MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Réseaux et Systèmes**

Présenté par :

- **Larbi Abdelillah Islam**
- **Ghomri Habib Anis**

THEME :

Optimisation et comparaison d'une technique de migration dans un environnement Cloud Computing

Soutenu le : 03/07/2022

Devant le jury composé de :

Mr. Moussa M.	Grade	Université de Mostaganem	Président
Mme. Meroufel B.	Grade	Université de Mostaganem	Examinateur
Dr. Filali F. Z.	Grade	Université de Mostaganem	Encadreur

Année Universitaire 2021-2022

Résumé

La virtualisation est un outil utile pour les administrateurs de datacenters et de clusters, c'est une technique qui permet à plusieurs systèmes d'exploitation de fonctionner simultanément sur une seule PM, elle est devenue un aspect essentiel dans les serveurs modernes et des datacenters grâce à plusieurs avantages tels que le partage souple et efficace des ressources, la tolérance aux pannes et la portabilité.

La migration est une des capacités importantes de la virtualisation des systèmes, qui permet aux entités d'être migrées, de manière transparente, ainsi que leurs environnements d'exécution à travers des PMs, des serveurs ou des datacenters, Il existe principalement deux types de migration : la migration à froid, qui nécessite la mise hors tension de la machine virtuelle lors de l'opération de migration pendant laquelle la machine virtuelle est inaccessible. La migration à chaud ou en direct, pour laquelle la machine virtuelle n'est pas mise hors tension et reste donc accessible**.**

Mots-clés : Cloud Computing, Virtualisation, Machines Virtuelles, Migration.

Abstract

Virtualisation is a useful tool for datacentre and cluster administrators, it is a technique that allows multiple operating systems to run simultaneously on a single PM, it has become an essential aspect in modern servers and datacentres due to several advantages such as flexible and efficient resource sharing, fault tolerance and portability.

Migration is one of the important capabilities of system virtualisation, which allows entities to be seamlessly migrated along with their execution environments across PMs, servers or datacenters. There are mainly two types of migration: cold migration, which requires the virtual machine to be powered down during the migration operation, while the virtual machine is inaccessible. Hot or live migration, where the virtual machine is not powered down and therefore remains accessible.

Keywords: Cloud computing, Virtualisation, Virtual Machines, Migration.

Dédicaces

Je dédie ce travail à mes très chers parents, pour leur présences, soutien moral permanent et leur amour infini tout au long de ma vie, ce travail est le fruit de leurs sacrifices qu'ils ont consentis pour mon éducation et ma formation.

À mes deux petites sœurs ***Malek*** et ***Yousra***.

À mon binôme pour sa bonne collaboration et son dévouement.

À Tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.

Larbi Abdelillah Islam

Je dédie cet ouvrage,

À ma maman qui m'a soutenu et encouragé durant ces années d'études, qu'elle trouve ici le témoignage de ma profonde reconnaissance.

À ma tente ou plutôt ma deuxième maman, mon père, mes frères et sœurs qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail, ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

À tous mes amis qui m'ont toujours encouragé, surtout mon binôme avec lequel j'ai partagé des moments inoubliables, et à qui je souhaite plus de succès et de réussite dans leur vie.

Ghomri Habib Anis

Remerciements

Nous remercions ALLAH de nous avoir donné la force et la capacité qui nous ont menées à ce niveau.

Nous tenons à exprimer notre profonde gratitude et reconnaissance à notre encadrante, Dr. Filali F. Z., nous vous remercions d'avoir bien assurer la direction et l'encadrement de notre travail malgré vos nombreuses occupations, nous vous remercions pour vos qualités humaines, votre précieuse attention, implication et vos précieux conseils et orientations qui ont mené à l'aboutissement de ce travail.

Nous remercions aussi les membres de jury pour avoir bien voulu donner de leur temps pour lire et juger notre projet.

Merci aussi à tous nos amis, nos collègues, et à tous ceux qui nous ont aidé de près ou de loin. Nous leur exprimons notre profonde sympathie et leur souhaitons beaucoup de bien.

Nous n'oublierons pas non plus de remercier toutes les personnes que nous avons pu côtoyer pendant ces cinq ans à la faculté des sciences exactes et d'informatique pour leur soutien moral et amical.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète
BP	Bande Passante
CPU	Central Processing Unit
IDE	Integrated Development Environment
JRE	Java Runtime Environmrnt
JVM	Java Virtual Machine
MIPS	Millions Instruction Per Second
OS	Operating System
PM	Physiscal Machine
QoS	Quality of Service
RAM	Random Access Memory
SE	Système d'éxploitation
SLA	Service Level Agreement
VM	Virtual Machine
VMM	Virtual Machine Manager

Table des matières

Introduction Générale

Le contexte :

Le Cloud Computing est un terme général employé pour désigner la livraison de ressources et de services à la demande par internet. Il désigne le stockage et l'accès aux données par l'intermédiaire d'internet plutôt que via le disque dur d'un ordinateur.

De manière générale, on parle de Cloud Computing lorsqu'il est possible d'accéder à des données ou à des programmes depuis internet, ou tout du moins lorsque ces données sont synchronisées avec d'autres informations sur internet. Il suffit donc pour y accéder de bénéficier d'une connexion internet.

L'image du Cloud est utilisée de façon métaphorique pour désigner internet. Cette comparaison date de l'époque à laquelle on représentait les infrastructures gigantesques des fermes de serveurs internet sous la forme d'un grand nuage blanc, acceptant les connexions et distribuant des informations tout en flottant.

Cette technologie permet aux entreprises d'acheter des ressources informatiques sous la forme de service, de la même manière que l'on consomme de l'électricité, au lieu d'avoir à construire et entretenir des infrastructures informatiques en interne.

Selon U.S. National Institute of Standards and Technology, le Cloud Computing est un modèle permettant d'établir un accès à la demande en réseau vers un bassin partagé de ressources informatiques configurable. Ces ressources sont par exemple des réseaux, des serveurs, de l'espace de stockage, des applications et des services. Elles peuvent être approvisionnées rapidement avec un effort de gestion et une interaction avec le fournisseur de services minimes. Le modèle Cloud met en avant la disponibilité, et se compose de cinq

caractéristiques essentielles, trois modèles de livraisons (SaaS, PaaS, IaaS), et quatre modèles de déploiement (Privé, Communautaire, Public, Hybrid). [1]

Le modèle Cloud Computing se différencie par cinq caractéristiques essentielles :

- Accès aux services par l'utilisateur à la demande.
- Accès réseau large bande.
- Réservoir de ressources (non localisées).
- Redimensionnement rapide (élasticité).
- Facturation à l'usage.

La problématique :

La demande croissante d'infrastructure dans les environnements de Cloud Computing nécessite la garantie d'un niveau de service minimal afin d'assurer le bon fonctionnement du service en qualité de performance, de disponibilité, de sécurité, ... A cet égard, les fournisseurs indiquent généralement ces valeurs dans un contrat appelé SLA (Service Level Agreement) sous forme d'accords de qualités de services (QoS).

Le présent travail est une étude de la migration de services dans les infrastructures Cloud sous ses différents aspects, avec comme objectif de proposer un algorithme qui permet la migration de machines virtuelles afin d'optimiser le temps d'indisponibilité des services Cloud et économiser l'énergie utilisée. Dans ce qui suit, nous allons présenter, étudier et comparer les techniques de migration existante afin de parvenir à proposer une approche optimale qui nous offre les meilleures performances possibles.

Organisation du travail

Ce document est structuré autour de quatre chapitres et une conclusion générale comme suit :

Chapitre 1 : dans ce chapitre nous présenterons en détails les concepts de base du Cloud Computing en montrant des généralités sur la technologie de virtualisation ainsi que

son fonctionnement et ses différentes techniques. Nous allons aussi présenter le principe des machines virtuelles et les différents types d'hyperviseurs.

Chapitre 2 : ce chapitre sera consacré à la présentation de la migration des machines virtuelles dans les environnements de Cloud Computing et ses techniques. Ensuite, un état de l'art sur les approches de migrations de machines virtuelles sera abordé. Nous conclurons par une comparaison entre ces dernières.

Chapitre 3 : Ce chapitre comprend analyse de l'existant, décrit l'architecture du modèle proposé, les métriques utilisées et qui sont nécessaires à la comparaison des résultats obtenus avec d'autres approches existantes. Nous présenterons dans le même chapitre les algorithmes des deux phases « sélection et allocation » pour décrire le principe de fonctionnement de notre approche.

Chapitre 4 : Dans ce chapitre qui conclue notre travail, nous présenterons le langage de programmation Java et l'environnement de développement Eclipse. Nous présenterons par la suite un aperçu sur le simulateur CloudSim, son architecture ainsi que ses classes fondamentales. Ensuite nous ferons l'analyse et discuterons des résultats que nous avons obtenu suite aux séries de simulations de notre approche dont il est question d'étudier les performances.

Conclusion générale : Nous clôturons ce projet par une conclusion et quelques perspectives.

Chapitre 1

Concepts de Base

1.1 Introduction

La grande majorité des environnements cloud performants sont basés sur une infrastructure virtualisée. Cela fait déjà plusieurs années que la virtualisation est utilisée avec succès dans les centres de données en tant que stratégie de consolidation des serveurs. Utilisée de façon plus large pour mettre en commun les ressources d'infrastructure, la virtualisation peut également fournir les éléments constitutifs élémentaires requis pour améliorer l'agilité et la flexibilité d'un environnement cloud. [2]

Il est facile de confondre virtualisation et Cloud Computing, car il s'agit dans les deux cas d'une création d'environnements utiles à partir de ressources abstraites. La virtualisation est une technologie qui permet de créer plusieurs environnements simulés ou ressources spécialisées à partir d'un seul système physique. Les Clouds sont des environnements qui dissocient, regroupent et partagent des ressources évolutives sur un réseau. Pour faire simple, la virtualisation est une technologie alors que le cloud est un environnement. [3]

.

1.2 Concepts de Virtualisation

1.2.1 Virtualisation

1.2.1.1 Définition

La virtualisation est la création d'une version virtuelle – plutôt que réelle – de quelque chose, comme un système d'exploitation (OS), un serveur, un périphérique de stockage ou des ressources réseau.

La virtualisation utilise un logiciel qui simule les fonctionnalités du matériel afin de créer un système virtuel. Cette pratique permet aux organisations informatiques d'exploiter plusieurs systèmes d'exploitation, plus d'un système virtuel et diverses applications sur un seul serveur. Les avantages de la virtualisation comprennent une plus grande efficacité et des économies d'échelle.

La virtualisation du système d'exploitation est l'utilisation d'un logiciel permettant à une pièce de matériel d'exécuter plusieurs images de système d'exploitation en même temps. Cette technologie a fait ses débuts sur les ordinateurs centraux il y a plusieurs décennies, permettant aux administrateurs d'éviter de gaspiller une puissance de traitement coûteuse. [4]

1.2.1.2 Historique

Bien que le concept de virtualisation soit apparu dans les années 1960, ce n'est que dans les années 2000 que son utilisation s'est généralisée. Les technologies qui sont à la base de la virtualisation, telles que les hyperviseurs, ont été développées il y a plusieurs dizaines d'années afin de permettre à divers utilisateurs d'accéder simultanément aux ordinateurs qui effectuaient des traitements par lots. Le traitement par lots, qui permettait d'exécuter très rapidement des tâches répétitives des milliers de fois (telles que la paie), était une technique très répandue dans le monde de l'entreprise

Toutefois, au fil des décennies, d'autres solutions qui permettaient l'utilisation simultanée d'une seule et même machine par de nombreux utilisateurs ont gagné en

popularité, tandis que la virtualisation restait en marge. L'une de ces solutions était la technique du temps partagé, qui isolait les utilisateurs au sein des systèmes d'exploitation. Elle a notamment conduit à la création d'autres systèmes d'exploitation comme UNIX, puis Linux. Pendant ce temps, la virtualisation restait une technologie de niche très peu utilisée.

La flexibilité de la virtualisation a permis de limiter la dépendance vis-à-vis d'un fournisseur et a jeté les bases du Cloud Computing. Aujourd'hui, elle est si répandue dans les entreprises qu'un logiciel de gestion de la virtualisation est souvent nécessaire pour en assurer le suivi. [5]

1.2.1.3 Fonctionnement de la virtualisation

Des logiciels, appelés hyperviseurs, isolent les ressources physiques des environnements virtuels, qui nécessitent ces ressources. Ces hyperviseurs peuvent reposer sur un système d'exploitation (ordinateur portable, par exemple) ou être directement installés sur un système physique (tel qu'un serveur), ce qui est l'option la plus souvent choisie par les entreprises qui ont recours à la virtualisation. Les hyperviseurs répartissent les ressources physiques pour permettre aux environnements virtuels de les utiliser.

Ces ressources sont partitionnées à partir de l'environnement physique et distribuées aux différents environnements virtuels. Les utilisateurs interagissent avec ces environnements (également appelés machines virtuelles ou hôtes) et y exécutent des calculs. La machine virtuelle opère comme un fichier de données unique. Comme n'importe quel fichier numérique, on peut la transférer d'un ordinateur à un autre, l'ouvrir sur l'un ou l'autre et l'utiliser de la même manière.

Lorsque l'environnement virtuel est exécuté et qu'un utilisateur ou un programme émet une instruction nécessitant des ressources supplémentaires à partir de l'environnement physique, l'hyperviseur transmet cette requête au système physique et met en cache les modifications. Le processus est presque aussi rapide que sur un système natif.

1.2.2 Machine Virtuelles

1.2.2.1 Définition

Une machine virtuelle ou VM est un environnement entièrement virtualisé qui fonctionne sur une machine physique. Elle exécute son propre système d'exploitation (SE) et bénéficie des mêmes équipement qu'une machine physique : CPU, mémoire RAM, disque dur et carte réseau. [6]

La machine virtuelle est utile dans différents domaines tels que les suivants :

- La création et le déploiement des applications dans le cloud ;
- L'essai d'un récent système d'exploitation (SE) comprenant les versions bêta ;
- La mise en œuvre d'un nouvel environnement afin de faciliter et d'accélérer par les développeurs les scénarios de développement et de test ;
- La sauvegarde de votre système d'exploitation ;
- L'exécution d'une ancienne application et l'accès à des données infectées par des virus ;
- L'application ou l'exécution de logiciels sur des systèmes d'exploitation non prévus à cela à l'origine. [7]

1.2.2.2 Fonctionnement

La technologie de virtualisation permet de partager un système avec de nombreux environnements virtuels. L'hyperviseur gère le matériel et sépare les ressources physiques des environnements virtuels. Ces ressources sont partitionnées selon les besoins à partir de l'environnement physique et distribuées aux machines virtuelles.

