chaude des VMs

La migration à chaude a un impact négatif sur les performances des applications en cours d'exécution dans une machine virtuelle pendant une migration. Voorsluys et al [Voo09], ont effectué une étude expérimentale pour étudier la valeur de cet impact et trouver un moyen de modélisation. Ils ont découvert que la dégradation des performances et des temps d'arrêt dépendent du comportement de l'application, c'est à dire, combien de pages mémoires sont mise à jour par l'application pendant son exécution. Pour les applications de web, la dégradation de performance moyenne et le temps d'arrêt peut être estimé à 10% de l'utilisation du processeur. Cela signifie que chaque migration peut entraîner une certaine violation SLA. Il est donc essentiel de réduire au minimum le nombre de migrations de machines virtuelles. La durée d'une migration à chaude dépend de la quantité totale de mémoire utilisée par la machine virtuelle et la bande

passante disponible. Pour nos expériences, nous définissons la dégradation des performances expérimentées par les VM_j par la formule 3.5 [BB10] :

$$T_{m_j} = \frac{M_j}{B_j} \tag{3.4}$$

$$U_{d_j} = 0.1 \cdot \int_{t_0}^{t_0+T_{m_j}} u_j(t)dt \tag{3.5}$$

Où U_{d_j} est la dégradation totale de performance par la VM_j, t_0 le temps où la migration commence, T_{m_j} est le temps nécessaire pour terminer la migration, $u_j(t)$ est l'utilisation de la CPU par la VM_j, M_j est la quantité de mémoire utilisée par la VM_j et B_j est la bande passante réseau disponible.

3.3.3. Violation de SLA

Les exigences de la qualité de service sont extrêmement importantes pour le Cloud computing. Elles sont généralement formalisées sous forme de SLA[1]. Le contrat SLA peut être déterminé en termes de plusieurs contraintes telles que le débit minimum ou le temps de réponse maximum fourni par le système déployé. Etant donné que ces caractéristiques peuvent varier selon les applications, il est nécessaire de définir une métrique générique qui peut être utilisée dans notre simulation pour estimer le niveau du SLA qui est délivré par l'infrastructure. Pour ce qui nos concerne, nous définissons le niveau global de violation SLA causé par le système comme une fraction de la différence entre les MIPS[2] demandées par toutes les machines virtuelles $U_{r_j}(t)$ et les MIPS réellement alloués $U_{a_j}(t)$. Cela est relatif au total des MIPS demandés au cours de la durée de vie des machines virtuelles (voir équation 3.6) [BB10] . Où M est le nombre de machines virtuelles.

1. Service Level Agreement (SLA) est un document qui définit la qualité de service requise entre un prestataire et un client.
2. MIPS : *Million d'instructions par seconde*, unité de mesure des processeurs.

$$SLA = \frac{\sum_{j=1}^{M} \int_t U_{r_j}(t) - U_{a_j}(t) dt}{\sum_{j=1}^{M} \int_t U_{r_j}(t)} \tag{3.6}$$

Cette métrique représente le pourcentage de la performance du CPU qui n'est pas affféctée lorsqu'elle est demandée par les applications relatives à la demande totale.

3.4. La stratégie de la minimisation des migrations des machines virtuelles

Notre approche est divisée en deux phases : la sélection des VM à faire migrer et l'allocation des VMs (voir Figure 3.2).

1. **Phase 1 :** Nous sélectionnons dans cette phase les machines virtuelles qui doivent être migrées (quelle VM doit-on choisir pour faire une migration ?) ;
2. **Phase 2 :** Les machines virtuelles choisies sont placées dans les machines physiques à l'aide d'un algorithme de placement (quel est le nouvel emplacement de la VM ?).

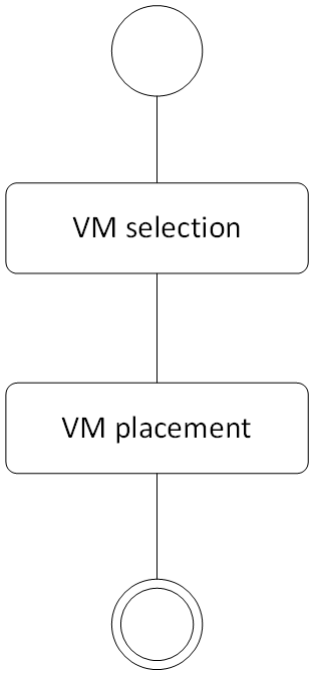

FIGURE 3.2 – *Diagramme d'activité de la migration des machines virtuelles*

3.4.1. Sélection des machines virtuelles

Pour déterminer quand les machines virtuelles doivent être migrées, nous utilisons deux seuils d'utilisation du CPU (voir Figure 3.3).

3.4.1.1. Seuils d'utilisation fixe

L'idée est de fixer deux seuils d'utilisation de CPU (seuil supérieur et seuil inférieur) et de garder l'utilisation totale du CPU de toutes les machines physique entre ces seuils. Si l'utilisation du CPU d'une machine physique est en dessous du seuil inférieur, toutes les VMs doivent être migrées de cette PM. Cette dernière est mise après en état hors tension (éteinte).

Si l'utilisation du CPU de la machine physique dépasse le seuil supérieur, on cherche la meilleure machine virtuelle pour faire une migration, afin de réduire l'utilisation de CPU. La meilleure VM est celle qui répond à deux conditions !

I) la machine virtuelle doit avoir une utilisation plus grande que la différence entre l'utilisation globale de CPU et le seuil supérieur.

II) la différence entre le seuil supérieur et la nouvelle utilisation de CPU est la plus petite. Si une telle VM n'existe pas, l'algorithme sélectionne la VM avec la plus grande utilisation CPU.

3.4.1.2. Seuils d'utilisation dynamique

Comme mentionné précédemment, les seuils fixes ne sont pas adaptés pour un environnement avec des charges de travail dynamiques et imprévisibles. Cet environnement possède différents types d'applications qui peuvent partager la même ressource physique. Le système doit être capable d'ajuster automatiquement son comportement en fonction des modèles de charge de travail exposées par les applications. C'est pourquoi, nous utilisons une nouvelle technique pour calculer automatiquement les deux seuils d'utilisation CPU. Cette technique est basée sur sur une analyse statistique des données historiques recueillies au cours de la durée de vie des machines virtuelles [SPD11]. Nous allons présenter maintenant les formules permettant de calculer les seuils fixes.

1. **Seuil supérieur** T_{U_i} **:** Nous avons calculé seuil supérieur T_{U_i} comme montré dans 3.9.

$$U_i = \sum_{j=1}^{m} u_j \tag{3.7}$$

$$S_{U_i} = \sqrt{\sum_{j=0}^{m} u_j^2} \tag{3.8}$$

$$T_{U_i} = 1 - (((P_{uu} * S_{U_i}) + U_i) - ((P_{ul} * S_{U_i}) + U_i) \tag{3.9}$$

u_j : L'utilisation du CPU de la machine virtuelle *j*.
m : Le nombre de machines virtuelles.
U_i : L'utilisation du CPU de la machine physique *i*.
P_{uu} : égale à 95% [SPD11].
P_{ul} : égale à 90% [SPD11].

2. **Seuil inférieur** T_l **:** Nous avons calculé seuil inférieur T_l comme montré dans 3.12.

$$U_i = \frac{1}{m} \sum_{j=1}^{m} u_j \tag{3.10}$$

$$S_{U_i} = \sqrt{(\sum_{j=0}^{m} u_j - U_i)^2} \tag{3.11}$$

$$T_l = \begin{cases} U_i - (P_l * S_{U_i}) & \text{, Si } L'utilisation \ du \ CPU < 0.3 \\ 0.3 & \text{, Si } L'utilisation \ du \ CPU > 0.3 \end{cases} \tag{3.12}$$

u_j : L'utilisation du CPU de la machine virtuelle *j*.
m : Le nombre de machines virtuelles.
U_i : L'utilisation du CPU de la machine physique *i*.
P_l : égale à 90% [SPD11].

3.4.1.3. Phase de sélection des VMs

La Figure 3.3 montre un diagramme d'activité qui résume la phase de sélection des machines virtuelles :

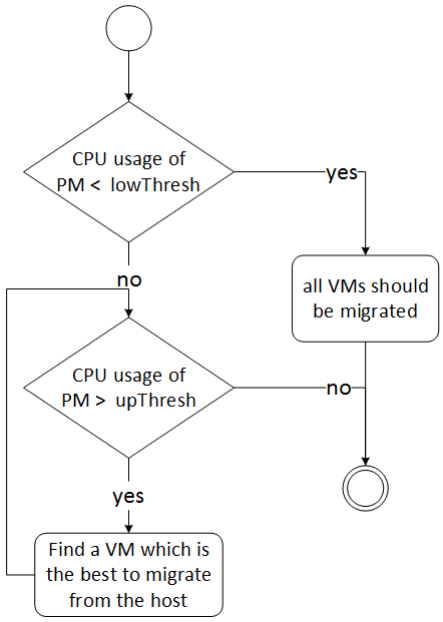

Figure 3.3 – *Diagramme d'activité de sélection des machines virtuelles*

L'algorithme 1 illustre le pseudo-code de la procédure de sélection des machines virtuelles.

Algorithme 1 : Algorithme de la sélection des machines virtuelles

Entrées : hostList, vmList
Sorties : migrationList

```
1  vmList.sortDecreasingUtilization();
2  pour chaque h dans hostList faire
3      hUtil ← h.util();
4      bestFitUtil ← MAX;
5      tant que hUtil > h.upThresh() faire
6          pour chaque vm dans vmList faire
7              si vm.util() > hUtil - h.upThresh() alors
8                  t ← vm.util() - hUtil + h.upThresh();
9                  si t < bestFitUtil alors
10                     bestFitUtil ← t;
11                     bestFitVm ← vm;
12             sinon
13                 si bestFitUtil = MAX alors
14                     bestFitVm ← vm;
15                 break;
16         hUtil ← hUtil - bestFitVm.util();
17         migrationList.add(bestFitVm);
18         vmList.remove(vm);
19     si hUtil < h.lowThresh() alors
20         migrationList.add(h.getVmList());
21         vmList.remove(h.getVmList());
22 retourner migrationList;
```

L'algorithme 1 est composé de deux parties : La première partie de la ligne 5 jusqu'à la ligne 18 s'agit d'une boucle qui permet de réduire l'utilisation totale du CPU en dessous de seuil supérieur par sélection de quelques VMs à faire migrer. La seconde partie de la ligne 19 jusqu'à la ligne 21, il permet de vérifier si le seuil inférieur est violé, alors toutes les machines virtuelles seront migrées.