Lors de l'exécution d'une machine virtuelle, si un utilisateur ou un programme lance une instruction qui sollicite des ressources supplémentaires de la part de l'environnement physique, l'hyperviseur planifie la demande auprès des ressources du système physique afin que le système d'exploitation et les applications de la machine virtuelle puissent accéder au pool partagé de ressources physiques. [8]

1.3 Avantages et inconvénients [9]

1.3.1 Avantages

- La virtualisation est moins chère.
- Maintien des coûts prévisibles.
- Réduit la charge de travail.
- Fournit une disponibilité améliorée.
- Permet un déploiement plus rapide des ressources.
- Favorise l'entrepreneuriat numérique.
- Offre des économies d'énergie.
- Moins de retards de chaleur.
- Redéploiement rapide.
- Sauvegardes plus faciles.
- Des pâturages plus verts.
- Grands tests.
- Aucun fournisseur requis.
- Bonne reprise après sinistre.
- Serveurs résolus.
- Migration simplifiée vers le cloud.

1.3.2 Inconvénients

- Coût de mise en œuvre plus élevé.
- Livré avec des limitations.
- Produit un risque pour la sécurité.
- Développe un problème de disponibilité.
- Produit un problème d'évolutivité.
- Nécessite divers maillons de chaîne pour un fonctionnement cohérent.
  Consomme du temps.

1.4 Hyperviseurs

1.4.1 Définition

Un hyperviseur est un logiciel qui permet de créer et d'exécuter des machines virtuelles. Un hyperviseur isole son système d'exploitation et ses ressources des machines virtuelles, et permet de créer et de gérer ces machines virtuelles.

Le matériel physique utilisé en tant qu'hyperviseur est appelé « hôte », tandis que toutes les machines virtuelles qui utilisent ses ressources sont appelées « invités ».

L'hyperviseur traite les ressources (telles que le processeur, la mémoire et le stockage) à la manière d'un pool qui peut être déplacé sans difficulté entre les invités existants ou vers de nouvelles machines virtuelles.

Pour exécuter des machines virtuelles, tous les hyperviseurs ont besoin de certains composants au niveau du système d'exploitation : gestionnaire de mémoire, ordonnanceur, pile d'entrées/sorties (E/S), pilotes de périphériques, gestionnaire de la sécurité, pile réseau, etc.

L'hyperviseur attribue à chaque machine virtuelle les ressources qui ont été allouées, et gère la planification de leurs ressources en fonction des ressources physiques. Cependant, l'exécution est toujours prise en charge par le matériel physique et le processeur exécute toujours les instructions des machines virtuelles. [10]

1.4.2 Types

Il existe deux types hyperviseurs à savoir les hyperviseurs de type 1 et les hyperviseurs de type 2 :

- **Un hyperviseur de type 1**, également appelé hyperviseur de système nu ou natif, s'exécute directement sur le matériel de l'hôte pour gérer les systèmes d'exploitation invités. Il prend la place du système d'exploitation de l'hôte et planifie directement les ressources des machines virtuelles sur le matériel.

Ce type d'hyperviseur est fréquemment utilisé dans les datacenters d'entreprise et dans d'autres environnements basés sur des serveurs.

KVM, Microsoft Hyper-V et VMware vSphere sont des exemples d'hyperviseurs de type 1.

- **Un hyperviseur de type 2,** également appelé hyperviseur hébergé, s'exécute sur un système d'exploitation traditionnel en tant que couche logicielle ou application.

  Il fonctionne en dissociant les systèmes d'exploitation invités du système d'exploitation hôte. Les ressources des machines virtuelles sont planifiées au niveau d'un système d'exploitation hôte, lui-même exécuté sur le matériel.

  L'installation d'un hyperviseur de type 2 convient aux utilisateurs qui souhaitent exécuter plusieurs systèmes d'exploitation sur un ordinateur personnel. VMware Workstation et Oracle VirtualBox sont des exemples d'hyperviseurs de type 2.

1.5 Techniques de virtualisations

Le schéma suivant (Figure1) montre un aperçu des principales techniques de virtualisation :

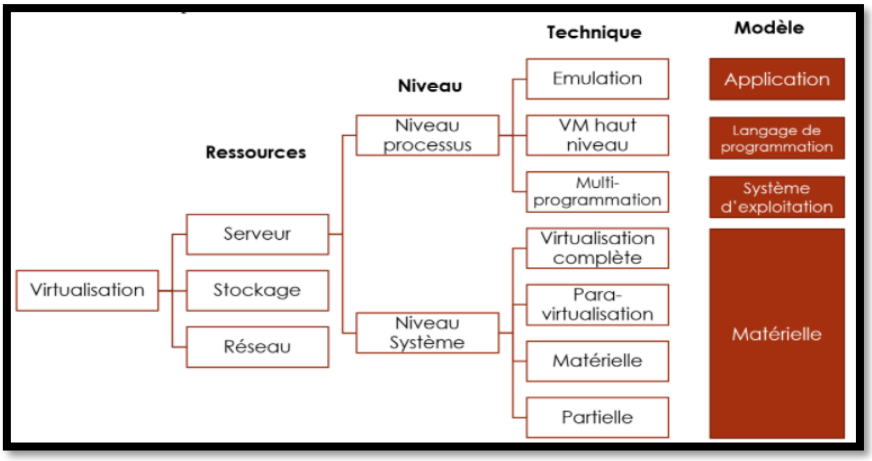

Figure 1. Techniques de virtualisation [11]

La virtualisation de serveur est divisée principalement selon deux niveaux : [11]

- Processus : qui inclus les techniques d'émulation pour les applications, de VM haut niveau et la multi-programmation (isolation).
- Système : qui inclut les virtualisations matérielles telles que : la virtualisation complète, la para-virtualisation, la virtualisation assistée par le matériel et virtualisation partielle.

1.5.1 L'émulation

C'est est une technique permettant aux applications d'être exécutées dans des environnements d'exécution qui ne supportent pas nativement toutes les fonctionnalités requises par de telles applications.

1.5.2 La virtualisation de langage de programmation

L'application de virtualisation émule les instructions utilisateurs et les appels système, appelée "runtime" (type de logiciel d'exécution). Elle est placée au niveau des interfaces ABI, au-dessus du système/matériel (voir Figure 2).

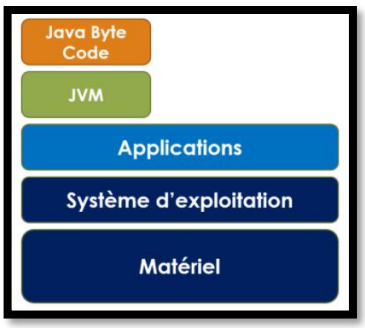

Figure 2.Virtualisation au niveau langage de programmation [11]

1.5.3 Isolation

Elle permet de créer des environnements d'exécution différents et séparés pour les applications gérées simultanément.

La virtualisation se fait au sein d'un système d'exploitation unique (voir Figure 3), où le noyau du système d'exploitation permet plusieurs instances d'espace utilisateur isolées (système de fichiers, adresses IP, des configurations logicielles, accès aux périphériques).

Contrairement à la virtualisation matérielle, il n'y a pas de gestionnaire de machine virtuelle ni d'hyperviseur.

On distingue deux types d'isolation : isolation d'application, isolation de système.

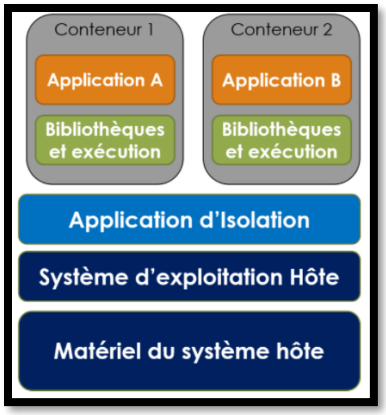

Figure 3. Virtualisation au niveau système (isolation) [11]

1.5.4 Virtualisation complète

La virtualisation complète fait référence à la possibilité d'exécuter un programme (généralement un système d'exploitation), directement sur une machine virtuelle et sans aucune modification, comme s'il était exécuté sur le matériel brut. Les gestionnaires de machines virtuelles doivent fournir une émulation complète de matériel sous-jacent.

Une implémentation réussie et efficace de la virtualisation complète est obtenue avec une combinaison de matériel et de logiciel, ne permettant pas l'exécution d'instructions potentiellement dangereuses directement sur l'hôte. C'est ce qui est accompli grâce à la virtualisation assistée par matériel.

Le VMM a une autorité d'anneau 0 tandis que le système d'exploitation invité a une autorité d'anneau 1. Les appels ISA du système invité sont convertis en appels ISA du système hôte en utilisant le processus de traduction binaire. Les instructions privilégiées sont déroutées (traps en anglais).

1.5.5 Para-virtualisation

La para-virtualisation est solution non transparente qui permet d'implémenter des VMM légers (voir Figure 4). Les techniques de paravirtualisation exposent une interface logicielle à la machine virtuelle légèrement modifiée depuis l'hôte et, par conséquent, les machines invitées doivent être modifiés.

Le but de la paravirtualisation est de fournir la capacité d'exiger l'exécution d'opérations critiques directement sur l'hôte, évitant ainsi des pertes de performances qui seraient autrement rencontrées dans l'exécution gérée. Cela permet une implémentation plus simple des gestionnaires de machines virtuelles qui doivent simplement transférer l'exécution de ces opérations, qui étaient difficiles à virtualiser, directement à l'hôte. Ceci est possible lorsque le code source du système d'exploitation est disponible, ainsi la paravirtualisation a été principalement explorée dans l'environnement open-source et académique.

Les instructions privilégiées du système invité sont envoyées à l'hyperviseur en utilisation des hyper-appels. Les hyper-appels gèrent ces instructions, accèdent au matériel et retourne le résultat. Les machines invitées ont l'autorité d'avoir un contrôle direct des ressources.

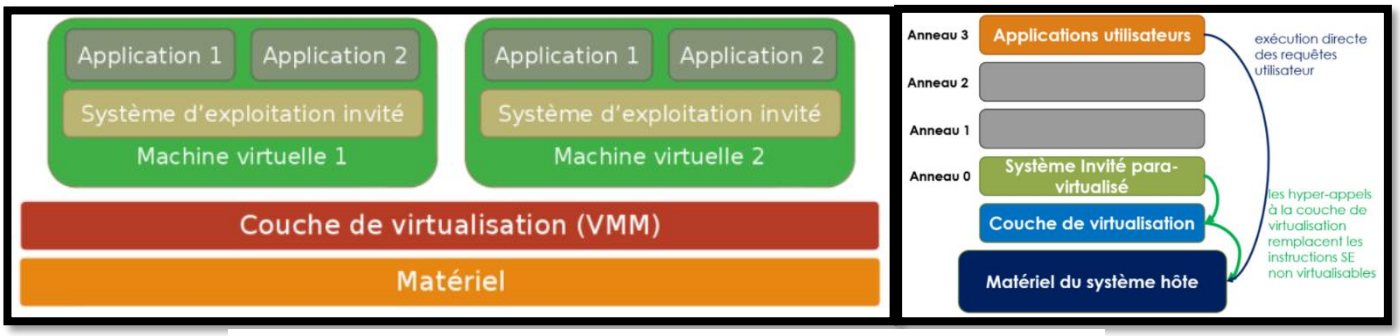

Figure 4. Para-virtualisation [11]

1.5.6 Virtualisation assisté par le matériel

Dans cette technique le matériel fournit un support architectural pour la construction d'un VMM capable d'exécuter un système d'exploitation invité dans une isolation complète (voir Figure 5). Cette technique a été initialement introduite dans IBM System / 370. En 2006, Intel et AMD ont introduit des extensions de processeurs, et une large gamme de solutions de virtualisation en a profité : KVM, VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels…

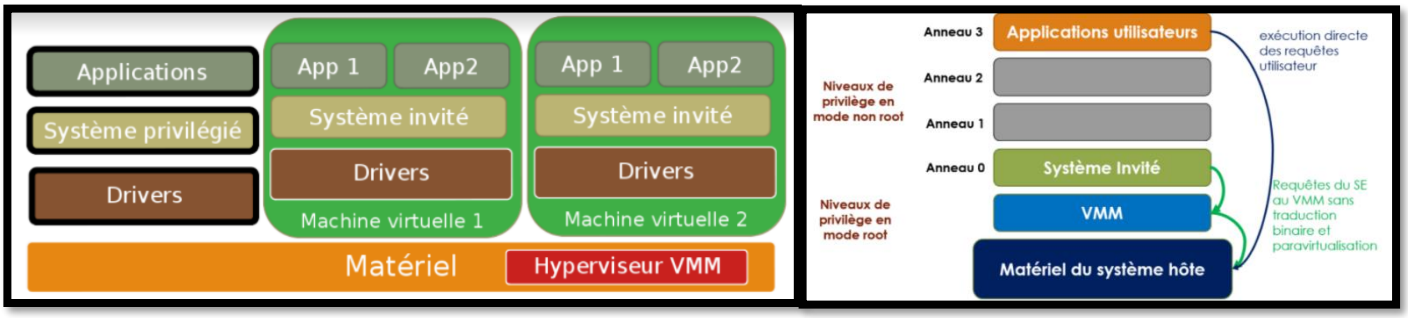

Figure 5. Virtualisation assistée par le matériel [11]

1.5.7 Virtualisation partielle

La virtualisation partielle est une émulation partielle du matériel sous-jacent. Elle ne permet pas l'exécution complète du système d'exploitation invité dans une isolation complète. Un exemple est la virtualisation de l'espace d'adressage des mémoires utilisée dans les systèmes à temps partagé : permet à plusieurs applications et utilisateurs de s'exécuter simultanément dans un espace mémoire distinct, mais ils partagent toujours les mêmes ressources matérielles (disque, processeur et réseau) qui est une caractéristique des systèmes récents. Historiquement, la virtualisation partielle a été une étape importante pour la virtualisation complète, et elle a été implémentée sur l'IBM M44 / 44X expérimental (1960).

1.6 Conclusion

La virtualisation est une technologie clé dans le cloud computing qui a contribué, de près, à son émergence. L'une des composantes principales de cette technologie est la machine virtuelle parce qu'elle contient tous les systèmes d'exploitation et les applications nécessaires au déploiement des services pour les clients.

Nous avons présenté à travers ce chapitre, le concept de base qui a été derrière l'apparition du cloud computing, à savoir la virtualisation. Les principes de cette technologie et ses techniques ont été décrites.

Le chapitre suivant sera consacré à la description de la migration dans le Cloud computing, ainsi que quelques travaux existants sur cette technique.

Chapitre 2

La migration

2.1 Introduction

La virtualisation est une technologie qui évolue rapidement et qui permet une allocation flexible des ressources dans les data centers du cloud. Les machines virtuelles (MV) sont créées en fonction de la quantité des ressources requises, puis exécuté sur des machines physiques (MP) pour héberger des applications afin de répondre aux exigences des clients. Cependant, la charge des applications change constamment dans l'environnement cloud computing, ce qui peut provoquer des violations des accords de qualités de services (QoS). Par conséquent, certaines MV sur la MP surchargés doivent être migrées, afin d'assurer le bon fonctionnement des data centers du cloud.