3.4.2. Placement des machines virtuelles MBFD

Pour déterminer l'emplacement des VMs migrées, nous utilisons l'algorithme MBFD [3]. Cet algorithme tri les machines virtuelles dans l'ordre décroissant de leur utilisation de CPU. Il alloue par la suite chaque VM dans une PM qui offre une consommation d'énergie minimale grâce à cette allocation (voir Figure 3.4).

La Figure 3.4 montre un diagramme d'activité qui résume la phase de placement des machines virtuelles :

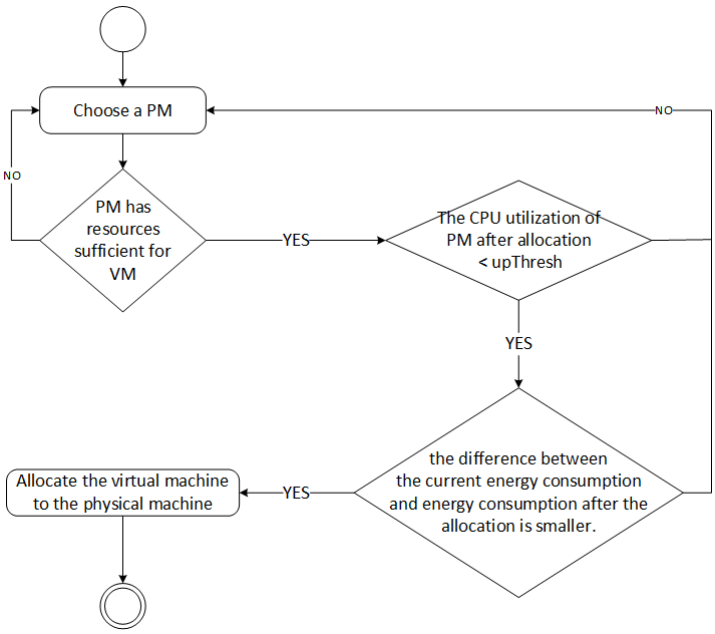

Figure 3.4 – *Diagramme d'activité de placement des machines virtuelles (MBFD)*

L'algorithme 2 illustre le pseudo-code de la procédure de placement des machines virtuelles.

3. Modified Best Fit Decreasing (MBFD)

Algorithme 2 : Algorithme de placement des machines virtuelles (MBFD)

Entrées : hostList, vmList
Sorties : allocation of VMs

```
1  vmList.sortDecreasingUtilization();
2  pour chaque vm dans vmList faire
3      minPower ← MAX;
4      allocatedHost ← NULL;
5      pour chaque host dans hostList faire
6          si host a suffisamment de ressources pour vm alors
7              power ← estimatePower(host, vm);
8              si power < minPower alors
9                  allocatedHost ← host;
10                 minPower ← power;
11     si allocatedHost ≠ NULL alors
12         allouer vm à allocatedHost;
13 retourner allocation;
```

L'algorithme 2 est composé de deux boucles imbriquées, la première boucle (ligne 2) permet de parcourir la liste des VMs à faire migrer. La seconde boucle (ligne 5) permet de parcourir la liste de toutes les PMs pour placer les VMs.

La complexité de l'algorithme 2 (MBFD) est $n * m$, où n est le nombre de noeuds et m est le nombre des machines virtuelles qui doivent être migrées.

3.5. Conclusion

Dans ce chapitre, nous avons présenté et décri les différentes étapes de notre stratégie proposée afin de minimiser le nombre de migration des machines virtuelles et par conséquence nous réduisant la consommation d'énergie .

Nous avons décrit le fonctionnement de notre proposition à l'aide d'un ensemble d'algorithmes, de formules et de diagrammes pour faciliter l'étude et la compréhension de notre proposition, d'une part, et pour donner un schéma général du travail demandé d'autre part.

En vue de concrétiser cette stratégie, nous allons décrire son implémentation dans le prochain chapitre sous le simulateur CloudSim, cela en utilisant le langage de programmation Java et l'environnement Netbeans.

Chapitre 4

Implémentation et Résultats

Sommaire

4.1. Introduction

Ce chapitre est consacré à la réalisation et la concrétisation de nos approches proposées, qui consistent à minimiser l'énergie consommée dans les Cloud Computing. Il aborde l'implémentation de notre stratégie. Nous allons commencer tout d'abord par fixer l'environnement dans lequel nous avons réalisé notre simulateur. Nous allons définir par la suite les métriques que nous avons utilisées. Enfin nous allons discuter et analyser les résultats de la minimisation des migrations des machines virtuelles dans le Cloud Computing que nous avons obtenus.

4.2. Langage et environnement de développement

L'environnement de développement est un facteur important qui doit être détaillé pour connaître dans quelles situations, le même travail peut être reproduit. La stratégie proposée dans le cadre de ce travail a été implémentée et testée dans l'environnement suivant :

- **Caractéristiques matérielles et logicielles du PC utilisé :** Nous avons développé notre application sur une machine avec un processeur Intel(R) Core(TM)i3-3110M CPU, une vitesse de 2.40Ghz et une capacité mémoire de 4GB. Le simulateur est sous Windows 8.1 de 64bits.
- **Simulateur utilisé :** CloudSim.
- **Langage utilisé :** Java.
- **IDE utilisé :** NetBeans.

4.2.1. Langage de programmation Java

Java est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution portable. Il est céé par James Gosling et Patrick

Naughton employés de Sun Microsystems avec le soutien de Bill Joy (co-fondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld [Wik14].

Le langage Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que : Unix, Microsoft Windows, Mac OS ou Linux avec ou sans modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java [Wik14].

Les applications Java peuvent être exécutées sur tous les systèmes d'exploitation pour lesquels a été développée une plate-forme Java dont le nom technique est JRE (Java Runtime Environment - Environnement d'exécution Java). Cette dernière est constituée d'une JVM (Java Virtual Machine - Machine Virtuelle Java), le programme qui interprète le code Java et le convertit en code natif. Mais le JRE est surtout constitué d'une bibliothèque standard à partir de laquelle doivent être développés tous les programmes en Java. C'est la garantie de portabilité qui a fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, ce qui est très difficile pour les gros systèmes [IPE10].

Java est devenu aujourd'hui une direction incontournable dans le monde de la programmation parmi les différentes caractéristiques qui sont attribuées à son succès , nous avons [Wik14] :

- L'indépendance de toute plate-forme : le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine.
- Java est également portable, permettant à la simulation d'être distribuée facilement sans avoir à recompiler le code pour les différents systèmes.
- Le code est structuré dans plusieurs classes dont chacune traite une partie différente de la simulation.
- Il assure la gestion dynamique de la mémoire.

- Java est multitâches : il permet l'utilisation de Threads qui sont des unités d'exécution isolées.

Aussi, une des principales raisons de ce choix est que le simulateur CloudSim est développé avec ce langage.

4.2.2. Environnements de développement

NetBeans est un environnement de développement intégré (EDI), placé en Open Source par Sun en Juin 2000. En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C + +, JavaScript, XML, Ruby, PHP et HTML téléchargeable du site : https://netbeans.org/downloads. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris, MacOSX ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

De plus, NetBeans est écrit en Open Source, téléchargeable directement du site http://java.sun.com. Il est puissant et compatible avec toutes les nouvelles technologies Java (les technologies Java EE, les bases de données, UML, XML, ...).

4.2.3. Simulateur CloudSim

CloudSim est une nouvelle structure de simulation généralisée et extensible qui permet la modélisation des environnements hétérogènes, la simulation et l'expérimentation de Cloud émergent des infrastructures de calcul et des services d'application. Il offre les fonctionnalités suivantes :

- Une plate-forme indépendante pour la modélisation des Data Centers, des Brokers, de l'ordonnancement et des politiques d'allocation des ressources ;

- Support pour la modélisation et la simulation à grande échelle d'infrastructure de Cloud Computing, y compris des centres de données sur un seul noeud physiques.

Parmi les principales caractéristiques de CloudSim, nous pouvons citer :

- La flexibilité pour commuter entre l'allocation en espace et en temps partagé et la prise en charge de la répartition des coeurs de traitement aux services virtualisés ;
- La disponibilité d'un moteur de virtualisation qui facilite la création et la gestion indépendante des services ainsi que l'hébergement des services virtualisés sur un noeud d'un Datacenter ;
- Un support pour la simulation des connexions réseau entre les éléments du système de simulation ;
- Simulation de la définition de matériel de centre de traitement des données (Datacenter) en termes de machines physiques composées de processeurs, de dispositifs de stockage, de mémoire et de largeur de bande interne ;
- Simulation des spécifications, de la création et de la destruction de machines virtuelles ;
- Simulation de l'exécution des programmes utilisateurs ou des demandes (Cloudlet) sur les machines virtuelles.

Nous avons utilisé pour la réalisation de notre travail la version du simulateur CloudSim 2.1.1 téléchargeable du site : https://code.google.com/p/cloudsim/downloads/list.

4.2.3.1. Architecture de CloudSim

La structure logicielle de CloudSim et ses composants sont représentés par une architecture en couches comme c'est montré dans la Figure 3.1. Les premières versions de CloudSim utilise SimJava, un moteur de simulation d'événement

discret qui met en oeuvre les principales fonctionnalités requises pour des structures de simulation de haut niveau. Parmi les fonctionnalités, nous avons la formation d'une file d'attente et le traitement d'événements, la création de composants système (les services, les machines (Host), le centre de données (Datacenter), le courtier (Broker), les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation :

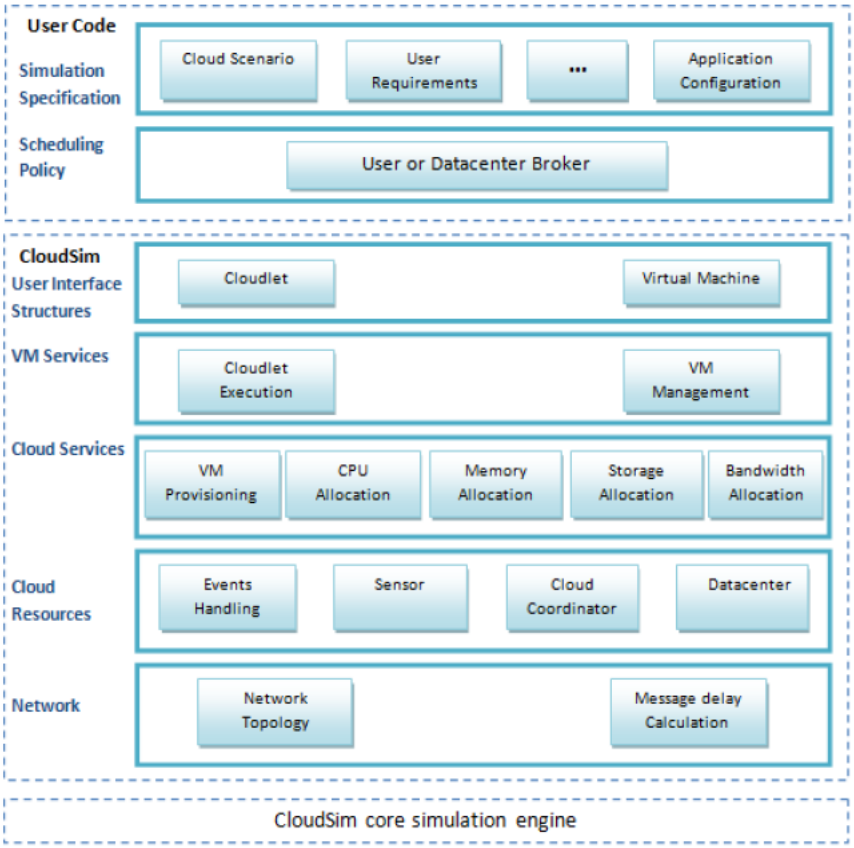

Figure 4.1 – *Architecture de CloudSim [CR10]*

4.2.3.2. Classes de CloudSim

Le simulateur CloudSim est composé de plusieurs classes que nous pouvons classer en deux catégories : des classes qui modélisent les entités comme le Data Center, le Broker, etc. Et des classes modélisant les politiques d'allocation.