La Migration des instances du système d'exploitation (VMs) d'une machine physique vers une autre est une caractéristique importante de la virtualisation. La migration des VMs inclus aussi le transfert de la mémoire, le CPU, les entrées /sorties vers la machine de destination.

La migration des machines virtuelles entre les différents hôtes physiques est rapidement devenue incontournable dans les data centers. Les capacités d'hébergement des serveurs devenant de plus en plus élevées, les besoins de consolidation et de répartition de charge ont suscité un fort intérêt pour la migration des VMs. En offrant la possibilité de basculer dynamiquement des charges de travail entre les serveurs, la migration est quotidiennement utilisée pour améliorer les performances des applications, réduire les

consommations énergétiques des data centers ou pour préparer des taches de maintenance sur des serveurs en production.

2.2 Techniques de migration des VMs

Les techniques de la migration sont classifiées globalement en deux types : **Migration à chaud (Live Migration)** et **Migration à froid (Non Live Migration). [12]**

2.2.1 Migration à froid « Stop and Copy Migration »

La migration à froid (méthode d'arrêt et de copie) est une technique où la VM cesse de fonctionner sur la machine source et son exécution avec sa mémoire sont migrées vers la machine destination. Une fois toutes les pages mémoires transférées, la VM est activée dans la machine destination. La VM reste en état désactivé jusqu'à ce que toutes les pages soient transférées à la machine destination. Dans ce type de migration, le temps total de la migration est égal au temps d'arrêt.

Pour décrire les étapes de migration ***Stop and Copy***, nous prenons en exemple la migration de la VM3 de la machine 1 vers la machine 2 : (voir Figure 6)

1 - L'exécution de la VM3 est arrêtée dans la machine cible.

2 - L'état d'exécution et les pages mémoires de la VM3 sont transférés vers la machine 2.

3 - La VM3 est activée dans la machine destination.

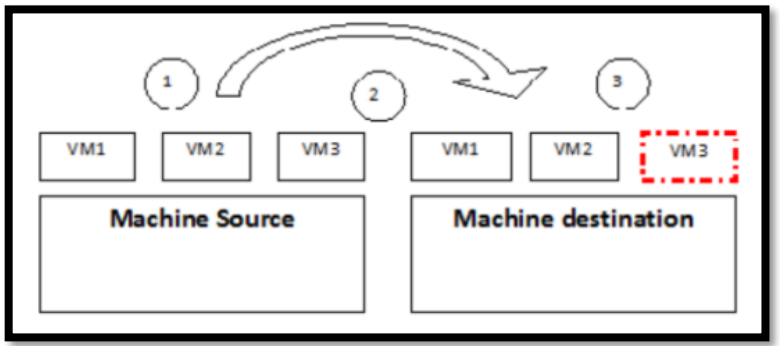

Figure 6. Processus de la migration à froid. [12]

2.2.2 Migration à chaud « Live Migration »

La migration à chaud (Live Migration) est utile pour la migration des systèmes d'exploitation complets ainsi que leurs applications entre les machines physiques éloignées des Data centers et des clusters. C'est une technologie impressionnante qui facilite l'équilibrage de charge, la gestion des pannes, la maintenance du système et la réduction de la consommation d'énergie. La migration à chaud des machines virtuelles est le processus qui permet de transférer une ou plusieurs machines virtuelles d'une machine physique ou d'un espace de stockage vers un autre sans interrompre les autres VMs.

Cette technique est utile dans plusieurs scénarios, parmi lesquels :

- Une VM dans un nœud physique défaillant peut être migrée vers un autre nœud non défaillant.
- Les VMs au mode repos sur une machine physique peuvent être migrées vers d'autres hôtes pour optimiser l'utilisation des ressources.
- Les VMs sur un nœud physique chargé peuvent être migrés vers différents nœuds pour équilibrer les charges.

Les techniques de la migration à chaud sont caractérisées par 2 types basés sur le processus de copie de la mémoire. Ces techniques sont : **"Post-copy"** et **"Pre-copy".**

2.2.2.1 Live Migration "Post-Copy"

Dans cette technique, la VM est suspendue dans la machine source et son état d'exécution (CPU, les registres et les pages mémoires nécessaires pour activer la VM dans la machine destination) est transféré vers la machine destination. La VM commence son exécution sur la machine destination même si les pages mémoires n'ont pas été copiées entièrement. L'inconvénient de cette méthode est l'apparition des défauts de pages mémoires dans la machine destination. Les défauts de page peuvent se produire lorsque les pages mémoires de la VM ne sont pas disponibles sur l'hôte destination.

Pour le principe de fonctionnement de la migration ***Post-Copy***, nous prenons en exemple la migration de la VM3 : (voir Figure 7)

1- La VM3 est suspendue dans la machine source.

2- L'état d'exécution de la VM3 est transféré vers la machine destination.

3- La VM3 est activée dans la machine destination.

4- Générer les défauts de pages mémoires.

5- Transférer les pages à défaut correspondante.

6- Les étapes 4 et 5 sont répétées jusqu'à ce que toutes les pages mémoires sont transférées vers la machine destination et la VM3

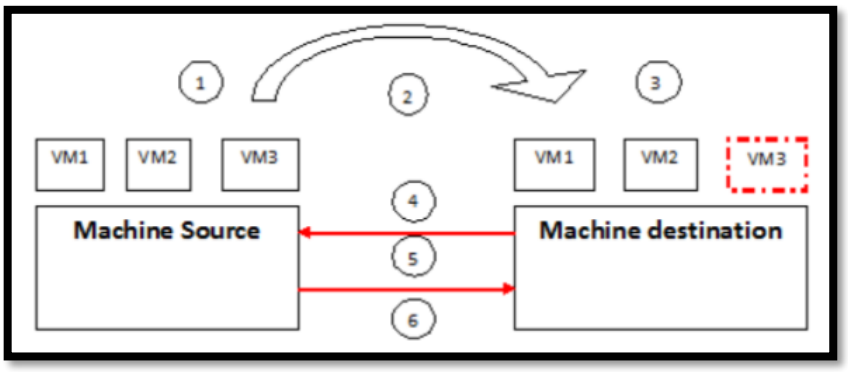

Figure 7. Processus de la migration Post-Copy. [12]

2.2.2.2 Live Migration "Pre-Copy"

C'est une technique de la migration à chaud qui est utilisée par les hyperviseurs comme XEN, KMV et VMware. Elle envoie la VM entière vers la machine destination durant la première itération puis transfert de manière itérative les pages mémoires modifiées. Ce processus se poursuit jusqu'à ce que toutes les pages mémoires suffisantes soient copiées sur l'hôte destination.

Le principe de la migration ***Pre-copy*** de la VM3 est comme suit : (voir Figure 8)

1- Envoyer toutes les pages mémoire de la VM3 de la machine source vers la machine destination.

2- Transférer les pages mémoire modifiées d'une manière itérative.

3- Arrêter la VM3 dans la machine la source.

4- Envoyer les états restants de la VM3.

5- Activer la VM3 dans la machine destination.

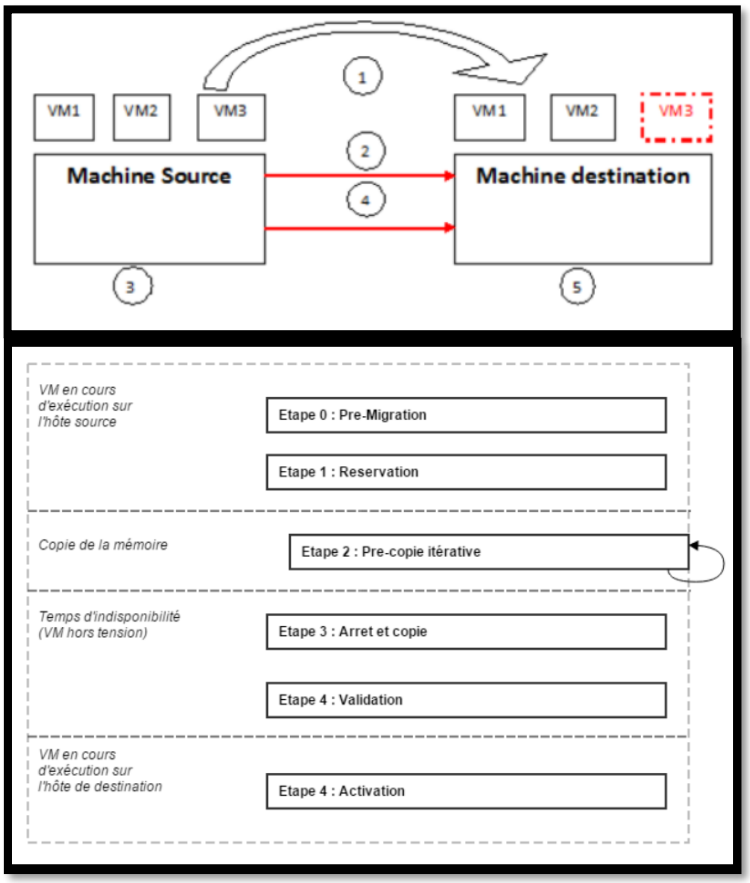

Figure 8. Processus de la migration Pre-Copy. [12]

En cas de panne de la machine destination, la technique Pre-copy est bien meilleure que la Post-copy. La récupération de la VM3 est possible par la méthode Pre-Copy car la dernière copie mémoire est conservée dans un lieu sûr sur la machine source y compris l'état du CPU.

Pour analyser les performances de la migration à chaud, les mesures suivantes sont utilisées.

- **Préparation :** Dans cette phase, les ressources appropriées sont réservées sur la machine destination et diverses opérations sont effectuées sur la machine source.

- **Temps d'arrêt :** moment de mise à l'arrêt de la machine VM sur la machine source.
- **Temps résumé :** la machine virtuelle s'exécute sur la machine physique destination avec le même état que celui de la machine source.
- **Temps total de migration :** c'est le temps total qu'il faut pour réaliser toutes ces étapes précitées.

2.3 Nature de l'entité à faire migrer

2.3.1 Migration des services ou des processus

Un processus correspond à l'exécution d'une séquence d'instructions. Il est représenté par un état qui évolue suivant l'avancement de l'exécution. L'état, ou encore l'image, du processus constitue l'ensemble des informations propres au processus et nécessaires à son exécution, comme par exemple le contenu de l'espace d'adressage du processus, son contexte d'exécution, etc. [13]

La migration d'un processus consiste à transférer son exécution d'un processeur source vers un processeur destination. Elle passe par trois phases :

- **L'extraction du processus :** Cette action comprend la suspension (ou gel) du processus, la détermination de son image et la libération de certaines ressources sur le processeur source.

- **Le transfert de l'image du processus :** (ou d'une partie) vers le processeur destination de façon à permettre la réactivation du processus.

- **La restauration de l'exécution du processus :** Cette action comprend la réallocation de l'image sur le processeur destination, le rétablissement des interactions entre le

processus et le reste du système (au niveau des communications entre processus et des accès aux entités) et la reprise de l'exécution.

2.3.2 Migration dans les bases de données

Les nouvelles applications sur Internet connaissant un grand succès et des défis comme : [13]

- La croissance exponentielle de volume de données qu'ils doivent gérer.
- La gestion des accès multiples à ces données

Ces applications nécessitent de créer de nouveaux systèmes de gestion de bases de données (SGBDs) qui sont adaptables au Cloud. C'est ce qui a donné naissance aux SGBD NoSQL (Not Only SQL). La migration des bases de données consiste à exporter le schéma de la base de données puis transférer les données vers un autre un nœud pour garantir une bonne qualité de services (Service Level Agreement) pour les utilisateurs, et un bon équilibrage de charge. Bien que la plupart des SGBDs NoSQL supporte la migration des données, ils utilisent des techniques de migration lourdes comme la migration à froid (Cold Migration or Stop and Migrate) qui consistent à mettre en attente (Downtime) toutes les transactions pendant la période de la migration engendrant une dégradation des performances du service.

Les nouvelles applications sur Internet connaissant un grand succès et des défis comme :

- La croissance exponentielle de volume de données qu'ils doivent gérer.
- La gestion des accès multiples à ces données

NoSQL (Not Only SQL) désigne une catégorie de SGBD non-relationnels proposant des alternatives au langage SQL et au modèle relationnel utilisés dans les bases de données traditionnelles. Ces nouveaux SGBD sont apparus il y a quelques années pour répondre aux besoins des sites à fort trafic comme Google, Amazon ou Facebook. C'est ce qui leurs a

permet de disposer de bases de données plus adaptées au stockage d'un volume massif de données. Ils offrent de meilleures performances en lecture/écriture que les bases de données relationnelles et possèdent une grande évolutivité (en termes de scalabilité horizontale). En contrepartie, ils ne possèdent pas toutes les propriétés ACID qui garantissent la fiabilité d'une base de données. NoSQL ne remplace donc pas le modèle relationnel mais offre des solutions intéressantes dans certaines situations où les performances priment sur la cohérence des données.

2.3.3 Migration des machines virtuelles

C'est une action permettant de déplacer une machine virtuelle d'une machine physique vers une autre. Une machine virtuelle fonctionnant sur un premier hôte physique peut être migrée vers un autre hôte physique. La migration elle-même englobe le transfert de l'état persistant de la machine virtuelle, le transfert de l'état instable de la machine virtuelle (par exemple contenu de la RAM et de l'état du processeur et la redirection du trafic réseau). Une fois que le transfert d'état est terminé, la machine virtuelle continue à fonctionner dans la nouvelle machine physique. La migration des VMs peut être effectuée soit par une migration à froid, ou par une migration à chaud. [13]

2.4 Avantages et inconvénients de la migration

2.4.1 Avantages

Les avantages de la migration des machines virtuelles comprennent [14] :

- **Tolérance aux pannes :** La tolérance aux pannes permet aux machines virtuelles de continuer leurs tâches, même si une partie quelconque de système échoue. Cette technique migre la machine virtuelle d'un serveur physique à un autre serveur en se basant sur la prédiction de la panne. La technique de migration à tolérance aux pannes permet d'améliorer la disponibilité du serveur physique et évite la dégradation des performances des applications.

- **Equilibrage de Charge :** La technique de migration d'équilibrage de charge vise à répartir la charge de travail sur les serveurs physiques pour améliorer l'évolutivité des serveurs physiques dans un environnement de cloud.

- **Réduction de la consommation d'énergie :** La consommation d'énergie des Data Centers est principalement basée sur l'utilisation des serveurs et leurs systèmes de refroidissement. Ces serveurs utilisent généralement jusqu'à 70% de leur consommation d'énergie maximale, même avec une faible utilisation des ressources. Par conséquent, il faut utiliser des techniques de migration qui conservent l'énergie consommée par les serveurs par une utilisation optimale des ressources.