Parmi les classes fondamentales qui forment les blocs constitutifs du simulateur CloudSim comme est présenté dans la Figure 4.2 , nous pouvons citer seulement celles utilisées :

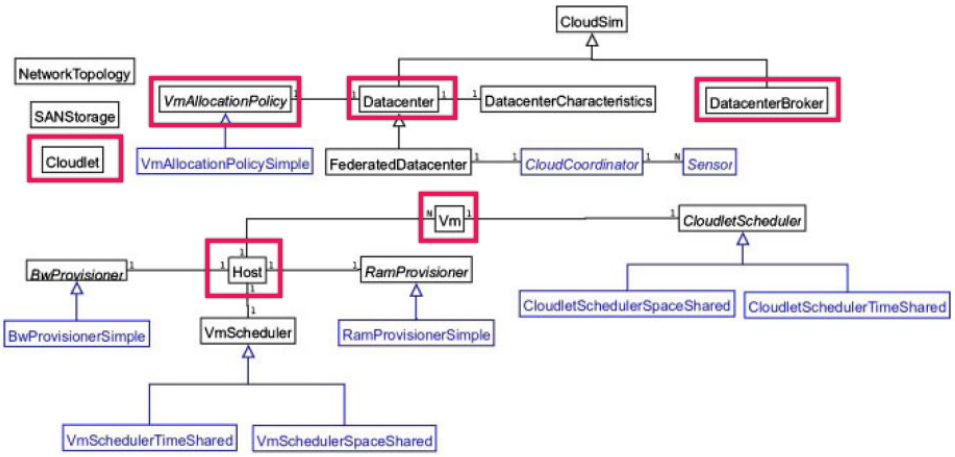

Figure 4.2 – *Diagramme de classe de la conception de simulateur CloudSim*

A. **Cloudlet :** Cette classe modélise les services d'application du Cloud (comme la livraison, réseaux sociaux, et le workflow d'affaires). CloudSim représente la complexité d'une application en fonction de ses besoins informatiques. Chaque service d'application a une taille d'instruction pré-assigne et la quantité de flux de transfert de données qu'il doit entreprendre au cours de son cycle de vie. Cette classe peut également être étendu pour supporter la modélisation de la performance et d'autres paramètres de composition pour les applications telles que les transactions dans les applications orientées base de données [RC10].

B. **Datacenter :** Cette classe modélise les services au niveau des infrastructures de base (matériel) qui sont offerts par les fournisseurs de Cloud (Amazon, Azure, App Engine). Elle encapsule un ensemble de hôtes qui peuvent être soit homogènes ou hétérogènes par rapport à leurs configurations matérielles (mémoire, noyaux, capacité et stockage) [RC10].

C. **DatacenterBroker :** Cette classe modélise le Broker, qui est responsable de la médiation entre les utilisateurs et les prestataires de service selon les conditions de QoS des utilisateurs et il déploie les tâches de service à travers les Clouds. Le Boker au nom des utilisateurs, agit sur les prestataires du service approprié du cloud par le service d'information du Cloud CIS (Cloud Information Services) et négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs. Les chercheurs et développeurs des systèmes doivent étendre cette classe pour évaluer et tester les politiques de courtage personnalisées. [RC10].

D. **DatacenterCharacteristics :** Cette classe contient les informations sur la configuration des ressources des centres de données [RC10].

E. **Host :** Cette classe modélise une ressource physique comme le serveur de stockage ou de calcul. Elle encapsule des informations importantes telles que la quantité de mémoire et de stockage, le type de coeurs de traitement (pour représenter une machine multi-core), une politique d'allocation pour le partage de la puissance du traitement entre les machines virtuelles et les politiques d'approvisionnement de mémoire et de bande passante pour les machines virtuelles [RC10].

F. **SimEntity :** Il s'agit d'une classe abstraite, elle représente l'entité de simulation qui est capable d'envoyer des messages à d'autres entités et de gérer les messages reçues ainsi que les événements. Toutes les entités doivent étendre cette classe et redéfinir ses trois principales méthodes : *startEntity()*, *processEvent()* et *shutdownEntity()*. Ces méthodes définissent les actions pour l'initialisation de l'entité, le traitement des événements et la destruction de l'entité [RC10].

G. **VM :** Cette classe représente une instance de machine virtuelle (VM) qui est gérée et hébergée par une machine physique (hôte). Chaque composant VM a accès à un composant qui stocke les caractéristiques suivantes liées à une VM telles que : mémoire accessible, le processeur, capacité de stockage, et les politiques de provisionnement interne de la machine virtuelle qui

est étendu à partir d'un composant abstrait appelé le CloudletScheduler [RC10].

H. **VMAllocationPolicy :** C'est une classe abstraite implémentée par un composant hôte qui modélise les politiques (d'espace partagé, de temps partagé) exigées pour allouer la puissance de traitement aux VMs. Les fonctionnalités de cette classe peuvent facilement être ignorées pour accommoder des politiques spécifiques à l'application de partage de processeur [RC10].

I. **VMProvisioner :** Cette classe abstraite représente la politique d'approvisionnement qu'un moniteur de VM utilisé pour allouer les VMs aux hôtes. La VMProvisioner sélectionne l'hôte qui répond à la quantité de mémoire demandée, le stockage pour le déploiement de la VM [RC10].

4.3. Description du fonctionnement de notre logiciel

Nous présentons dans cette partie une vue globale de notre simulateur en détaillant les différentes étapes à effectuer pour réaliser une simulation.

Selon le diagramme de la Figure 4.3, la simulation commence après :

1. **Création du Cloud :** étape permettant de créer l'architecture Cloud en spécifiant le nombre de DataCenters.
2. **Configuration de Data Center :** Cette étape permet de spécifier les caractéristiques du Data Center telle que le nombre de nœuds physiques, la bande passante, etc.
3. **Configuration des PMs :** Cette étape permet de créer une machine physique et spécifier ses caractéristique telle que nombre de processeurs, la capacité de stockage, etc.
4. **Configuration des VMs :** Cette étape permet de créer une machine virtuelle.

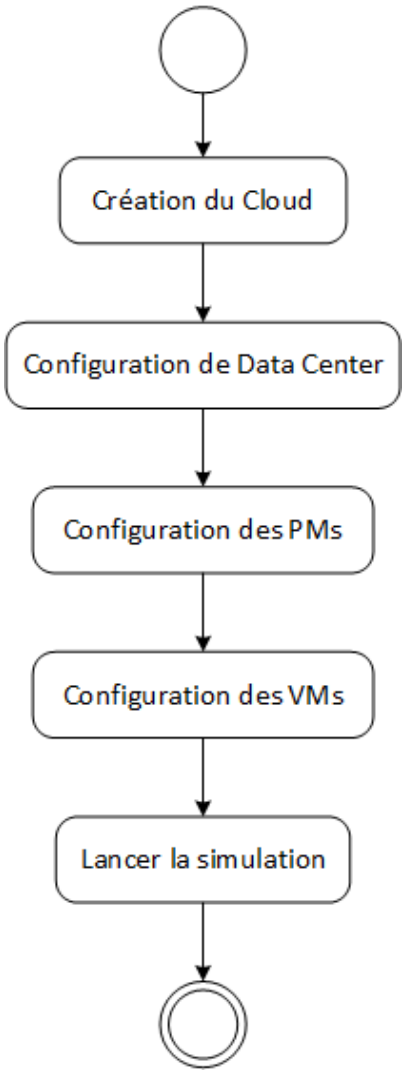

FIGURE 4.3 – *Diagramme d'activité du fonctionnement de notre logiciel*

4.4. IMPLÉMENTATION

Dans cette partie, nous allons nous intéresser à la démonstration de notre application à travers un exemple en faisant référence à quelques interfaces graphiques.

4.4.1. Accès à l'interface

La version de CloudSim n'a pas d'interface graphique, son exécution se fait sur la console donc nous avons créé une interface qui facilite l'accès au simulateur. L'interface doit faire appel à CloudSim ainsi qu'aux différentes approches qui se trouvent dans différents packages.

La Figure 4.4 représente la première interface de notre simulateur qui apparaît à l'utilisateur.

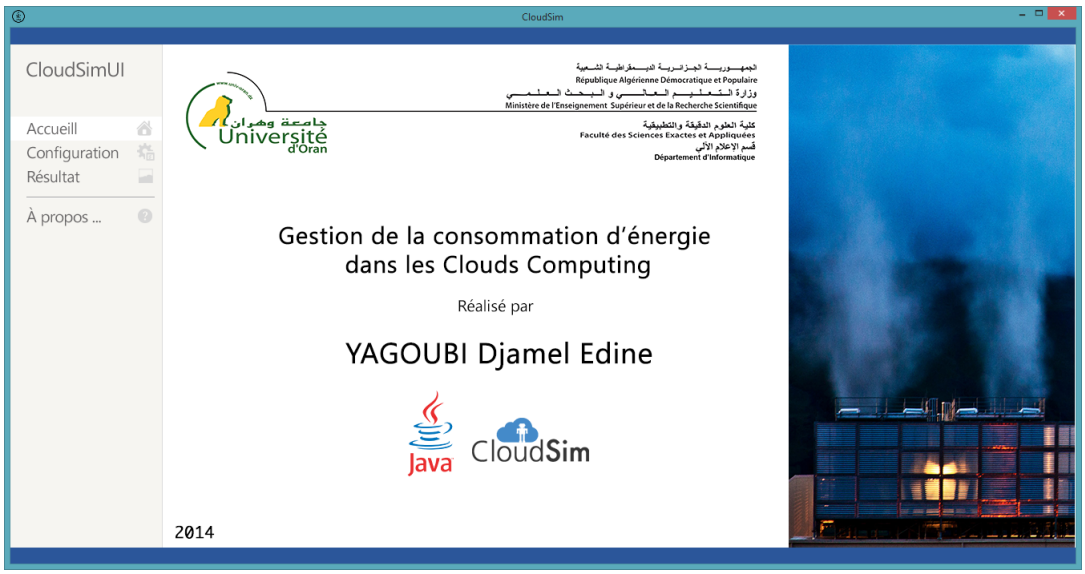

Figure 4.4 – *Interface principale*

4.4.2. Configuration de Simulation

La première action à effectuer est la configuration des composants du Cloud. Cette configuration est réalisée en quatre étapes :

4.4.2.1. Configuration du Data Center

Cette étape consiste à faire entrer les caractéristiques du Data Center (voir Figure 4.5) comme : le nom du Data Center, l'architecture de système d'exploitation, le coût de traitement, le coût de la mémoire, le coût de stockage,

le coût de la bande passante , l'intervalle de l'ordonnancement et le système d'exploitation.

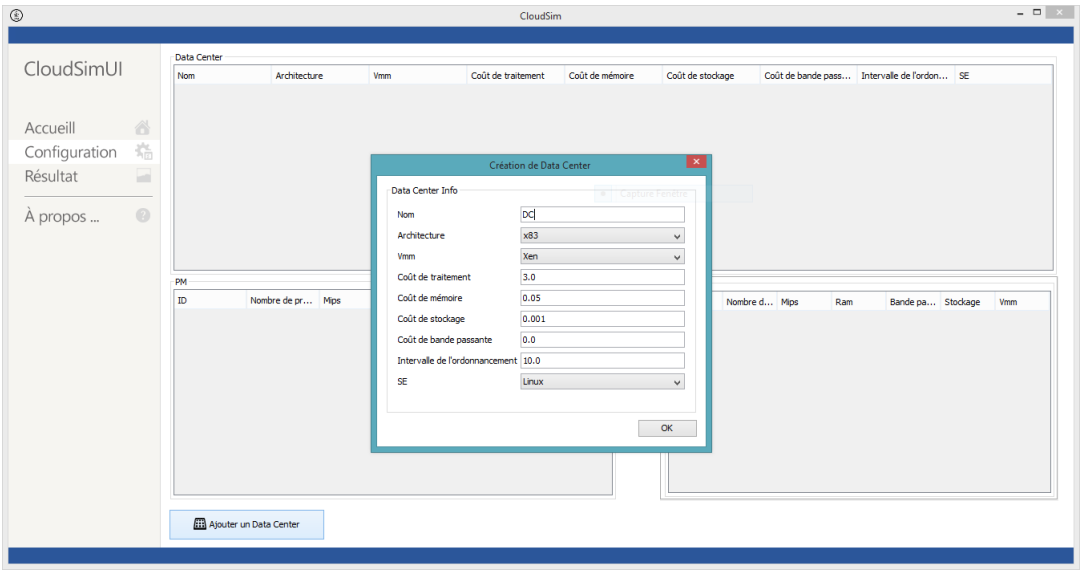

FIGURE 4.5 – *Configuration du Data Center*

4.4.2.2. CONFIGURATION DES MACHINES PHYSIQUES

La Figure 4.6 représente l'étape de création des machines physiques. Le bouton ***Ajouter des machines physiques*** permet de créer des machines physiques hétérogènes en précisant le nombre des hôtes, nombre de processeur, MIPS, RAM, capacité de stockage et la bande passante.

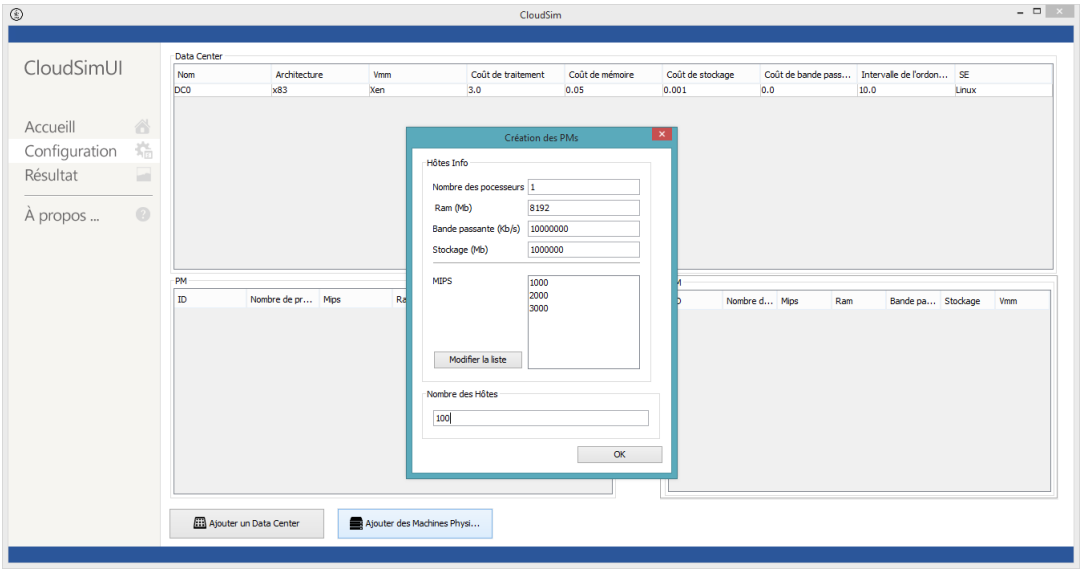

Figure 4.6 – *Configuration des machines physiques*

4.4.2.3. Configuration des machines virtuelles et Cloudlets

La création des Cloudlets et des machines virtuelles hétérogènes se fait en cliquant sur le bouton **Ajouter des machines virtuelles** (voir Figure 4.7). Pour cela, il faut préciser le nombre des Cloudlets, le nombre de processeur, MIPS, RAM, capacité de stockage, la bande passante, taille de cloudlet, taille de fiché de cloudlet et la taille de sortie de cloudlet.

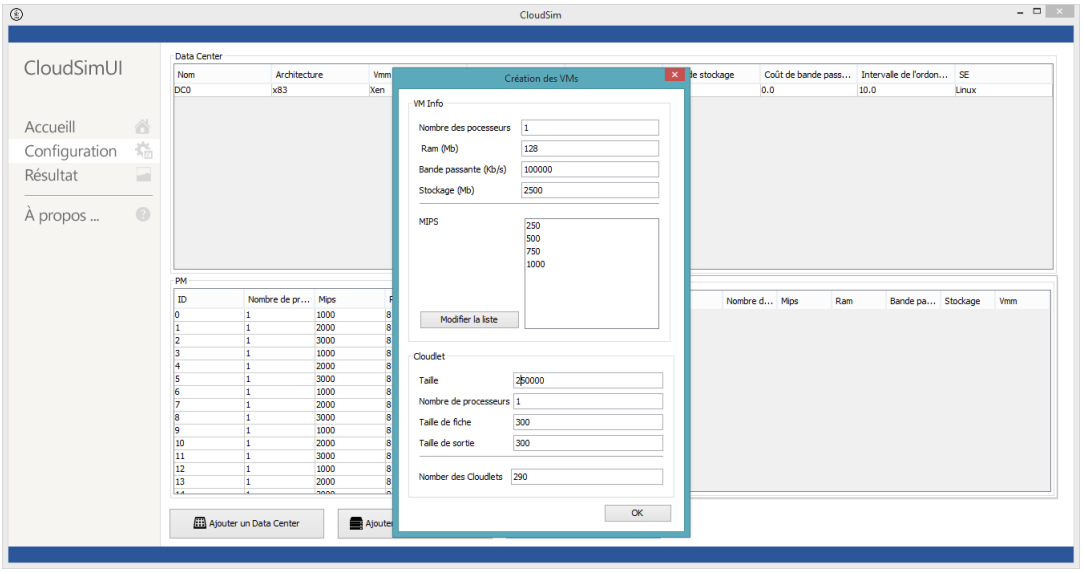

FIGURE 4.7 – *Configuration des machines virtuelles et Cloudlets*

4.4.2.4. LANCEMENT DE LA SIMULATION

Le bouton ***Lancer la simulation*** illustré dans la Figure 4.8 permet de choisir les paramètres de simulation en spécifiant le nombre de simulations, le seuil de l'approche ***Single Threshold*** et les seuils supérieurs et inférieurs de l'approche ***Fixed Double Threshold***. Nous pouvons lancer les simulations en cliquant sur le bouton ***OK***.

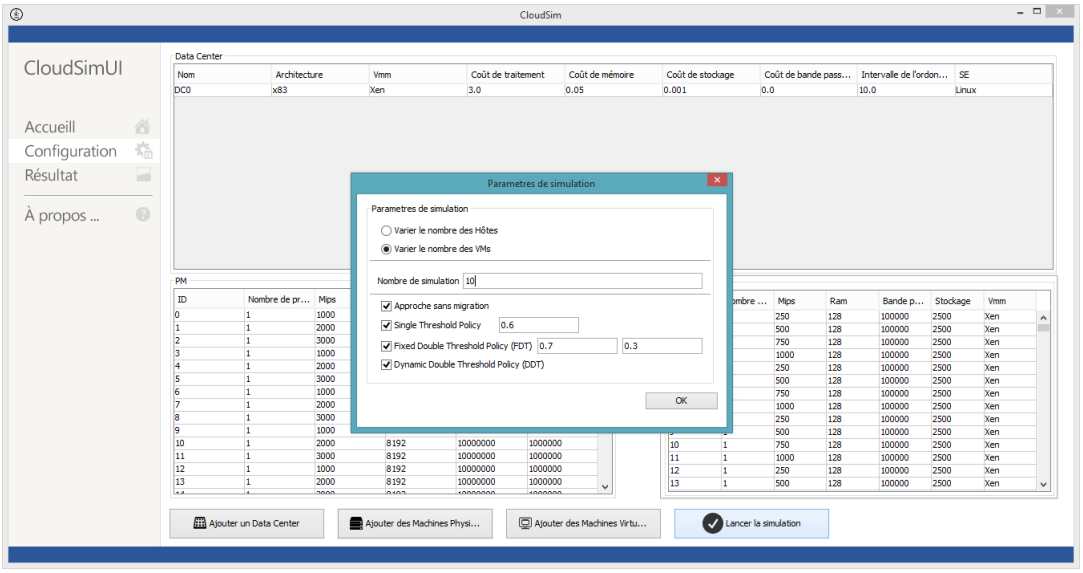

FIGURE 4.8 – *Lancement de la simulation*

4.4.2.5. RÉSULTAT

Le bouton « Résultat » permet à son tour d'afficher les résultats des simulations représentées dans le graphe (voir Figure 4.9) à l'aide d'un API java JFreeChart [1].