- **L'adaptation automatique à l'évolution de la charge de travail (en hausse, en baisse).**

- **Garantir l'élasticité et la mise à l'échelle**

- **La possibilité de faire des tâches de maintenances et d'entretiens sans interruption de services.**

- **La capacité de faire des mises à jour et des mises à niveau de façon transparente aux clients.**

2.4.2 Inconvénients

Les inconvénients de la migration des machines virtuelles comprennent [15][16] :

- **Défauts de pages :** Lors de la migration de mémoire par pré-copie, plus les services écrivent sur la mémoire plus il y aura de défauts de pages. Ce nombre de défauts de page important impliquera par conséquent un temps d'arrêt du système plus important.
- **Conflits d'adressage :** La migration par internet permet de transférer l'environnement d'exécution des machines virtuelles d'un cloud à un autre. Ce type de migration peut engendrer des problématiques d'adressage liées au changement de réseaux

- **Problème de réseaux lors de migration.**
- **Problèmes de communication entre les VMs après la migration.**
- **Problème de routage.**

2.5 Etat de l'art sur les approches de migration des machines virtuelles dans le Cloud Computing

- **Allocation exacte et migration :** [23]

C'est une approche de Bin-Packing étendue grâce à l'inclusion de conditions valides exprimées sous forme de contraintes ou d'inégalités. L'objectif est de charger des éléments (VM dans notre cas) dans un ensemble de bacs (serveurs ou nœuds hébergeant les Machines virtuelles) caractérisées par leurs consommations d'énergie. Cette optimisation est soumise à un certain nombre de contraintes linéaires reflétant les limites de capacité des serveurs et des faits évidents tels qu'une machine virtuelle ne peut être affectée qu'à un serveur ou à un serveur ne peut héberger que des VM virtuelles en fonction de la quantité de ressources restantes.

- **Min-Cut hierarchical clustering :** [24]

Cette méthode construit les clusters en divisant récursivement les instances selon leur capacité et les classifie afin d'ordonner la liste d'hôtes sur les quelles une classe d'instance (machine virtuelle) peut être migré.

- **MFR :** [25]

C'est un algorithme de gestion de migration des VMs afin de minimiser le nombre de PMs requis pour supporter une charge de travail à un taux de violation SLA spécifié. Il utilise le modèle estimé pour faire des prédictions sur la future demande des ressources de

manière standard, et MFR mappe l'ensemble des VMs aux PMs sans considérer le nombre de migration.

- **Algorithme de paquetage de groupe :** [26]

L'algorithme de regroupement est un algorithme qui regroupe des objets de forme ou de taille similaire, et il est développé sur la base d'une méthodologie comparative basée sur la forme totale pour trouver des formes et des tailles similaires.

- **Algorithme de dimensionnement de VM :** [27]

L'approche sert à estimer le montant des sources qui devraient être attribuées à une VM dans le but de faire correspondre les rendements de VMs à sa charge de travail. Le dimensionnement de VM doit être effectué de manière à éviter la violation de SLA résultant du sous-provisionnement des ressources.

- **VectorDot :** [28]

L'algorithme dynamique VectorDot est proposé pour la gestion de la charge. Cet algorithme prend en charge la structure en couches du centre de données et gère également la multi-dimensionnalité des commutateurs réseau, la charge de ressources dans les serveurs et le stockage dans un centre de données afin d'équilibrer la charge au niveau de Datacenter en appliquant de migration des VMs.

Le tableau suivant résume tous les travaux dans la migration de VMs par rapport aux différents aspects :

Algorithme	Basé sur	Ressources considérée	Nouvel aspect	Puissance	Faiblesse	Performent que
Allocation exacte et Migration [23]	Programmation par contrainte	CPU	Fonctions objectives pour l'optimum	Réduction de nombre d'APMs et le coût de la migration	Exécution légèrement lent	Best-fit Heuristic
Min-Cut hierarchical clustering [24]	Programmation par contrainte	CPU et BP	Optimisation de MLU et réutilisation de VM	Réduction de la consommation énergétique et le réseau de trafic	Coût élevé de la migration	BFD et Random algorithm
MFR (Measure-Forecast-Ramap) [25]	Programmation stochastique entière	CPU	Longueur de l'intervalle de temps « t »	Réduction de nombre d'AMPs	Besoin d'extension de nombreuses ressources	Static algorithm
Algorithme de paquetage de groupe [26]	Emballage stochastique	CPU et BP	Variable aléatoire pour prédire le future BP	Réduction de nombre d'AMPs	Besoin d'extension de nombreuses ressources	First-fit, FFD and Harmonic algorithm

Algorithme de dimensionnement de VM [27]	Programmation stochastique entière	CPU et mémoire	Probabilité « P » de débordement de serveur	Réduction de nombre d'AMPs et O(l)-approximation	Besoin d'extension de nombreuses ressources	FFD algorithm
VectorDot [28]	Emballage stochastique	CPU, mémoire et réseau : E/S de BP	Métrique EVP, Serveur intégré et Stockage de migration	Equilibrage de charge dynamique, gestion de nœuds surchargés	Besoin de suivre les modèles prédictifs et statistiques	Best-fit, First-fit, Worst-fit and Relaxed-Bestfit heuristics

Tableau 1. Exemples d'algorithme de migration

Chapitre 3

Conception

3.1 Introduction

Nous avons vu dans le chapitre précédent quelques travaux de la littérature qui traitent les différents problèmes ou difficultés qui peuvent survenir lors la migration des machines virtuelles et nuire au bon déroulement de cette dernière. Ces travaux traitent la migration des machines virtuelles comme des problèmes d'affectation ou d'allocation, et ils se sont basés sur des heuristiques et des méta-heuristiques. De plus, dans d'autres travaux, les auteurs ont préféré établir des modèles de coût basés sur des paramètres statistiques en récoltant des données et en observant le comportement des serveurs et des machines virtuelles vis-à-vis des charges de travail.

Le problème de migration de machines virtuelles est traité comme un problème d'allocation où il est question d'attribuer des machines virtuelles à des machines physiques. Pour traiter ce problème, nous allons proposer une stratégie de migration optimisée basée sur des travaux connexes, cette stratégie sera basée principalement sur deux phases : « **Phase de sélection »** et **« Phase d'allocation ».**

Dans ce présent chapitre nous présentons l'architecture du modèle sur lequel nous avons travaillé ainsi que quelques métriques qui vont nous permettre de tester les performances de notre approche. Par la suite nous décrirons l'implémentation de notre modèle avec les différents les algorithmes.

3.2 Analyse de l'existant

3.2.1 Les différentes approches de migration de VMs

Généralement il existe deux approches de migration de $\boldsymbol{VM_s}$ qui sont les suivantes :

- Approche basée sur la qualité de service.
- Approche basée sur la consommation énergétique.

3.2.1.1 L'approche basée sur la qualité de service

La qualité de service (QoS) dénote le degré de performance, de fiabilité et de disponibilité offert par une application et par la plateforme ou l'infrastructure qui l'héberge. La qualité de service est fondamentale aux utilisateurs du cloud, qui attendent des fournisseurs de délivrer les caractéristiques annoncées et pour les fournisseurs cloud, qui ont besoin de trouver le bon compromis entre les degrés de la qualité de service et le coût opérationnel. Trouver le compromis optimal n'est pas une décision facile parce qu'elle implique la « Service Level Agreements (SLAs) » qui spécifie les cibles de la QoS et les pénalités économiques associées aux violations de la SLAs.

Les fournisseurs de service doivent remplir des contrats SLA qui déterminent les revenus et les pénalités sur la base du degré de réussite de la performance. [21]

3.2.1.2 L'approche basée sur la consommation énergétique

Cette approche est beaucoup plus utilisée pour réduire le coût énergétique très élevé dans les Datacenter. Les hôtes qui ont un très petit nombre de machines virtuelles pourront donc être éteints. Elle comprend trois phases habituellement :

- **La détection de la surcharge de l'hôte :**

La technique de planification doit avancer un seuil limite dans le but de décider quand un certain hôte/serveur est surutilisé. Cette limite peut être appelée « Seuil chaud » ou « Hot

Threshold » en anglais, et quand cette limite est dépassée, certaines des machines virtuelles de l'hôte doivent être migrées vers d'autres hôtes. [22]

- **La détection de la charge réduite de l'hôte :**

Si un certain serveur est sous-utilisé, c.-à-d. qu'il a atteint sous le « Seuil froid » ou en anglais « Cold Threshold », le scénario est juste contraire de celui de l'hôte surchargé, le but est d'identifier ce serveur et de migrer tous ses machines virtuelles vers d'autres hôtes actifs. Pour ce, l'hôte sous utilisé est libéré et peut être éteint pour économiser l'énergie. [22]

- **La sélection de la machine virtuelle et la migration :**

Les candidats appropriées (les VMs) sont sélectionnés en raison de hôtes sur ou sous-utilisés pour la migration. Les machines virtuelles sélectionnées dans les étapes précédentes sont alors placées sur une autre machine physique en fonction des critères de mappage adéquats. [22]

3.3 Les métriques utilisées

Pour évaluer la performance d'une approche et la comparaison entre les algorithmes, il est essentiel de définir des métriques de performance qui déduisent les caractéristiques pertinentes des algorithmes. Pour prouver l'efficacité de notre approche proposée, nous avons utilisé 4 métriques qui sont la consommation d'énergie par les Data Center, le nombre de migration des VMs, le pourcentage de violation des SLAs et le temps total de migration.

3.3.1 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 [29] [30] 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 ont confirmé que les serveurs inactifs consomment en moyenne 70 % de l'énergie consommée par rapport aux serveurs pleinement utilisés.

D'après [31], si l'utilisation du processeur est supérieure à 30%, la valeur inférieure est toujours 0,3. Donc, ils ont défini la consommation d'énergie $\boldsymbol{P(u)}$ par la Formule $(\mathbf{3.1})$:

$$\boldsymbol{P(u) = P_{max} * (0.7 + 0.3u)} \qquad (\mathbf{3.1})$$

Tel que :

$\boldsymbol{P_{max} = 250w}$ Pour des serveurs modernes. La constante 0.7 est la puissance moyenne consommée par un serveur inactif**.**

$\boldsymbol{u}$ **:** 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 $\boldsymbol{u(t)}$. Par conséquent, pour définir la consommation d'énergie totale par un serveur, nous utilisons l'équation $(\mathbf{3.2})$:

$$\boldsymbol{E_s = \int t\ P(u(t))dt} \qquad (\mathbf{3.2})$$

Donc la consommation d'énergie par rapport à un Data Center est calculée par la formule$(\mathbf{3.3})$ [32]

$$\boldsymbol{E_d = \frac{\sum_{j=1}^{m} P(u)_j}{m}} \qquad (\mathbf{3.3})$$

$\boldsymbol{m}$: nombre de machines physiques dans un Data Center.

3.3.2 Nombre de VMs migré

Un nombre plus élevé de VMs migrées augmente la charge du réseau et entraîne dégradation de la performance. Nous pouvons calculer le nombre de migrations effectuées pendant un certain intervalle de temps en utilisant l'équation $(\mathbf{3.4})$ suivante : [33]

$$\boldsymbol{Migrations\ (P;\ t1;\ t2) = \sum_{j=1}^{j} \int_{t1}^{t2} mig_j(P)} \qquad (\mathbf{3.4})$$

Avec P répresentant le placement courant des VMs, j étant le nombre d'hôtes, $mig_j(P)$ le nombre d'hôtes j entre l'intervalle $t1$ et $t2$ pour le placement P.

3.3.3 Violation des SLAs

Dans un environnement cloud, un accord de niveau de service (SLA) est conclu entre le fournisseur de services et l'utilisateur pour assurer le niveau de service requis.

SLA contient divers détails de niveau de service qui seront fournies à un utilisateur, telles que les capacités minimales de CPU, RAM, stockage et bande passante. En cas de violation du SLA, une partie qui est responsable de sa violation doit payer une amende à l'autre partie. L'utilisation du CPU par une VM arbitrairement varie dans le temps. L'hôte est sursouscrit (c'est-à-dire si tous les VM demandent leurs performances CPU maximales autorisées, et la demande totale de CPU dépasse la capacité du CPU). Il est défini que lorsque la requête pour le CPU dépasse la capacité disponible, une violation du SLA conclu entre le fournisseur de ressources et le client se produit. Pour nos études, la violation de SLA est calculée comme suit : [34]

$$\boldsymbol{SLA\ Violations\ (SLAV) = SLATAH * PDM} \qquad (\mathbf{3.5})$$

Où : $\boldsymbol{SLAV}$ représente $\boldsymbol{SLA\ Violations}$,

$\boldsymbol{SLATAH}$ Dénote $\boldsymbol{SLA\ violation\ Time\ per\ Active\ Host,}$

$\boldsymbol{PDM}$ Représente $\boldsymbol{Performance\ Degradation\ due\ to\ Migrations.}$

Les équations suivantes peuvent être utilisées pour calculer $SLATAH$ et PDM.

SLA violation Time per Active Host **(*SLATAH*) :** est l'observation que si un hôte exécutant des applications est utilisé à 100%, ce qui fait que la performance de l'application est limitée par la capacité de l'hôte ; Donc, Les machines virtuelles ne sont pas fournies avec les performances requises. Niveau de menace. En d'autres termes, cela signifie des violations de SLA dues à la surutilisation.

$$\boldsymbol{SLATAH = \frac{1}{J}\sum_{j=1}^{J}\frac{T_{sj}}{T_{aj}}} \qquad (\mathbf{3.6})$$

Avec J nombre d'hôtes, T_{sj} temps total d'utilisation d'un hôte j à 100% tandis que T_{aj} est la durée de vie (le temps durant lequel un hôte est actif) d'un hôte j. Quand l'utilisation de l'hôte atteint 100 %, les performances des applications sont délimitées par l'hôte.

Performance Degradation due to Migrations **(*PDM*) :** la migration à chaud est le processus de déplacement des machines virtuelles d'un hôte à un autre (sans suspension), il a un effet négatif sur les performances des applications utilisateur. Dumitrescu et Foster montrent que cet impact dépend de comportement de l'application, et la dégradation des performances peut être estimée à 10 % de l'utilisation du processeur.

En d'autres termes, cela signifie que les violations du SLA sont dues à la migration.

$$PDM = \frac{1}{V}\sum_{v=1}^{V}\frac{Cd_v}{Cr_v} \qquad (3.7)$$

V Etant le nombre de VMs, Cd_v c'est l'utilisation du CPU par la VM_v lors de toutes les migrations. Cette valeur estimée à 10%. Cr_v Est le total de CPU requis par la VM_v.