1. **JFreeChart** est une API Java permettant de créer des graphiques et des diagrammes de très bonne qualité.

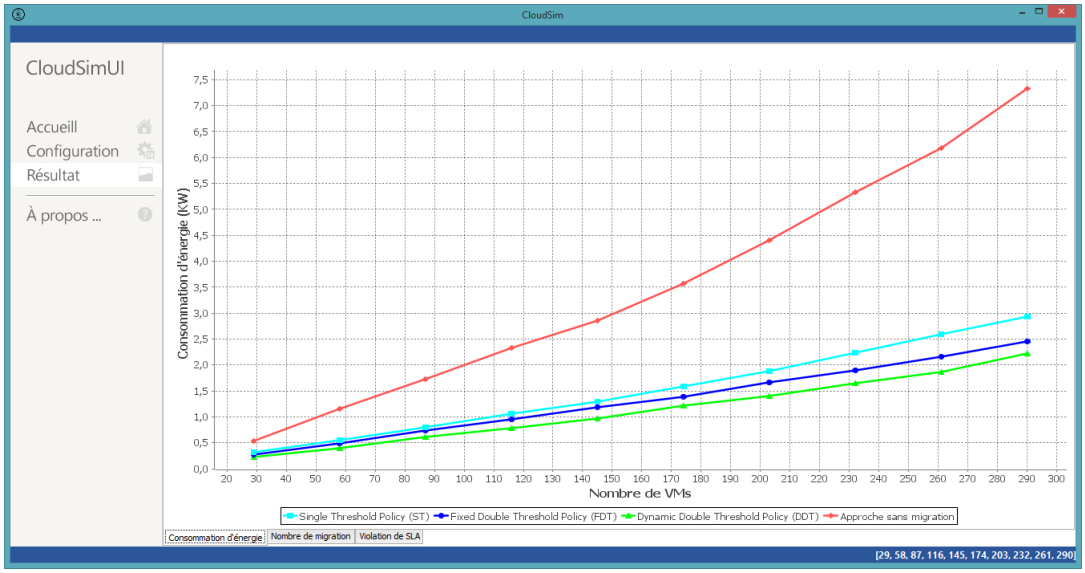

FIGURE 4.9 – *Résultat*

4.5. EXPÉRIMENTATIONS

Pour mettre en valeur les apports de nos approches, nous allons se focaliser sur les métriques suivantes : l'énergie consommée, nombre de migrations et violations de SLA. Dans le but d'étudier le comportement de nos propositions et d'analyser ses résultats obtenus par la simulation, nous allons les comparer à deux approches : l'approche sans migration et l'approche ***Single Threshold***. Plusieurs séries de simulation ont été lancées selon plusieurs paramètres.

4.5.1. EXPÉRIENCE 1 : VARIATION DU NOMBRE DE VMS

Dans cette simulation, nous avons créé un Data Center contenant 100 hôtes hétérogène, Chaque hôte posséde 1 processeur avec une vitesse variante en MIPS (1000, 2000 ou 3000), 8GB de mémoire, 1TB de stockage et consomme de 175 W avec 0% d'utilisation du processeur jusqu'à 250 W avec 100% d'utilisation du processeur. Le Broker fait varier un nombre de VMs hétérogène de 29 à 290 pour chaque VM a 1 processeur avec une vitesse variante en MIPS (250, 500,

750 ou 1000), 128MB de mémoire, 1GB de stockage. Nous avons considéré le seuil de l'approche **Single Threshold** comme 0.6 d'après [BB10]. Comme seuil supérieur et inférieur de l'approche **Fixed Double Threshold** , nous avons pris les valeurs 0,6 et 0,3 respectivement d'après [BB10].

4.5.1.1. Consommation d'énergie

Pour analyser la consommation d'énergie, les résultats obtenus par les simulations sont résumés numériquement dans les tableaux 4.1 et 4.2 et schématisés par la Figure 4.10.

Nombre de VMs	29	58	87	116	145	174	203	232	261	290
Approche sans migration	5.21	10.83	16.61	22.45	27.57	33.69	38.65	44.15	50.22	60.55
Single Threshold	4.69	9.30	13.78	18.39	22.80	27.78	33.21	38.14	43.97	49.98
Fixed Double Threshold	3.90	7.86	11.73	15.76	19.53	23.66	27.62	32.06	36.38	40.93
Dynamic Double Threshold	3.67	7.19	10.56	13.89	17.62	20.63	24.61	27.97	31.91	36.26

Table 4.1 – *Comparaison entre la consommation d'énergie des quatre approches*

Le tableau 4.2 montre la différence entre les quatre approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit l'énergie avec un gain moyen de 36,21% par rapport à l'approche sans migration, un gain moyen de 24,81% par rapport à l'approche **Single Threshold** et un gain moyen de 10,61% par rapport à l'approche **Fixed Double Threshold**.

Nom de l'approche	Gain
Approche sans migration et Dynamic Double Threshold	36,21%
Single Threshold et Dynamic Double Threshold	24,81%
Fixed Double Threshold et Dynamic Double Threshold	10,61%

Table 4.2 – *Comparaison entre la consommation d'énergie des quatre approches*

Nous remarquons dans cette simulation représentée par la Figure 4.10, une augmentation d'énergie consommée par rapport aux quatre approches. Nous pouvons dire que cela est dû à l'augmentation du nombre de VMs. Nous remarquons aussi que les courbes des approches proposées (la courbe bleue et la courbe

verte) se trouvent au-dessous des courbes des autres approches (la courbe rouge et la courbe cyan). Donc nos propositions sont bien meilleure par rapport aux autres approches et elle permettent d'économiser de l'énergie. Nous déduisons que notre approche dégage moins de chaleur, cela permet de la classer parmi les approches de l'informatique verte.

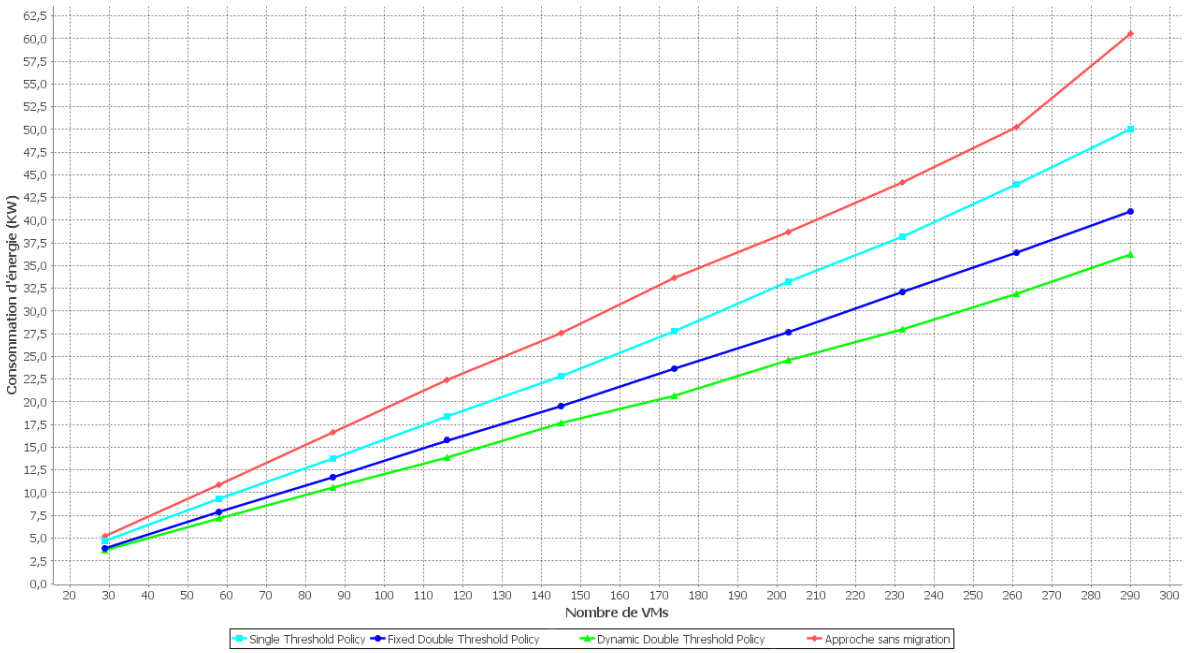

Figure 4.10 – *Influence du nombre des VMs sur l'énergie*

4.5.1.2. Nombre de migrations

Dans cette série de simulations représentée par la Figure 4.11, nous remarquons une augmentation du nombre de migrations dans les trois approches. Nous pouvons justifier cela par l'augmentation du nombre des VMs. Nous remarquons aussi qu'avec l'augmentation du nombre de VMs, la différence entre les courbes augmente et les courbes des approches proposées se trouvent au-dessous de la courbe de l'approche **Single Threshold**. Donc les approches **Fixed Double Threshold** et **Dynamic Double Threshold** sont bien meilleures que l'approche Single Threshold et permettent de minimiser le nombre de migrations.

Le tableau 4.3 compare entre les trois approches (**Single Threshold**, **Fixed Double Threshold** et **Dynamic Double Threshold**). Nous remarquons que les approches proposées réduisent le nombre de migration des VMs par rapport à l'approche **single Threshold**.

Nombre de VMs	29	58	87	116	145	174	203	232	261	290
Single Threshold	4589	10885	16807	22970	29323	35502	41896	48240	54692	61072
Fixed Double Threshold	2060	5638	10049	15038	18630	25452	29861	34003	40886	44736
Dynamic Double Threshold	415	1446	1976	2674	3715	4363	5291	6714	7448	9346

Table 4.3 – *Comparaison entre le nombre de migrations des trois approches*

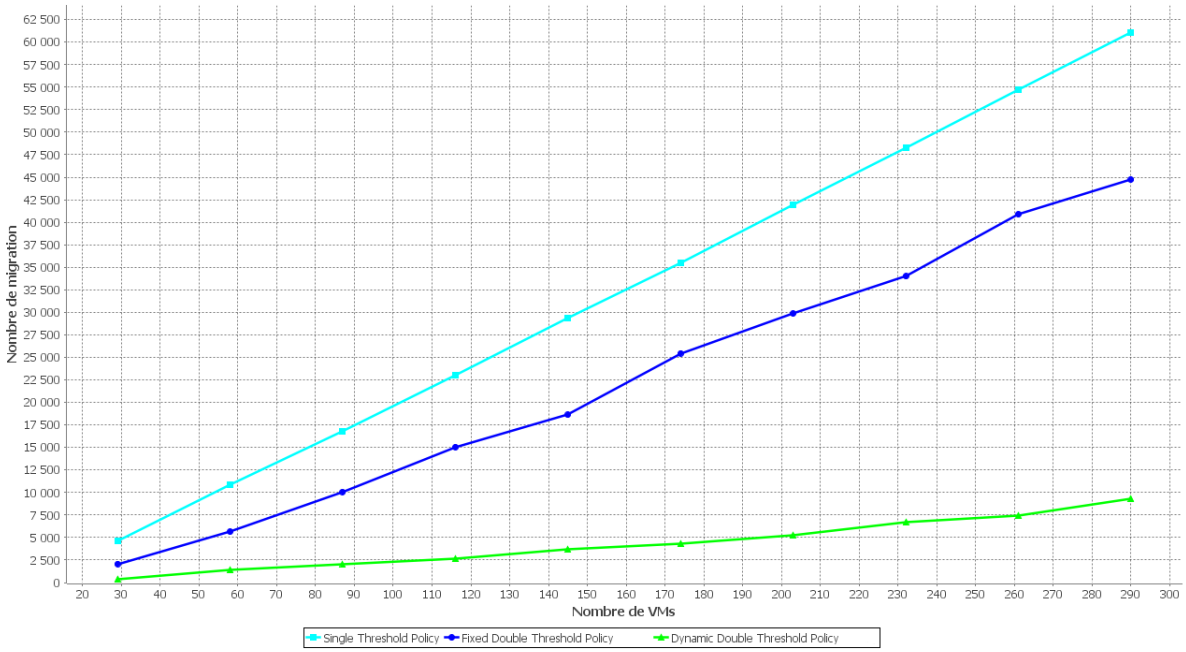

Figure 4.11 – *Influence du nombre des VMs sur le nombre de migrations*

Le tableau 4.4 montre la différence entre les trois approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit le nombre de migrations avec un gain moyen de 87,38% par rapport à l'approche **Single Threshold** et un gain moyen de 80,31% par rapport à l'approche **Fixed Double**

Threshold.

Nom de l'approche	Gain
Single Threshold et Dynamic Double Threshold	87,38%
Fixed Double Threshold et Dynamic Double Threshold	80,31%

Table 4.4 – *Comparaison entre le nombre de migrations des trois approches*

4.5.1.3. Violation de SLA

La Figure 4.12 résultante du tableau 4.5 montre le comportement des deux approches par rapport au nombre de violations de SLA. Nous remarquons que les courbes des approches **Fixed Double Threshold** et **Dynamic Double Threshold** se trouvent au-dessous de la courbe de l'approche **Single Threshold**. Nous remarquons aussi qu'avec l'augmentation du nombre de VMs, la différence entre les courbes augmente. Donc les approches **Fixed Double Threshold** et **Dynamic Double Threshold** permettent de minimiser le nombre de violations de SLA.

Nombre de VMs	29	58	87	116	145	174	203	232	261	290
Single Threshold	4786	12420	20010	27810	36461	44241	52839	61543	70830	79590
Fixed Double Threshold	3023	8827	16125	23449	30604	40439	48878	55547	67278	75000
Dynamic Double Threshold	993	3840	6402	8112	12821	15182	17325	22882	26316	29488

Table 4.5 – *Comparaison entre le nombre de violations de SLA des trois approches*

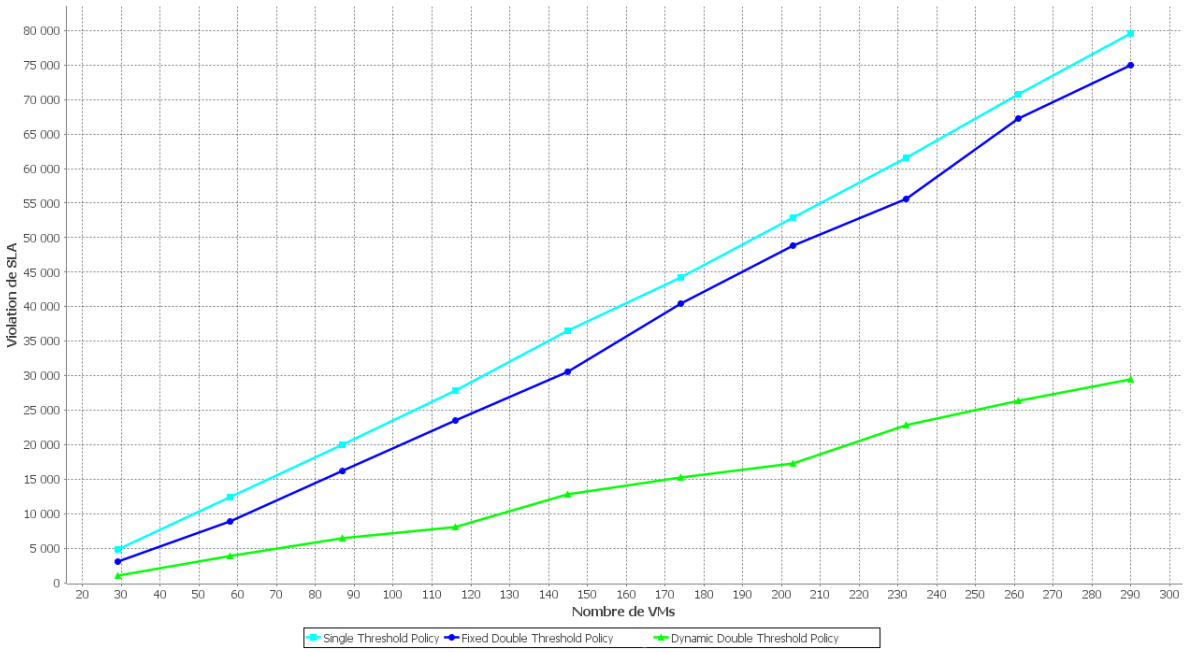

FIGURE 4.12 – *Influence du nombre des VMs sur le nombre de violations de SLA*

Le tableau 4.6 montre la différence entre les trois approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit le nombre de violation de SLA avec un gain moyen de 67,35% par rapport à l'approche **Single Threshold** et un gain moyen de 61,48% par rapport à l'approche **Fixed Double Threshold**.

Nom de l'approche	Gain
Single Threshold et Dynamic Double Threshold	67,35%
Fixed Double Threshold et Dynamic Double Threshold	61,48%

TABLE 4.6 – *Comparaison entre le nombre de violations de SLA des trois approches*

Le tableau 4.7 montre que l'approche **Dynamic Double Threshold** a un pourcentage moyen de violation de SLA moins que l'approche **Single Threshold** et l'approche **Fixed Double Threshold**.

Nombre de VMs	29	58	87	116	145	174	203	232	261	290	Moyen
Single Threshold (%)	40.83	41.51	44.80	47.39	48.33	48.71	49.33	49.54	48.89	47.81	46.61
Fixed Double Threshold (%)	12.92	17.11	20.32	20.48	22.22	21.92	24.41	25.65	24.24	26.82	21.90
Dynamic Double Threshold (%)	6.28	7.37	7.82	7.48	8.57	8.25	10.49	11.40	10.25	10.35	9.12

Table 4.7 – *Comparaison entre le pourcentage de violation de SLA des trois approches*

4.5.2. Expérience 2 : Variation du nombre de hôtes

Pour étudier l'impact du nombre d'hôtes sur la consommation d'énergie, le nombre de migrations et le nombre de violations des SLA, nous avons mesuré les trois principales métriques en appliquant les quatre approches. Nous avons créé un Data Center avec un nombre d'hôtes qui varie de 100 à 280 hôtes avec un pas de 20 où chaque hôte est caractérisée par 1 processeur avec une vitesse variante en MIPS (1000, 2000 ou 3000), 8GB de mémoire, 1TB de stockage et consomme une énergie de 175 W avec 0% d'utilisation du processeur jusqu'à 250 W avec 100% d'utilisation du processeur. Le Broker crée pour chaque simulation 290 VMs hétérogènes. Chaque VM possède 1 processeur avec une vitesse variante en MIPS (250, 500, 750 ou 1000), 128MB de mémoire, 1GB de stockage. Nous avons considéré le seuil de l'approche **Single Threshold** comme 0.6 et les deux seuils supérieur et inférieur de l'approche **Fixed Double Threshold** comme 0,6 et 0,3 respectivement.

4.5.2.1. Consommation d'énergie

Pour analyser la consommation d'énergie, les résultats obtenus par les simulations sont résumés numériquement dans le tableau 4.8 et schématisés par la Figure 4.13. Nous remarquons durant cette série de simulations que les courbes de nos approches se trouvent au-dessous des courbes des autres approches. Donc nos approches sont bien meilleures que les autres et permettent d'économiser l'énergie consommée.

Nombre de PMs	100	120	140	160	180	200	220	240	260	280
Approche sans migration	35.94	32.78	31.34	30.13	28.79	28.68	28.70	28.70	28.67	28.65
Single Threshold	29.96	30.10	28.13	26.05	25.37	24.99	24.38	23.84	23.38	23.27
Fixed Double Threshold	21.70	20.57	19.98	19.43	19.13	19.04	19.05	19.06	19.14	19.14
Dynamic Double Threshold	17.71	16.67	16.60	16.46	16.40	16.39	16.46	16.28	16.42	16.23