3.4 Description de la solution proposée

3.4.1 Architecture

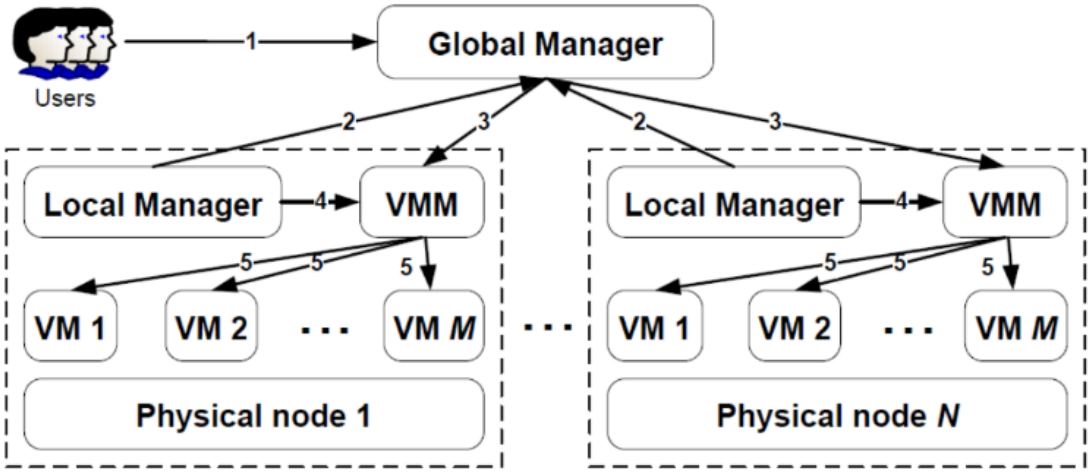

Figure 9. Architecture du système [34]

L'architecture de ce modèle (voir Figure 9) comporte comme composants principaux Global Manager et Local Manager. Les requêtes des utilisateurs sont soumises pour la fourniture en $\boldsymbol{VM_s}$ hétérogènes. Au sein d'un nœud physique réside le local manager qui sert de branche au virtuel machine monitor ($\boldsymbol{VMM}$). Les local Managers sont chargés de veiller à la supervision de l'utilisation des $\boldsymbol{CPU}$ par les nœuds qui les hébergent. Ils gèrent aussi le redimensionnement des $\boldsymbol{VM_s}$ en fonction des ressources dont elles ont besoin ou utilisées. Ils décident aussi de la VM à migrer et du moment qu'il faille faire la migration. Tant dis que les global Managers sont à l'intérieur d'un nœud global ou nœud maître pour collecter les informations stockées au niveau des Local Managers afin d'avoir une vue générale de l'utilisation des ressources. C'est le global Manager qui ordonne l'optimisation pour le placement de $\boldsymbol{VM}$. Il n'en demeure pas moins que les $\boldsymbol{VMM_s}$ s'occupent du redimensionnement, migration de $\boldsymbol{VM_s}$ et le changement de niveau d'énergie des nœuds. Les différents composants sont :

1. Data center : C'est un site physique constitué de machines physiques. Il est influencé par sa bande passante, le stockage, processeurs, capacité de la mémoire.

2. Broker : C'est une entité jouant le rôle intermédiaire de gérer les VMs dans plusieurs Data centers ainsi que le routage du trafic vers les data centers appropriés. Il se charge également de sélectionner le data center optimal pour les requêtes des clients dont il prend en charge avec les data centers.

3. Physical Node : C'est la machine qui est physiquement dans l'enceinte des data centers et qui de facto doit contenir les $\boldsymbol{VM_s}$. L'hôte physique se caractérise par la vitesse du CPU et le nombre de cœurs, nombre de requêtes envoyés, le nombre de $\boldsymbol{VM_s}$, capacité de stockage.

4. VMs : Une $\boldsymbol{VM}$ est une instance ou environnement d'exécution sur une même machine physique pour permettre aux utilisateurs d'utiliser ces $\boldsymbol{VM_s}$ comme des vraies machines physiques.

5. VM Manager : qui garantit l'utilisation des ressources et la disponibilité des machines virtuelles. C'est grâce au $\boldsymbol{VMM}$ que se fait l'affectation de $\boldsymbol{VM}$ d'une machine physique à une autre.

6. Local Manager : Ils sont présents dans tous les nœuds comme un module du $\boldsymbol{VMM}$. Ils aident à redimensionner les $\boldsymbol{VM_s}$ par rapport au besoin, ils définissent la politique de migration de VM. Le suivi du processeur d'un nœud est d'ordre de VM Managers.

7. Global Manager : sert de satellite pour un nœud maitre, ensuite il a besoin du gestionnaire local pour recueillir des informations pour avoir le contrôle sur la gestion des ressources.

3.4.2 Stratégie de migration des machines virtuelles proposée

Pour mener à bien nos travaux sur la migration des machines virtuelles en temps réel au niveau du Cloud Computing, nous nous sommes concentrés sur le problème de consommation d'énergie dans le cloud, et plus précisément pour les Datacenter. Une restriction qui a été évoquée fréquemment ces dix dernières années pour le "Green Computing" ou l'informatique verte qui luttent pour protéger l'environnement. Pour atteindre cet objectif, nous allons nous baser sur la méthode du seuil (threshold), qui entraîne la migration des machines virtuelles et une diminution de la charge de chaque PM qui les supporte.

3.4.2.1 L'approche « threshold »

D'après les auteurs [35], la méthode de threshold est distinguée des deux phases ; la phase de sélection et la phase d'allocation. Elle est basée sur un seuil dynamique ou statique

sous l’influence de la consommation d’énergie et ce seuil est obtenu en utilisant une formule mathématique.

3.4.2.2 L’approche proposée

Notre modèle est défini comme un centre de données composé d'un ensemble de PM. Chaque hôte physique peut contenir un nombre de VM hétérogènes. Ces PM et VM ont diverses ressources physiques telles que : l'utilisation du processeur, la capacité de la RAM, le MIPS, les plafonds d'utilisation des ressources, la bande passante, etc.

Le but de notre approche est de proposer une stratégie basée sur la migration des machines virtuelles. Cette proposition nous permet de sélectionner un nombre minimal de machines virtuelles afin de réduire principalement l’énergie consommée par les Data Centers et déminuer les violations des contrats des SLAs.

Afin d’atteindre cet objectif, et comme indiqué auparavant, nous nous sommes basés sur l’approche « threshold » de [35], cette dernière est composée de deux phases principales : sélection et allocation.

3.4.2.2.1 Diagramme d’activité

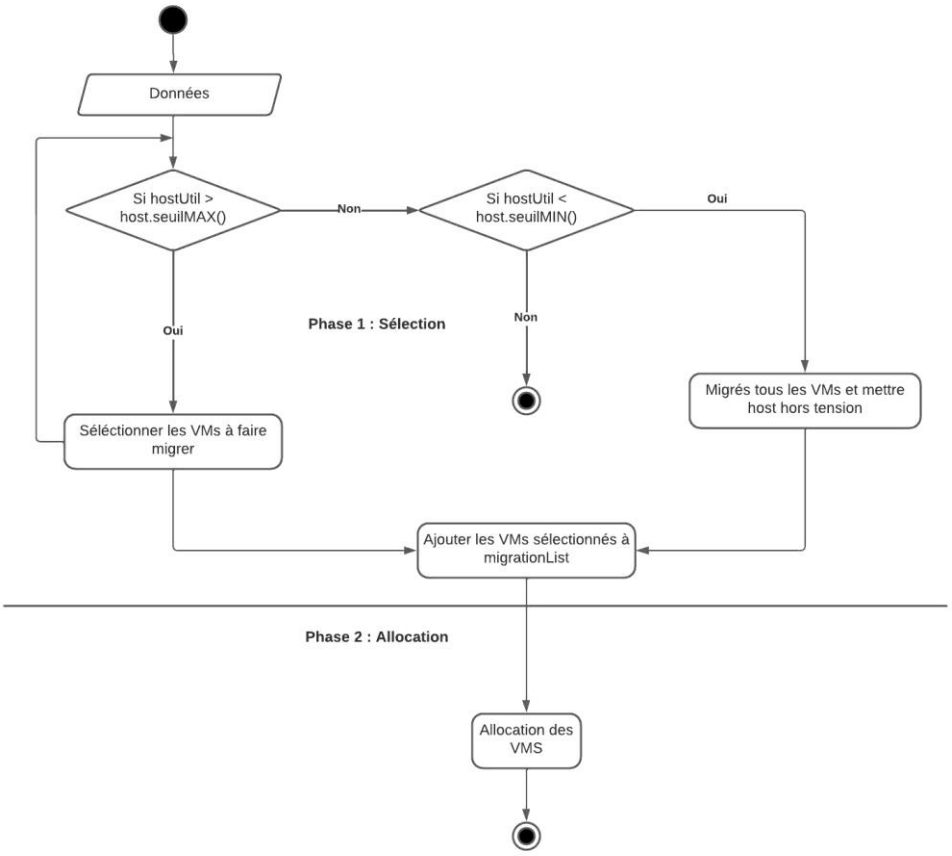

Figure 10. Diagramme d'activité de l'approche proposé

3.4.2.2.2 La phase de sélection

Dans cette phase, nous sélectionnons une liste des VMs à faire migrer vers une autre PM. Dans l'objectif de réduire la consommation énergétique et cela en appliquant notre algorithme proposé « Seuil Dynamique Hybride ».

Dans l'algorithme « SDHybride », pour sélectionner la liste des VMs à faire migrer vers d'autres machines, on procède comme suit :

• Prendre en considération les types de ressources physiques : Utilisation du CPU, la capacité de la RAM, la bande passante consommée.

• Utiliser deux seuils (inférieur et supérieur) pour les ressources physiques (Utilisation du CPU, la capacité de la RAM, la taille de la bande passante).

• Mettre en état "off" les PMs non utilisées (qui ne possèdent aucune VM).

• Les seuils inférieur et supérieur de cette approche sont dynamiques.

3.4.2.2.2.1 Seuil supérieur

Le seuil supérieur $\boldsymbol{S_{max}}$ est calculé par la formule ($\mathbf{3.12}$) [31]:

$$\boldsymbol{U_{vm} = \sum_{j=1}^{n} u_j} \qquad (\mathbf{3.8})$$

$$\boldsymbol{Bw = \sum_{j=1}^{n} Bw_j} \qquad (\mathbf{3.9})$$

$$\boldsymbol{Ram = \sum_{j=1}^{n} R_j} \qquad (\mathbf{3.10})$$

$$\boldsymbol{Temp1 = U_{vm} + \left(\frac{Bw}{Bw(PM)}\right) + \left(\frac{Ram}{Ram(PM)}\right)} \qquad (\mathbf{3.11})$$

$$\boldsymbol{S_{max} = 1 - \left(\left((0.95 * Temp1) + U_{vm}\right) - \left((0.9 * Temp1) + U_{vm}\right)\right)} \qquad (\mathbf{3.12})$$

Avec :

- $\boldsymbol{U_{vm}}$: l'utilisation de CPU par l'ensemble des machines virtuelles.
- $\boldsymbol{u_j}$: l'utilisation de CPU par la machine virtuelle $\boldsymbol{j}$.
- $\boldsymbol{Bw}$: la taille de la bande passante consommée par l'ensemble des machines virtuelles.
- $\boldsymbol{Bw_j}$: la taille de la bande passante consommée par la machine virtuelle $\boldsymbol{j}$.
- $\boldsymbol{Ram}$: la taille de la RAM allouée à l'ensemble des machines virtuelles.

- $\boldsymbol{R_j}$: la taille de la RAM allouée à la machine virtuelle $\boldsymbol{j}$.

- $\boldsymbol{Temp1}$: l'utilisation des ressources physiques.

- $\boldsymbol{n}$: le nombre des machines virtuelles.

3.4.2.2.2.2 Le seuil inférieur

Le seuil supérieur $\boldsymbol{S_{min}}$ est calculé par la formule $(\mathbf{3.16})$ [31]

$$\boldsymbol{Temp2} = \frac{(\boldsymbol{U_{vm}} + \boldsymbol{Bw} + \boldsymbol{Ram})}{(\boldsymbol{Bw(PM)} + \boldsymbol{Ram(PM)})} \qquad (\mathbf{3.13})$$

$$\boldsymbol{Sqrt} = \sqrt{(\boldsymbol{U_{vm}} - \boldsymbol{Temp2}) * (\boldsymbol{U_{vm}} - \boldsymbol{Temp2})} \qquad (\mathbf{3.14})$$

$$\boldsymbol{U_{pm}} = \frac{(\boldsymbol{CPU(PM)} + \boldsymbol{Bw(PM)} + \boldsymbol{Ram(PM)})}{\mathbf{3}} \qquad (\mathbf{3.15})$$

$$\begin{cases} \boldsymbol{S_{min}} = \boldsymbol{Sqrt} - (\mathbf{0.3} * \boldsymbol{Temp2}) & \boldsymbol{si\ U_{pm}} < \mathbf{0.3} \\ \boldsymbol{S_{min}} = \mathbf{0.3} & \boldsymbol{si\ U_{pm}} < \mathbf{0.3} \end{cases} \qquad (\mathbf{3.16})$$

Avec :

- $\boldsymbol{Temp2}$: l'utilisation des ressources physiques.

- $\boldsymbol{Sqrt}$: la racine carrée de la différence d'utilisation.

- $\boldsymbol{U_{pm}}$: la moyenne de l'utilisation des ressources physiques par l'hôte.

- Les autres notations sont les mêmes que celles qui sont citées ci-dessus

3.4.2.2.2.3 Algorithme « SDHybride »

L'algorithme 1 de la phase de sélection de notre approche vérifie l'utilisation des ressources physiques (CPU, RAM et Bande Passante) de l'hôte par rapport au seuil supérieur (Ligne 5). Si l'utilisation de la PM est supérieure au seuil supérieur, l'algorithme migre quelques VMs possédant la plus grande utilisation des ressources physiques vers un autre hôte et cela afin de réduire l'utilisation en dessous du seuil supérieur.

De la ligne 18 jusqu'à la ligne 21, l'algorithme vérifie si l'utilisation de ressources de la PM est inférieure au seuil inférieur. Si c'est le cas, toutes les VMs de cette PM sont migrées vers d'autres hôtes. Cette PM est mise hors tension Dans le but de minimiser l'énergie consommée dans le Data Center.

Algorithme 1 : Algorithme de sélection « SDHybride ».