TABLE 4.8 – *Comparaison entre la consommation d'énergie des quatre approches*

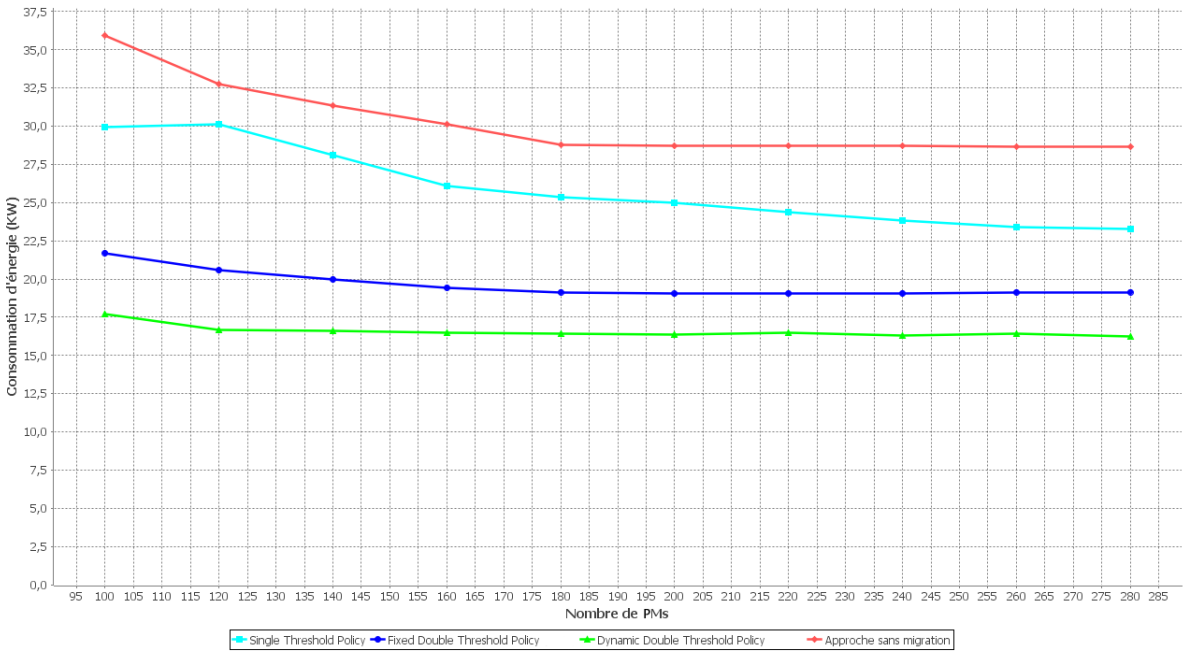

FIGURE 4.13 – *Influence du nombre des PMs sur l'énergie*

Le tableau 4.9 montre la différence entre les quatre approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit l'énergie avec un gain moyen de 45,01% par rapport à l'approche sans migration, un gain moyen de 35,67% par rapport à l'approche **Single Threshold** et un gain moyen de 15,53% par rapport à l'approche **Fixed Double Threshold**.

Nom de l'approche	Gain
Approche sans migration et Dynamic Double Threshold	45,01%
Single Threshold et Dynamic Double Threshold	35,67%
Fixed Double Threshold et Dynamic Double Threshold	15,53%

Table 4.9 – *Comparaison entre la consommation d'énergie des quatre approches*

4.5.2.2. Nombre de migration

D'après la Figure 4.14 et le tableau 4.10, nous pouvons remarquer que nos approches **Fixed Double Threshold** et **Dynamic Double Threshold** ont réduit le nombre de migration par rapport à l'approche **Single Threshold**.

Nombre de PMs	100	120	140	160	180	200	220	240	260	280
Single Threshold	57883	61306	62329	62475	62417	62363	62432	62171	62161	61992
Fixed Double Threshold	20117	20269	20271	19929	18984	19916	19811	20009	19864	19403
Dynamic Double Threshold	4105	4273	3963	3867	3937	3881	3602	4222	4076	4071

Table 4.10 – *Comparaison entre le nombre de migration des quatre approches*

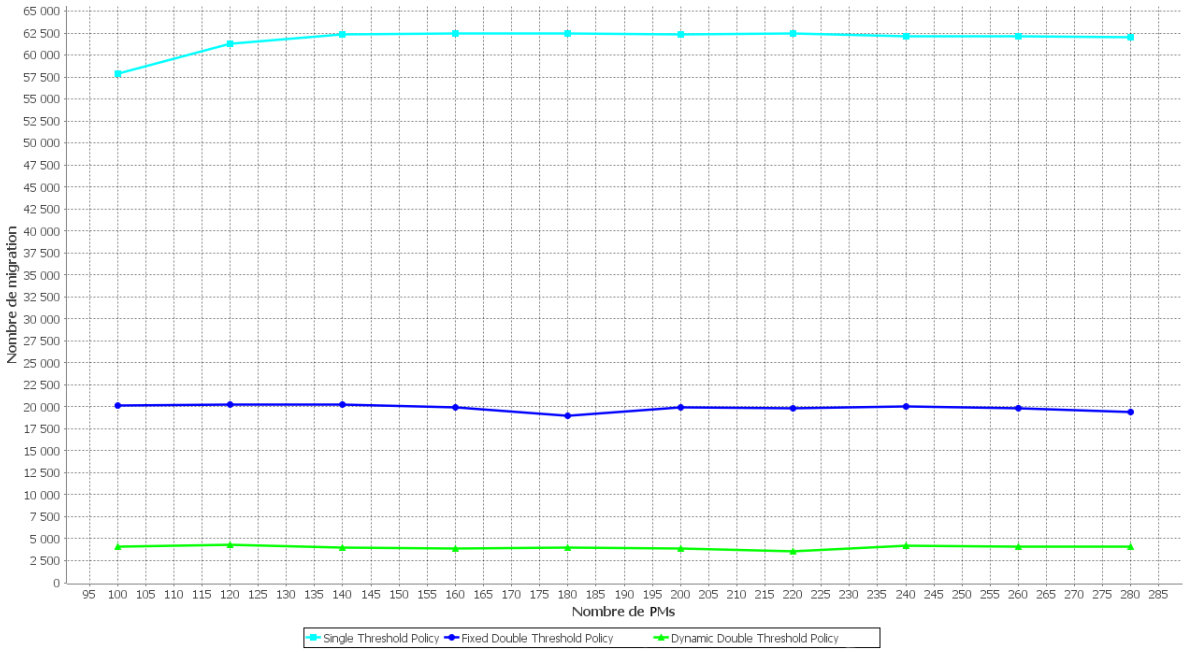

Figure 4.14 – *Influence du nombre des PMs sur le nombre de migration*

Le tableau 4.11 montre la différence entre les trois approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit le nombre de migration avec un gain moyen de 93,51% par rapport à l'approche **Single Threshold** et un gain moyen de 79,85% par rapport à l'approche **Fixed Double Threshold**.

Nom de l'approche	Gain
Single Threshold et Dynamic Double Threshold	93,51%
Fixed Double Threshold et Dynamic Double Threshold	79,85%

Table 4.11 – *Comparaison entre le nombre de migration des trois approches*

4.5.2.3. Violation de SLA

La Figure 4.15 résultante du tableau 4.12 montre que les approches **Fixed Double Threshold** et **Dynamic Double Threshold** permettent de minimiser le nombre de violation de SLA par rapport à l'approche **Single Threshold**.

Nombre de PMs	100	120	140	160	180	200	220	240	260	280
Single Threshold	62422	63861	64238	64087	64043	63792	64065	63723	63669	63346
Fixed Double Threshold	33563	33200	32658	32385	31741	32394	32847	32401	32396	31655
Dynamic Double Threshold	12025	12637	11739	11237	10863	11062	10237	12238	11834	11832

Table 4.12 – *Comparaison entre le nombre de violations de SLA des quatre approches*

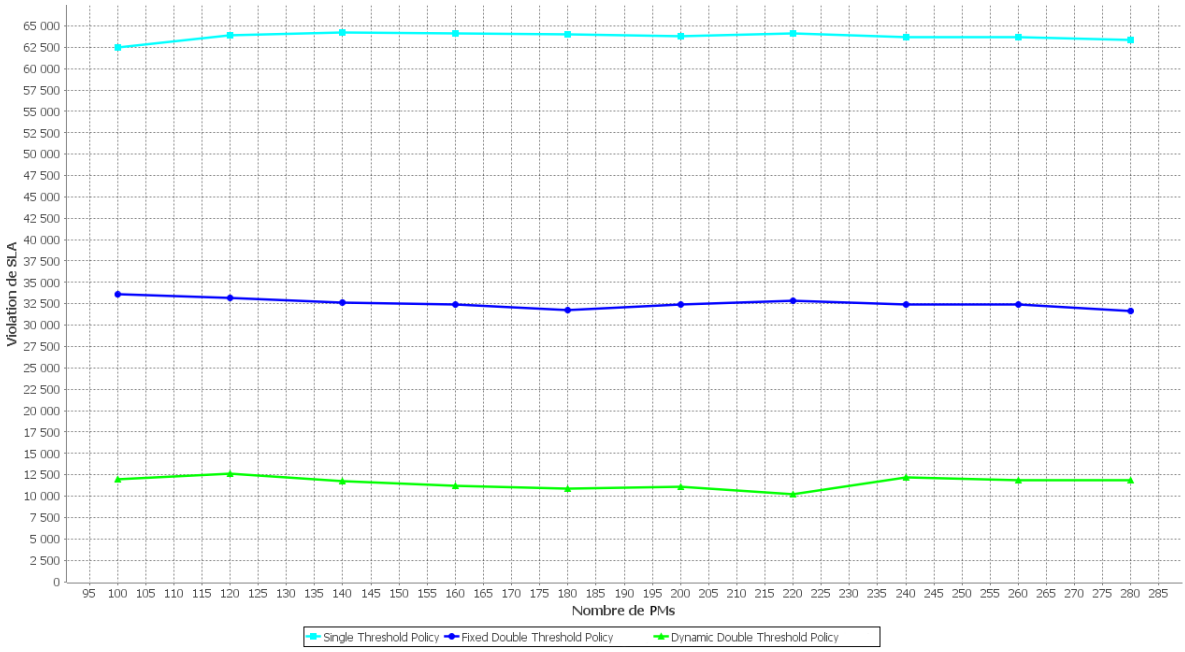

Figure 4.15 – *Influence du nombre des PMs sur le nombre de violations de SLA*

Le tableau 4.13 montre la différence entre les trois approches, où nous pouvons déduire que notre approche **Dynamic Double Threshold** a réduit le nombre de violation de SLA avec un gain moyen de 81,83% par rapport à l'approche **Single Threshold** et un gain moyen de 64,42% par rapport à l'approche **Fixed Double Threshold**.

Nom de l'approche	Gain
Single Threshold et Dynamic Double Threshold	81,83%
Fixed Double Threshold et Dynamic Double Threshold	64,42%

Table 4.13 – *Comparaison entre le nombre de violation de SLA des trois approches*

Le tableau 4.14 montre que l'approche **Dynamic Double Threshold** a un pourcentage moyen de violation de SLA moins que l'approche **Single Threshold** et l'approche **Fixed Double Threshold**.

Nombre de PMs	100	120	140	160	180	200	220	240	260	280	
Single Threshold	47.81	49.74	49.87	50.16	49.99	49.87	49.58	49.80	49.55	49.68	49.60
Fixed Double Threshold	26.82	25.29	25.08	24.33	24.92	25.04	25.57	24.87	25.75	26.18	25.21
Dynamic Double Threshold	9.60	10.00	9.60	10.17	9.70	9.68	8.83	9.90	9.39	9.77	9.66

Table 4.14 – *Comparaison entre le pourcentage de violation de SLA des trois approches*

4.6. Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre application ainsi que les résultats obtenus. Aussi, nous avons réalisé plusieurs séries de simulations dans le but de comparer nos approches avec l'approche sans migration et l'approche **Single Threshold** tout en variant différents paramètres comme : le nombre d'hôtes, le nombre de VM. Les résultats de la comparaison ont montré que notre approche permet de minimiser la consommation d'énergie du système, réduire le nombre de migrations des VMs et par conséquence réduire le nombre de violations de SLA.

Conclusion générale

Le cloud computing est en pleine expansion et tend à s'imposer comme un des paradigmes dominants dans l'univers informatique. Les infrastructures proposant des services de cloud computing deviennent donc de plus en plus nombreuses, et de plus en plus complexes pour répondre à cette demande croissante de services décentralisés. Cette augmentation amène bien évidemment divers problèmes, dont celui de la consommation d'énergie. Il faut donc concevoir des techniques et outils afin de répondre à ces nouveaux besoins de gestion.

La migration des machines virtuelles est une technique qui permet de régler en général certains problèmes dans le Cloud tels que les problèmes de la consommation d'énergie.

Cette technique consiste à déplacer une machine virtuelle d'un nœud à un autre dont le but d'augmenter considérablement le taux d'utilisation et réduire la consommation d'énergie selon certains critères.

Au cours de ce projet, nous avons développé une stratégie pour bien améliorer l'efficacité énergétique dans le Cloud Computing. Nous avons proposé une approche qui se base sur le mécanisme de la migration des machines virtuelles, tout en appliquant certaines méthodes. Et afin de mettre en évidence l'approche proposée, nous avons réalisé plusieurs séries d'expérimentations en faisant varier plusieurs paramètres. Nous avons utilisé, également, les trois principales métriques qui sont l'énergie consommée, nombre de migrations et violations de SLA Sur le plan d'expérimentation, nous avons positionné et comparé nos propositions par rapport à l'approche sans migration et à l'approche *Single Threshold*.

Perspectives

Pour une continuation de notre travail, plusieurs perspectives peuvent être envisagées :

1. Ajouter un seuil de température afin de limiter la chaleur dégagée des machines physiques et de minimiser l'énergie consommée des serveurs et des systèmes de refroidissement.
2. Étudier l'influence des systèmes de climatisation des data center sur leur efficacités et leur performances.
3. Implémenter les deux approches proposées dans un environnement de Cloud réel.
4. En plus de l'utilisation du processeur, prendre en considération plusieurs ressources au niveau de la phase de migration tels que : la capacité de la RAM, la capacité de stockage et la bande passante.

Table des figures

Liste des tableaux

Liste des Algorithmes

Bibliographie

[AGE07] US Environmental Protection AGENCY. Report to congress on server and data center energy, août 2007. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Report_Exec_Summary_Final.pdf consulté le 3/02/2014.

[BB00] T Burd and R Brodersen. Design issues for dynamic voltage scaling. In *Proceedings of the 2000 international symposium on Low power electronics and design*, pages 9–14, New York, NY, USA, 2000.

[BB10] A Beloglazov and R Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, number 4, New York, NY, USA, 2010.

[BBB+91] D H Bailey, E Barszcz, J T Barton, D S Browning, R L Carter, L Dagum, R A Fatoohi, P O Frederickson, T A Lasinski, R S Schreiber, H D Simon, V Venkatakrishnan, and S K Weeratunga. The nas parallel benchmarks, summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, Supercomputing '91, pages 158–165, New York, NY, USA, 1991.

[BDF+03] P Barham, Boris Dragovic, K Fraser, S Hand, T Harris, A Ho, R Neugebauer, I Pratt, and A Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003.

[CFH+05] C Clark, K Fraser, S Hand, J G Hansen, E Jul, C Limpach, I Pratt, and A Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design &*

Implementation, volume 2 of *NSDI'05*, pages 273–286, Berkeley, CA, USA, 2005.

[CR10] R N Calheiros and R Ranjan. Cloudsim : A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, 2010.

[DGH+] C Dupont, G Giuliani, F Hermenier, T Schulze, and A Somov. An energy aware framework for virtual machine placement in cloud federated data centres. In *roceedings of the 3rd International Conference on Future Energy Systems*, e-Energy 2012.

[Dre08] U Drepper. The cost of virtualization. *Queue*, pages 28–35, January 2008.

[DSI10] T V T Duy, Y Sato, and Y Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *Parallel kamp ; Distributed Processing, Workshops and Phd Forum (IPDPSW) ,2010 IEEE International Symposium on*, pages 1–8, April 2010.

[EPA07] Report to congress on server and data center energy efficiency : Public law 109- 431. 2007.

[FW07] L A B Xiaobo Fan and W D Weber. Power provisioning for a warehousesized computer. In *Proceedings of the 34th Annual Intl. Symp. On Computer Architecture*, page 41, June 2007.

[GCNS08] C Gunaratne, K Christensen, B Nordman, and S Suen. Reducing the energy consumption of ethernet with adaptive link rate (alr). *IEEE Transactions on Computers*, 57(4) :448–461, 2008.

[GLA12] J GLANZ. The cloud factories : Power, pollution and the internet. *New York Times*, Septembre 2012. http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html consulté le 28/01/2014.

[HF09] S Huang and W Feng. Energy-efficient cluster computing via accurate workload characterization. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 68–75, Washington, DC, USA, 2009.

[HHN08] J Hikita, A Hirano, and H Nakashima. Saving 200kw and $200 k/year by power-aware job/machine scheduling. In *Parallel and Distributed Processing Symposium, International*, pages 1–8, 2008.

[HLM+09] F Hermenier, X Lorca, J M Menaud, G Muller, and J Lawall. Entropy : a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/ SIGOPS international conference on Virtual execution environments*, VEE '09, pages 41–50, New York, NY, USA, 2009.

[HSN10] M Hoyer, K Schröder, and W Nebel. Statistical static capacity management in virtualized data centers supporting fine grained qos specification. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 51–60, New York, NY, USA, 2010.

[IPE10] IPETI. definition de langage java java script, 2010. consulté le 10/03/2014 sur http://ipeti.forumpro.fr/t21-definition-de-langage-java-java-script.

[Kar09] S Karayi. Rapport sur l'efficacité énergétique des serveurs de 1e, 2009. http://www.1e.com/download/9291_1E_Server_FR.pdf consulté le 20/02/2014.

[KBSW09] J Koomey, S Berard, M Sanchez, and H Wong. Assessing trends in the electrical efficiency of computation over time. *IEEE Annals of the History of Computing*, August 2009. http://download.intel.com/pressroom/pdf/computertrendsrelease.pdf consulté le 25/01/2014.

[KKJ08] D Kusic, J O Kephart, and G Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 34th Annual Intl. Symp. On Computer Architecture*, page 41, September 2008.

[Koo08] J G Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3) :034008 (8pp), 2008.

[KOWN10] K Kurowski, A Oleksiak, M Witkowski, and J Nabrzyski. Distributed power management and control system for sustainable computing environments. In *Proceedings of the International Conference on Green Computing*, GREENCOMP '10, pages 365–372, Washington, DC, USA, 2010.

[LH12] K L Teh Lee and H Huang. Dynamic resource management for energy saving in the cloud computing environment. In *Proceedings of the International Conference on Information Science and Industrial Applications*, volume 04, pages 176–181, May 2012.

[LO10] L Lefèvre and A C Orgerie. Designing and evaluating an energy efficient cloud. In *Journal of Supercomputing*, number 51, pages 352–373, 2010.

[LWL+09] L Liu, H Wang, X Liu, X Jin, W B He, Q B Wang, and Y Chen. Greencloud : a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, ICAC-INDST '09, pages 29–38, New York, NY, USA, 2009.

[NCS09] D Niyato, S Chaisiri, and L B Sung. Optimal power management for server farm to support green computing. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, Washington, DC, USA, 2009.

[RC10] R R Rodrigo and N Calheiros. Cloudsim : A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Technical report, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, 2010.

[SBW+] B Schäppi, T Bogner, B Przywara S Weeren, F Bellosa, A Anglade, and B Harrison. Guide d'achat et de gestion des équipements et des infrastructures pour des serveurs sobres en énergie.

[SKM08] A Singh, M Korupolu, and D Mohapatra. Server-storage virtualization : integration and load balancing in data centers. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 1–53, Piscataway, NJ, USA, 2008.

[SKZ08] S Srikantaiah, A Kansal, and F Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008.

[SP11] H D Richa Sinha and N Purohit. Energy efficient dynamic integration of thresholds for migration at cloud data centers. *Special Issue of International Journal of Computer Applications on Communication and Networks*, (11), Décembre 2011.

[SPD11] R Sinha, N Purohit, and H Diwanji. Power aware live migration for data centers in cloud using dynamic threshold. *International Journal of Computer Technology and Applications*, 2(6) :2041–2046, 2011.

[SRWM10] J H Schönherr, J Richling, M Werner, and G Mühl. Event-driven processor power management. In *Proceedings of the 1st International Conference on Energy- Efficient Computing and Networking*, volume e-Energy '10, pages 61–70, New York, NY, USA, 2010.

[VBVB09] W Voorsluys, J Broberg, S Venugopal, and R Buyya. Cost of virtual machine live migration in clouds : A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009.

[Voo09] W Voorsluys. Cost of virtual machine live migration in clouds : A performance evaluation. In *Proceedings of the 1st international conference on Cloud Computing*, pages 254–265, 2009.

[WEBa] Le green it : Le power usage effectiveness (pue). http://www.greenvision.fr/menu-greenit/menu-pue consulté le 5/03/2014.

[WEBb] Les opportunités de réduction de consommation d'énergie de vos salles serveurs. http://www.sequovia.com/les-opportunites-de-reduction-de-consommation-d-energie-de-vos-salles-se php consulté le 10/02/2014.

[WEBc] Top 500 supercomputer sites. top 500 supercomputer list of june 2012. http://www.top500.org/lists/2012/06.

[WEB00] Specweb99 benchmark, 1999-2000. http://www.spec.org/web99/ consulté le 05/04/2014.

[WEB10] Greenpeace international. make it green : Cloud computing and its contribution to climate change, 2010. http://www.greenpeace.org/usa/en/mediacenter/reports/makeit-greencloudcomputing/ consulté le 16/02/2014.

[WEB11] Datacenters et développement durable : Etat de l'art et perspectives. Syntec Numérique, 2011.

[WEB13] International atomic energy agency. power reactor information system, Jan 2013. http://pris.iaea.org/PRIS/CountryStatistics/CountryDetails.aspx?current=FR consulté le 18/03/2014.

[Wik14] Wikipedia. Java„ 2014. consulté le 14/04/2014 sur http://fr.Wikipédia.org/wiki/Java_(langage).

[ZMM04] D Zhu, R Melhem, and D. Mosse. The effects of energy management on reliability in real-time embedded systems. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, ICCAD '04, pages 35–40, Washington, DC, USA, 2004.