Entrées : *hostList, vmList*

Sorties : *migrationList*

```
1.  VmToMigrate ← Null
2.  Pour chaque h dans hostList faire
3.      hUtil ← h.util();
4.      bestFitUtil ← MAX;
5.      Tant que (hUtil > h.seuilMAX()) faire
6.          Pour chaque Vm dans vmList
7.            Si Vm.Util() > (hUtil – h.seuilMAX())
8.              t ← Vm.Util() - (hUtil – h.seuilMAX());
9.                 Si (t < bestFitUtil)
10.                    bestFitUtil ← t
11.                    VmToMigrate ← Vm
12.                Sinon Si (bestFitUtil = MAX)
13.                    VmToMigrate ← Vm
14.                    Break ;
15.            hUtil ← hUtil – Vm.Util() ;
16.            migrationList.add(Vm) ;
17.            vmList.remove(Vm) ;
18.    Si hUtil < h.seuilMIN()
19.            migrationList.add(vmList) ;
20.            vmList.remove(h.getVmList()) ;
21.            h.offState() ;
22.    retourner migrationList;
```

3.4.2.2.3 La phase d'allocation

Cette phase a pour rôle de placer les VMs récupérées lors de la phase de sélection dans les hôtes moins utilisés et cela en appliquant l'algorithme « Energy-Aware Best Fit Decreasing Hybride ».

3.4.2.2.3.1 Le principe de l'algorithme Best Fit Decreasing (BFD)

L'algorithme BFD, trie les VMs dans l'ordre décroissant, puis affecte chaque VM tour à tour au nœud le plus approprié pour lui. [27]

3.4.2.2.3.2 Algorithme Energy-Aware Best Fit Decrreasing Hybride (EBFDHybride)

Afin de permettre le placement de VM, dans une PM, il est nécessaire que l'algorithme vérifie que la machine soit prête à recevoir la VM, autrement dit la machine doit avoir assez de ressources. Ensuite, il vérifie que les utilisations du CPU sont en dessous des valeurs fixées. Enfin, il vérifie que la consommation d'énergie de la machine physique par rapport aux autres PMs est la plus petite. Cet algorithme permet de placer les VMs sélectionnées par l'algorithme de sélection dans les meilleurs hôtes disponibles.

Algorithme 2 : Algorithme d'allocation « EBFDHybride ».

Entrées : *hostList, vmList, CPUThrl, CPUThrm, CPUThrh, upperRAMUtilThr*

Sorties : *allocatedHost*

```
1. vmList.sortDecreasingByCpuUtilization() ;
2. Pour chaque vm dans vmList faire
3.     minPow ← MAX
4.     allocatedHost ← Null;
5.     Pour chaque host dans hostList faire
6.         Si (isHostHeavilyLoaded(host))
7.             HeavilyLoadedHosts.add(host) ;
8.         FinSi
9.         vmMig ← getVmFromHeavilyLoadedHosts(HeavilyLoadedHosts) ;
10.        HeavilyLoadedHosts.clear() ;
11.        vmMig.clear() ;
12.    Pour chaque host dans hostList faire
13.        Si (host est prête à accueillir vm) Alors
14.            ratio ← GetCPUresUtilRatioApresAlloc(host, vm) ;
15.            Si ((ratio < CPUThrl) || (ratio < CPUThrm)) et
       (RamResUtilRatioApresAlloc < upperRAMUtilThr) Alors
16.                energyConsom ← getPowerApresVm(host, vm) ;
17.                Si (energyConsom < minPow) Alors
18.                    allocatedHost ← host
19.                    minPow ← energyConsom
20.                FinSi
21.        FinPour
22.    FinPour
23. retourner allocatedHost;
```

3.5 Conclusion

A travers ce chapitre nous avons présenté la description de l'approche proposée avec les deux phases que comporte cette technique ainsi que le mode de calcul de seuil. La première phase est la phase de sélection qui consiste en général à indiquer quand est-ce que les VMs doivent être migrées ? Quant à la deuxième phase, l'allocation, elle permet de choisir dans quel hôte nous pourrions migrer les VMs. Cette migration est globalement déclenchée par l'augmentation de consommation d'énergie, notamment l'utilisation de ressources (CPU, RAM, BP, …) au niveau d'un hôte physique. Enfin, nous avons décrit l'architecture globale de l'approche threshold avec ses différents éléments de base.

Dans le chapitre suivant, nous présenterons l'implémentation de notre approche dans le simulateur CloudSim ainsi que les résultats de notre travail.

Chapitre 4

Implémentation et Résultats

4.1 Introduction

Notre proposition a pour objectif principal de réduire la consommation d'énergie des Data Center mais aussi de minimiser le nombre de migration des VMs ainsi que les violations des SLAs. Afin de démontrer et confirmer la fiabilité de notre approche, nous avons utilisé CloudSim pour simuler, tester et comparer les résultats obtenus avec les autres approches déjà approuvé dans la littérature.

Dans ce chapitre qui conclue notre travail, nous allons présenter le langage de programmation Java, l'environnement de développement Eclipse, ces deux outils qui ont été choisis pour bien mener nos expériences. Ensuite nous discutons le simulateur CloudSim, son architecture et ses classes fondamentales. Après nous citerons les différentes étapes et les configurations nécessaires avant le lancement de la simulation de notre approche. Enfin nous interprétons les résultats obtenus.

4.2 Environnement de développement

Nous avons effectué l'ensemble des tests sur une machine avec les caractéristiques suivantes :

- Une machine avec un processeur Intel(R) Core (TM) i5-7200U CPU, une vitesse de 2.50GHz et une capacité mémoire de 8Go. Le simulateur CloudSim est sous Windows 10 de 64bits.

- Un IDE Eclipse version Luna.
- Un simulateur CloudSim version 3.0.3 développés à base de Java.
- Le langage de programmation Java

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 a été créé par l'équipe de James Golscing et Patrick Naughton de Sun Microsystems avec le soutien de Bill Joy (co-fondateur de Sun Microsystems en 1982) et a été officiellement présenté à SunWorld le 23 mai 1995. [36]

La spécificité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation, tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. A cet effet, diverses plateformes et frameworks associés sont conçus pour permettre, sinon garantir, cette portabilité des applications développées en Java. [36]

Les applications Java peuvent fonctionner sur tous les systèmes d'exploitation grâce à la plateforme JRE (Java Runtime Environment), cet avantage est dû à la JVM (Java Virtual Machine), c'est le programme qui interprète le code Java et le convertit en code natif. Mais JRE est surtout constitué d'une bibliothèque standard qui fait l'objet d'un squelette de tous les programmes Java. Cette faculté de portabilité qui a fit la réussite de Java dans l'architecture client-serveur en facilitant la migration entre serveurs, chose compliqué pour les grands systèmes.

Java est devenue actuellement une direction incontournable dans le monde de la programmation. Parmi les différentes caractéristiques qui sont attribuées à son succès, nous avons : [36]

- L'indépendance de la plateforme : c'est-à-dire le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuté des programmes Java sur tous les environnements qui possèdent une JVM.

- La portabilité : il permet 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.
- Java assure la gestion dynamique de la mémoire.
- Il est multitâche : qui veut dire qu'il permet l'utilisation de Threads7 qui sont des unités d'exécution isolées.

4.2.2 Environnement de développement Eclipse

Eclipse est une communauté open-source dont les projets visent à fournir une plateforme de développement ouverte, comprenant des espaces de travail modulaires, des outils et environnements d'exécution, pour construire, déployer et gérer des applications sur tout leur cycle de vie.

Eclipse est surtout reconnu pour son IDE Java mais son champ d'action est devenu beaucoup plus large. Les projets Eclipse offrent des outils et environnements qui couvrent tout le cycle de développement : outils de modélisation, développement, déploiement, reporting, manipulation de données, tests, etc. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

Il est téléchargeable directement sur son site officiel http://www.eclipse.org/downloads/ et est disponible sous Windows, Linux et Mac OS. Il est puissant et compatible avec toutes les nouvelles technologies Java (Java EE, Bases de données UML, XML, etc.).

4.2.3 Le simulateur CloudSim

4.2.3.1 Définition

CloudSim est une boîte à outils (bibliothèque) pour la simulation de scénarios informatiques en nuage. Il fournit des classes de base pour décrire les centres de données, les machines virtuelles, les applications, les utilisateurs, les ressources informatiques et les politiques pour la gestion de diverses parties du système.

Ces composants peuvent être mis en place pour que les utilisateurs évaluent de nouvelles stratégies dans l'utilisation de Clouds (politiques, algorithmes de planification, de cartographie et d'équilibrage de charge, etc.). Il peut également être utilisé pour évaluer l'efficacité des stratégies à partir de différentes perspectives, du coût / profit pour accélérer le temps d'exécution de l'application. Il soutient également l'évaluation des politiques d'informatique verte.

Il y a de nombreux avantages de l'utilisation CloudSim :

- L'efficacité de temps : il prend moins de temps et d'efforts pour mettre en oeuvre des applications de cloud computing.
- La flexibilité : les développeurs peuvent facilement modéliser et tester les performances de leurs applications et ses services dans des environnements hétérogènes.

4.2.3.2 L’architecture CloudSim

La structure logicielle de CloudSim et ses composants est représentée par une architecture en couche (voir Figure 11)

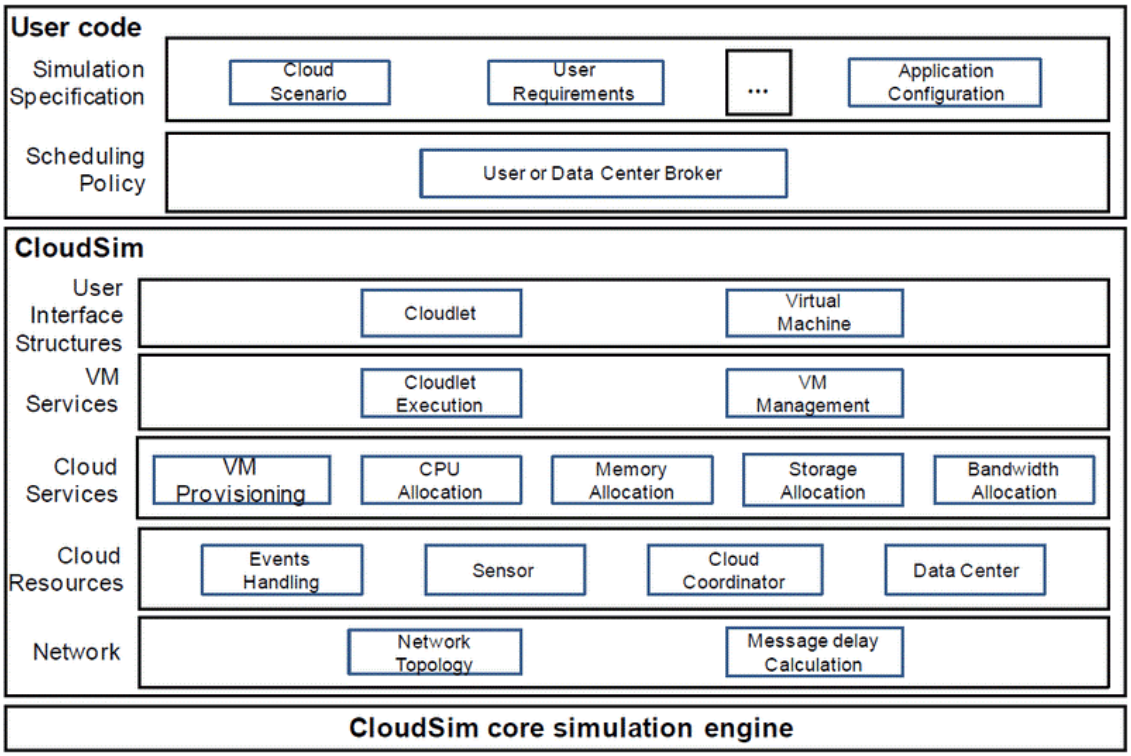

Figure 11. Architecture de CloudSim [37]

Au niveau le plus bas se trouve le moteur de simulation des événements discrets SimJava, qui implémente les fonctionnalités de base requises, tels que les files d'attente, les traitements des événements, la création des entités du système Cloud (services, hôtes, Datacenter, Broker, VM…), la communication entre les composants et la gestion d'horloge de simulation.

CloudSim supporte la modélisation et la simulation de l'environnement de Datacenter basé sur le Cloud, tel que des interfaces de gestion dédiées aux VMs, la mémoire, le stockage et la bande passante. Elle gère l'instanciation et l'exécution des entités de base (VM, hôtes, Datacenters, applications) au cours de la période de simulation.

Dans la couche plus haute de la pile de simulation, on trouve le code de l'utilisateur qui expose la configuration des fonctionnalités liées aux hôtes (nombre de machines, leurs

spécifications), les politiques d'ordonnancement de Broker, applications (nombre de tâches et leurs besoins), VM, nombre d'utilisateurs. Elle contient les informations de base nécessaires au bon fonctionnement des hôtes et des applications comme le nombre de machines, leurs spécifications, le choix des stratégies d'ordonnancement du broker, etc.

4.2.3.3 Les classes de CloudSim

Nous distinguons principalement deux catégories de classes pour le simulateur CloudSim : Les Classes qui modélisent les entités comme le data center, le Broker, etc. De l'autre côté nous avons les classes qui modélisent les politiques d'allocation. Nous présentons les classes de base de CloudSim et les classes du package Power particulièrement avec lesquelles nous avons travaillé. Nous avons intégré d'autres classes qui s'inscrivent dans le cadre de notre solution pour la simulation. (Voir Figure 12)

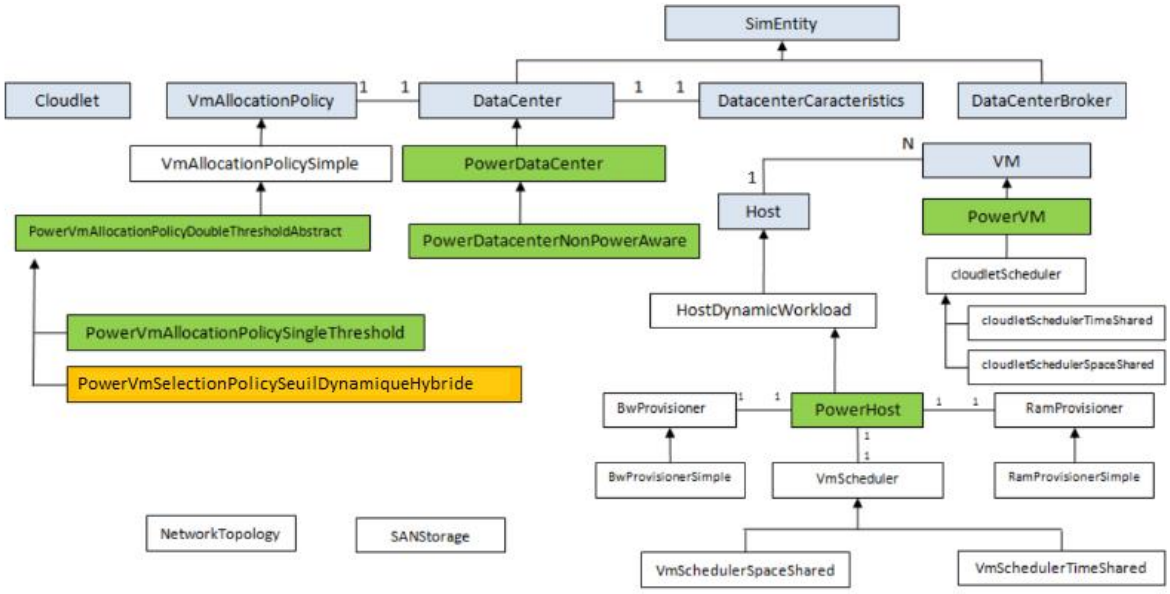

Figure 12. Diagramme de Classe de la conception de CloudSim

.

• **Cloudlet :** C’est une classe qui représente une tâche. Elle modélise les services d’application du Cloud (comme la livraison, réseaux sociaux et les sites d’affaires). Cette classe peut aussi être étendue pour supporter la modélisation des performances et d’autres paramètres de composition pour les applications telles que les transactions dans les applications orientées bases de données (Oracle, SQL). [37]

• **Datacenter :** Cette classe modélise l’infrastructure du noyau (matériel, logiciel) offert par des fournisseurs de service dans un environnement de Cloud Computing. Il encapsule un ensemble de machines de calcul qui peuvent être homogènes ou hétérogènes en ce qui concerne leurs configurations de ressources (mémoire, noyau, capacité et stockage). En outre, chaque composant du Datacenter instancie un composant généralisé d’approvisionnement en ressources qui implémente un ensemble de politiques d’allocation de bande passante, de mémoire et des dispositifs de stockage. [37]

• **datacenterBroker :** Cette classe modéliser le courtier, 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 Broker agissant au nom des utilisateurs identifie les prestataires de service appropriés du cloud par le service d’information du cloud CIS (cloud information services) en négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs. [37]

• **DatacenterCaracteristic :** C’est la classe qui contient les informations sur la configuration des ressources des Datacenter. Parmi eux le système d’exploitation, le VMM, la liste des hôtes, le coût par seconde d’utilisation des ressources, etc. [37]

• **Host :** Cette classe représente un serveur informatique physique dans un Cloud. Host exécute des actions liées à la gestion des machines virtuelles et a une politique définie pour l'approvisionnement mémoire et bande passante, ainsi que d'une politique de répartition des PE (Processeur Element) à des machines virtuelles. Un hôte est associé à un Datacenter. Il peut héberger un ou plusieurs VMs. [37]

• **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 reçus ainsi les évènements. Toutes les entités doivent étendre cette classe et redéfinir ses trois méthodes : StartEntity(), processeEvent() et shutdownEntity(). Ces méthodes définissent les actions pour l'initialisation, le traitement des événements et la destruction de l'entité. [37]

• **VM :** Cette classe modéliser une instance de machine virtuelle, qui est géré et hébergé pendant son cycle de vie par le composant Cloud Host. Chaque composant de VM a accès à un composant qui stocke les caractéristiques liées à elle telles que : l'accès mémoire, le processeur, la capacité de stockage et les politiques de provisionnement interne de la machine virtuelle. Dans le but de réduire l'énergie consommée par les Datacenters, la classe VMPower héritant le classe VM enregistre l'historique de l'utilisation de CPU. Cet historique est utilisé dans les politiques de sélection et d'allocations des VMs. [37]

• **VMAllocationPolicy :** C'est une classe abstraire qui représente la politique de provisionnement des hôtes aux machines virtuelles dans un Datacenter. Pour notre approche nous avons travaillé avec la méthode de sélection. L'algorithme proposé est implémenté dans la classe PowerVmSelectionPolicyDS.java qui hérite de la classe PowerVmSelectionPolicy.

4.3 Expérimentations et résultats

4.3.1 Configuration

- **Configuration des Data Centers (DC) :** Pour configurer un data center sous CloudSim, il faut remplir les informations suivantes : le nom du Data Center, son architecture, le gestionnaire des VMs(VMM), les différents coûts(mémoire, traitement, stockage, bande passante), l'intervalle d'ordonnancement ainsi que le système d'exploitation utilisé par le Data Center.

- **Configuration des Machines Physiques (PM) :** La création d'une PM nécessite la configuration suivante : le nombre d'hôtes, nombre de processeurs, la capacité de la

Ram, le nombre des MIPS, la bande passante et la capacité de stockage. Pour notre part les hôtes configurés sont hétérogènes avec différentes valeurs de MIPS et de la RAM.

- **Configuration des Machines Virtuelles (VMs) :** Il faudra renseigner les champs suivants pour pouvoir configurer les VMs : le nombre de VMs, le nombre de processeurs, la RAM pour chaque VM, la bande passante et le stockage, le nombre de MIPS. En résumé pour configurer les cloudlets, il faudra préciser la taille du cloudlet, le nombre de processeurs, le nombre de cloudlet. Comme il a été dit précedemment pour notre simulation, les machines virtuelles sont hétérogènes et les cloudlets égalent les VMs en nombre.

- **Lancement de la simulation :** pour lancer la simulation, après la configuration, une classe contient une méthode main avec les paramètres de simulation nécessaires à la simulation. Elle contient le type de workload, la politique de sélection de VM choisie, la politique d'allocation.

4.3.2 Résultats Expérimentaux

Pour tester notre approche, et comme préciser auparavant, nous allons utiliser trois métriques : la consommation d'énergie, le nombre de VMs migrées et les violations des SLAs. Afin de mesurer l'efficacité de notre travail nous avons jugé utile de comparer les résultats obtenus à ceux obtenus en exécutant les approches proposées par « Rabé Labo & Moussa Salifou Chapiou » l'année dernière.

4.3.2.1 Première expérience : Un nombre élevé de VMs et PMs

Dans cette première expérience, nous avons configuré le Data Center avec 100 hôtes physique et 200 machines virtuelles de différents types. Le Broker utilise un pas de 50 pour varier le nombre de VMs de 50 à 200.

4.3.2.1.1 Consommation d'énergie

Après avoir terminé la simulation et récupérer les résultats (voir Figure 13), la consommation d'énergie s'avère moindre avec notre approche SeuilDynamiqueHybride par rapport aux approches MiMcHybrid et MmtMiMcHybrid. D'après l'analyse des résultats obtenus et comme peut l'indiquer le graphe, nous constatons que les politiques que nous avons implémentés, ont un ratio de consommation d'énergie relativement faible par rapport aux autres, ce qui nous ramène à dire que nos approches convergent vers l'objectif de l'informatique verte (Green Computing) c'est-à-dire produisent moins de chaleur et économisent de l'énergie. Nous y avons obtenu ces résultats avec les charges de travail random (random workload), les tests dans le package power.random de cloudsim.

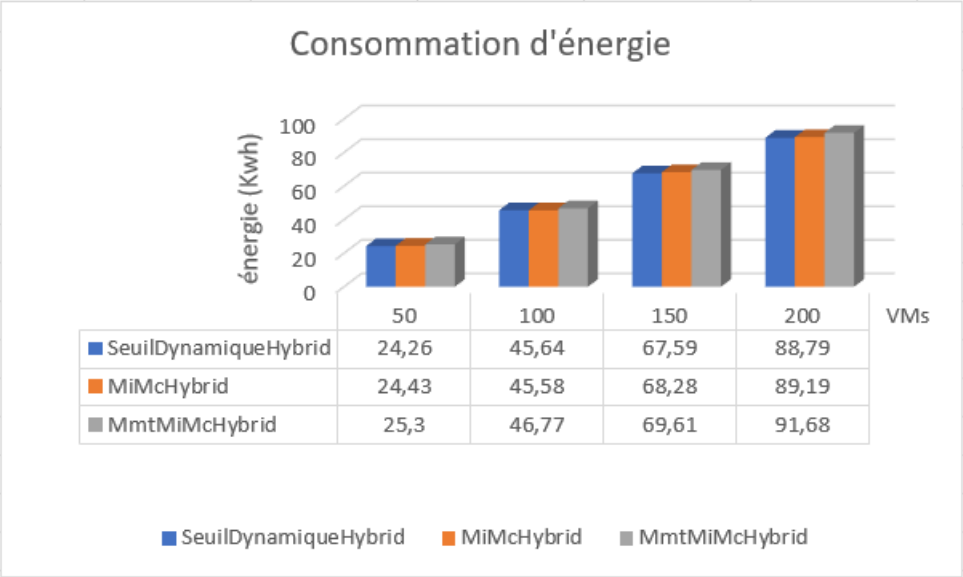

Figure 13. Consommation d'énergie

4.3.2.1.2 Nombre de migration de VMs

En ce qui concerne cette métrique, les résultats de simulation (voir Figure 14) indique que notre approche SeuilDynamiqueHybrid réalise un nombre de migration bien inférieur à l'approche MmtMiMcHybrid, par contre elle affiche des résultats presque similaires à l'approche MiMcHybrid avec une légère efficacité de cette dernière. On peut noter qu'avec 200 machines virtuelles notre approche provoque un nombre de migration bien réduit que les deux autres approches.

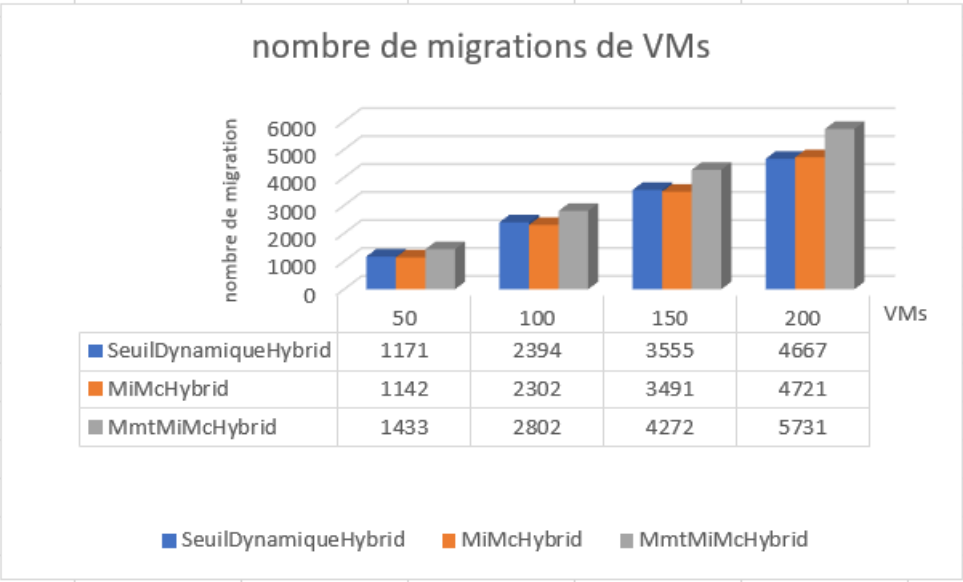

Figure 14. Nombre de migration de VMs

4.3.2.1.3 Violation des SLAs :

Comme illustré dans le graphe (voir Figure 15), on remarque que les deux approches SeuilDynamiqueHybride et MiMcHybrid provoque le plus de violation des SLAs avec des résultats presque similaires, par contre MmtMiMcHybrid enregistre une réduction de violation des SLAs considérable.

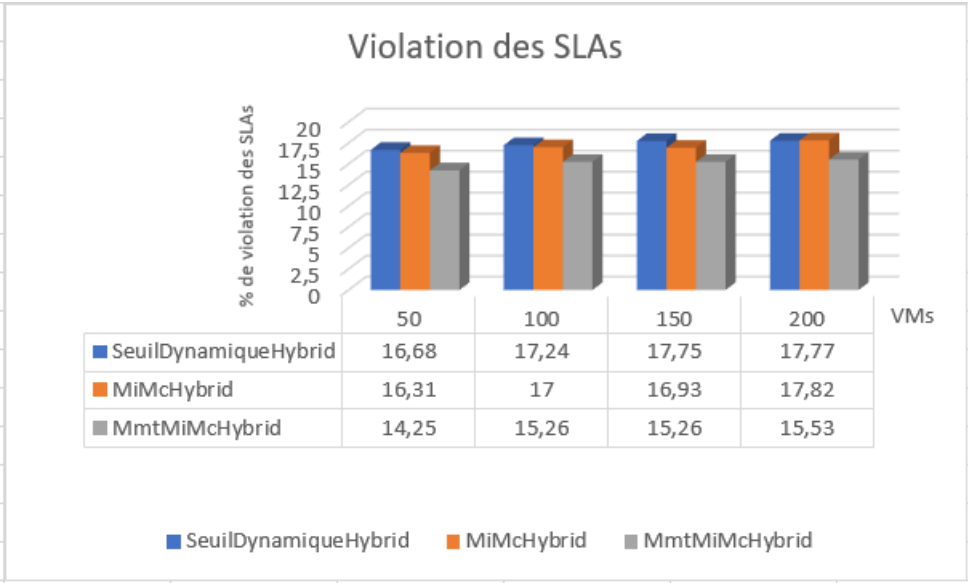

Figure 15. % de violation des SLAs

4.3.2.2 Deuxième Expérience : Un nombre réduit de PMs et VMs

Dans le cadre cette deuxième expérience nous avons considéré 10 machines physiques et 20 machines virtuelles hétérogènes. Le nombre de VMs passe de 5 à 20 sur un pas de 5 grâce au Broker.

4.3.2.2.1 Consommation d'énergie

En observent les résultats obtenus suite à la simulation (voir Figure 16), notre approche SeuilDynamiqueHybrid enregistre une consommation d'énergie réduite par rapport à MiMcHybrid et MmtMiMcHybrid

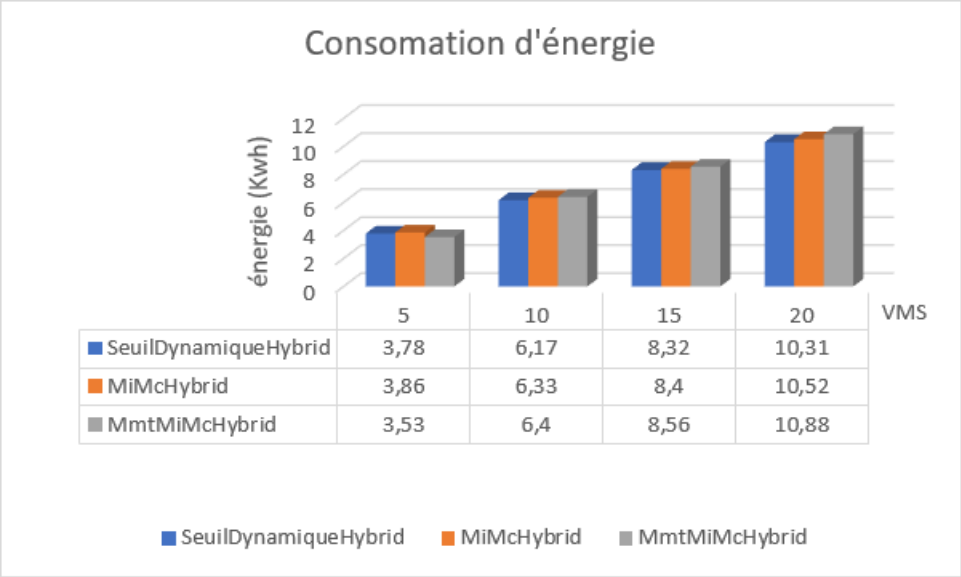

Figure 16. Consommation d'énergie

4.3.2.2.2 Nombre de migration de VMs

Les résultats enregistrés (voir Figure 17) démontrent que l'approche SeuilDynamiqueHybride génère un nombre de migration réduit par rapport aux autres approches avec 5 et 10 machines virtuelles respectivement, mais en augmentant le nombre de machines virtuelles nous notons une hausse considérable des chiffres enregistrés par notre approche tout comme l'approche MmtMiMcHybrid.

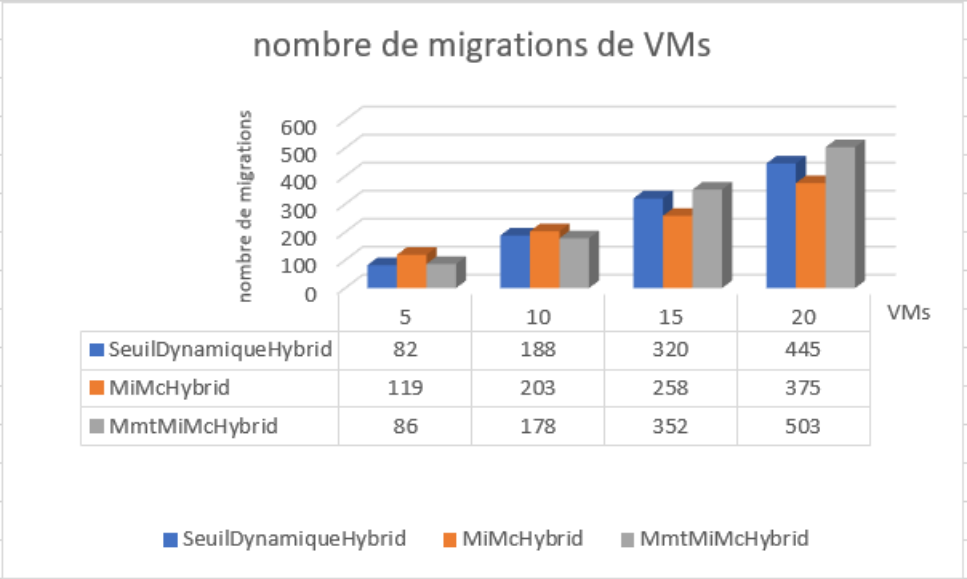

Figure 17. Nombre de migrations de VMs

4.3.2.2.3 Violation des SLAs

Une autre fois les résultats de la simulation (voir Figure 18) montre que l'approche SeuilDynamiqueHybride n'est pas très efficace en ce qui concerne cette métrique. Les résultats enregistrés par MmtMiMcHybrid sont légèrement meilleurs que MiMcHybrid.

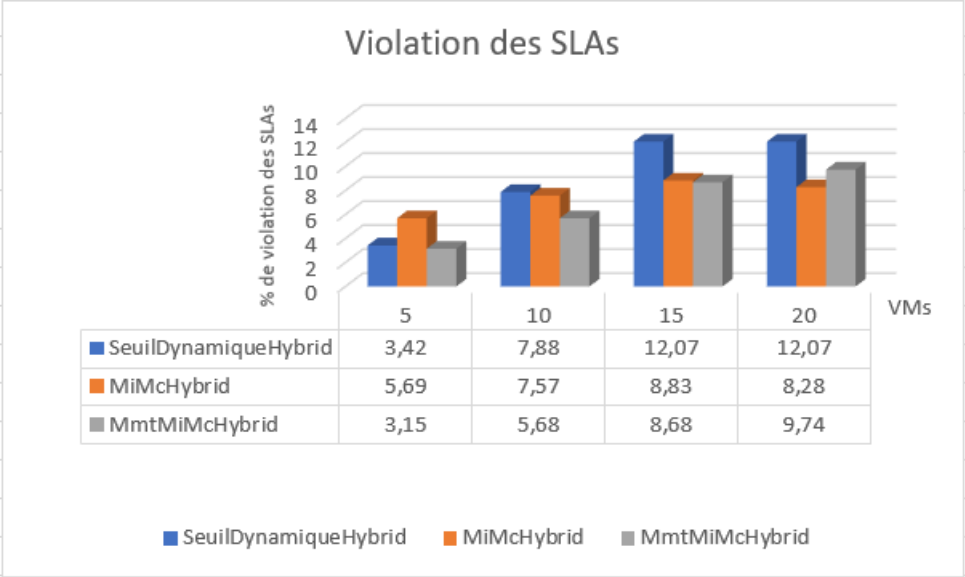

Figure 18. Violation des SLAs

4.4 Conclusion

Le simulateur CloudSim nous a offert la possibilité d'intégrer notre algorithme afin de tester, analyser et comparer notre approche avec d'autres approches présentes dans la littérature, nous avons remarqué que la politique SeuilDynamiqueHybride a obtenu des résultats satisfaisant en ce qui concerne la consommation d'énergie et le nombre de migration de VMs, tandis que lorsque il s'agit de la métrique de violation des SLAs, notre approche n'est pas la plus optimal mais nous avons eu des résultats plus au moins correctes.

En résumé, nous pouvons dire que notre approche « SeuilDynamiqueHybrid » peut être classé parmi les approches de l'informatique verte (Green Computing) grâce à ces performances enregistrées sous CloudSim.

Conclusion Générale

Le Cloud Computing est un nouveau paradigme qui consiste à proposer des ressources informatiques sous forme de services à la demande accessibles de n'importe où, n'importe quand et par n'importe qui.

Ce paradigme s'appuie principalement sur la virtualisation qui permet de faire fonctionner sur une seule machine physique plusieurs systèmes d'exploitation (machines virtuelles), La demande croissante pour ce type de service oblige les fournisseurs des services clouds à augmenter la taille de leur infrastructure pour s'adapter aux changements de charge et garantit la qualité de service (QoS).

Le problème de la gestion des ressources est devenu un sujet très important et d'actualité dans le Cloud Computing, afin d'améliorer l'utilisation des ressources, plusieurs techniques sont utilisées par les fournisseurs des services tels que : la virtualisation et la migration, cette dernière est l'axe de notre travail de recherche.

Ces dernières années, la notion de « Green Computing » a été évoquer fréquemment, cette pratique est axée sur la réduction de la consommation d'énergie. Afin de contribuer à l'optimisation de cette dernière, nous avons proposé une politique dans le cadre de notre travail, cette approche est SeuilDynamiqueHybride qui assure la phase de sélection, elle est basée sur l'utilisation des ressources physiques (CPU, RAM et Bande passante). Dans la réalisation de cet objectif de minimisation de la consommation d'énergie, notre approche a prouvé son efficacité en réduisant également la quantité du CO2 émise et la chaleur produite par les serveurs.

Dans le cadre de notre simulation, il a été question d'étudier le comportement de notre propre approche et comparer les résultats avec les autres approches. Les simulations effectuées sur CloudSim à travers une série d'expériences ont produit des résultats concluants. Les métriques mises en avant sont la consommation d'énergie, le nombre de migrations, la violation SLAs.

Dans la continuité de notre travail, nous envisageons beaucoup des perspectives. Nous projetons dans nos futurs travaux de pouvoir étudier et appliquer cette étude dans un environnement réel de Cloud, Nous voudrons également perfectionner cette approche avec des critères tels que la détection d'augmentation de chaleur et la surcroissance de trafic réseaux.

BIBLIOGRAPHIE

[1] B. L, «lebigdata.fr,» 10 février 2017. [En ligne]. Available: https://www.lebigdata.fr/definition-cloud-computing. [Accès le 18 décemrbre 2021].

[2] *Guide pratique | Virtualisation et cloud computing*, Intel Corporation, 2013.

[3] «Red Hat,» 7 Mars 2018. [En ligne]. Available: https://www.redhat.com/fr/topics/cloud-computing/cloud-vs-virtualization. [Accès le décembre 2021].

[4] «Actualité Informatique,» [En ligne]. Available: https://actualiteinformatique.fr/cloud/definition-virtualisation. [Accès le décembre 2021].

[5] «Red Hat,» 2 mars 2018. [En ligne]. Available: https://www.redhat.com/fr/topics/virtualization/what-is-virtualization. [Accès le Décembre 2021].

[6] Oracle, «Oracle,» [En ligne]. Available: https://www.oracle.com/fr/cloud/definition-machine-virtuelle-vm. [Accès le décembre 2021].

[7] «Oo2 Formations & Consulting,» 7 Mai 2021. [En ligne]. Available: https://www.oo2.fr/actualites/comprendre-la-technologie-virtualisation-et-la-machine-virtuelle. [Accès le Décembre 2021].

[8] «Red Hat,» 9 Septembre 2019. [En ligne]. Available: https://www.redhat.com/fr/topics/virtualization/what-is-a-virtual-machine. [Accès le Decembre 2021].

[9] «ISTARTIPS,» [En ligne]. Available: https://fr.istartips.com/benefits-of-virtualization.html. [Accès le Décembre 2021].

[10] «Red Hat,» 10 Janvier 2020. [En ligne]. Available: https://www.redhat.com/fr/topics/virtualization/what-is-a-hypervisor. [Accès le Decembre 2021].

[11] D. F. Z. Filali, *Concepts et techniques de virtualisation*, 2021.

[12] L. M. &. E. R, «Live Migration of Virtual Machines in Cloud Environment: A Survey,» *Indian Journal of Science and Technology*, vol. 8, pp. 1-7, 2015.

[13] K. Abdelaziz, «VERS UNE GESTION DE MIGRATION DES MACHINES VIRTUELLES POUR AMÉLIORER LA QUALITÉ DE SERVICE ET RÉDUIRE LA CONSOMMATION D'ÉNERGIE DANS LES CLOUD COMPUTING,» 2015.

[14] M. K. M. D. B. e. M. B. P. Pradip D. Patel, «Live Virtual Machine Migration Techniques in Cloud Computing: A Survey,» *International Journal of Computer Applications,* vol. 86, n° %116, pp. 18-21, 2014.

[15] A. S. e. B. Hudzia, «Pre-Copy and Post-Copy VM Live Migration,» *Euro-Par 2012: Parallel Processing Workshops. Euro-Par 2012. Lecture Notes in Computer Science,* vol. 7640, pp. 539-547, 2013.

[16] K. F. S. H. J. G. H. E. J. C. L. I. P. A. W. Christopher Clark, «Live migration of virtual machines,» *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation,* vol. 2, pp. 273-286, 2005.

[17] H. J. X. L. L. H. C. Y. Haikun Liu, «Live migration of virtual machine based on full system trace and replay,» *HPDC '09: Proceedings of the 18th ACM international symposium on High performance distributed computing,* pp. 101-110, 2009.

[18] U. D. ,. K. G. Michael R. Hines, «Post-copy live migration of virtual machines,» *SIGOPS Operating Systems Review,* 2009.

[19] T. H. R. T. a. S. H. S. Akiyama, «MiyakoDori: A Memory Reusing Mechanism for Dynamic VM Consolidation,» *2012 IEEE Fifth International Conference on Cloud Computing,* pp. 606-613, 2012.

[20] B. H. L. a. J. L. a. J. H. a. T. W. a. Q. L. a. L. Zhong, «EnaCloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments,» *2009 IEEE International Conference on Cloud Computing,* pp. 17-24, 2009.

[21] D. A. a. B. P. a. M. T. a. L. Zhang, «Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments,» *IEEE Transactions on Services Computing,* vol. 5, pp. 2-19, 2012.

[22] R. B. a. A. B. a. J. H. Abawajy, «Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges,» *ArXiv,* 2010.

[23] W. S. a. Z. X. a. Q. C. a. H. Luo, «Adaptive Resource Provisioning for the Cloud Using Online Bin Packing,» *IEEE Transactions on Computers,* vol. 63, pp. 2647-2660, 2014.

[24] C. a. H. M. a. Z. D. Ghribi, «Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms,» chez *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, pp. 671-678.

[25] J. D. a. H. W. a. S. Cheng, «Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling,» *China Communications,* vol. 12, pp. 155-166, 2015.

[26] N. B. a. A. K. a. K. A. Beaty, «Dynamic Placement of Virtual Machines for Managing SLA Violations,» *2007 10th IFIP/IEEE International Symposium on Integrated Network Management,* pp. 119-128, 2007.

[27] R. E. «An Improved Algorithm for Optimal Bin Packing, » chez *Proceeding of the 18th International Joint Conference on Artificial Intelligence*, 2003, pp. 1252-1258.

[28] M. C. a. H. Z. a. Y.-Y. S. a. X. W. a. G. J. a. K. Yoshihira, «Effective VM sizing in virtualized data centers,» *12th IFIP/IEEE International Symposium on Integrated*

Network Management (IM 2011) and Workshops, pp. 594-601, 2011.

[29] X. a. W. W.-D. a. B. L. A. Fan, «Power Provisioning for a Warehouse-Sized Computer,» *SIGARCH Comput. Archit. News*, vol. 35, n° %12, pp. 13-23, 2007.

[30] D. K. a. J. K. a. J. E. H. a. N. K. a. G. Jiang, «Power and Performance Management of Virtualized Computing Environments Via Lookahead Control,» chez *ICAC*, 2008.

[31] R. S. a. N. P. a. H. Diwanji, «Article: Energy Efficient Dynamic Integration of Thresholds for Migration at Cloud Data Centers,» *IJCA Special Issue on Communication and Networks*, n° %11, pp. 44-49, 2011.

[32] K. L. T. Lee and H. Huang, «Dynamic resource management for energy saving in the cloud computing environment,» *Proceedings of the International Conference on Information Science and Industrial Applications*, vol. 4, pp. 176-181, 2012.

[33] B. N. A. H. A. u. R. K. S. A. M. Saad Mustafa, «Resource management in cloud computing: Taxonomy, prospects, and challenges,» *Computers & Electrical Engineering*, vol. 47, pp. 186-203, 2015.

[34] A. a. B. R. Beloglazov, «Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers,» *Concurr. Comput. : Pract. Exper.,* vol. 4, pp. 1397-1420, 2012.

[35] J. C. a. Y. W. a. M. Li, «Energy Efficient Allocation of Virtual Machines in Cloud Computing Environments Based on Demand Forecast,» chez *GPC*, 2012.

[36] «Java (langage),» [En ligne]. Available: https://fr.wikipedia.org/wiki/Java_(langage). [Accès le Mai 2022].

[37] R. a. R. R. a. C. R. N. Buyya, «Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities,» chez *2009 International Conference on High Performance Computing & Simulation*, 2009, pp. 1-11.