REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 20 Août 1955-Skikda

Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études pour l'obtention du diplôme de Master professionnel en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Le Recuit Simulé pour le Placement des Machines Virtuelles dans les Centres de Données du Cloud Computing sous Contrainte d'énergie

Réalisé par :

LAOUDJA Djihane

KADDECHE Rayane

Encadré par :

Mr. LAOUAR Walid

Mr. BOUAITA Riad

Session : Juin 2022

Remerciement

D'abord nous tenons à remercier Dieu le tout puissant de nous avoir donné la santé, le courage et la foi pour réaliser ce modeste travail avec volonté.

Nous tenons à exprimer nos gratitude à Mr LAOUAR Walid et Mr BOUAITA Riad en tant que encadrant de ce mémoire. Ils nous ont permis de réaliser ce travail sous leurs directions et aussi pour leurs conseils, orientations durant toute la réalisation de ce travail.

Nous remercions les membres du jury pour avoir accepté d'examiner et évaluer notre travail.

Nous remercions également à tous nos enseignants durant les années de nos études.

Enfin, nous tenons à remercier toute personne qui nous ont aidé et encouragé au long de ce travail.

Dédicace

C’est avec profonde gratitude et sincères mots, que nous dédions ce modeste travail de fin d’étude à nos chers parents qui ont sacrifié leur vie pour notre réussite et nous ont éclairé le chemin par leurs conseils judicieux.

Nous dédions aussi ce travail à nos frères et sœurs, nos familles, nos amis, tous nos professeurs qui nous ont enseigné et à tous ceux qui sont chers.

Résumé

Le Cloud Computing est un paradigme informatique qui utilise l'internet pour fournir des ressources informatiques en tant que service. Il permet aux clients d'accéder rapidement et à la demande aux ressources. Les fournisseurs de services Cloud déploient rapidement des centres de données partout dans le monde.

Dans ce travail, on se focalisera sur le problème de placement de machines virtuelles sur des centres de données d'une infrastructure Cloud. Ce problème consiste à placer les machines virtuelles sur un nombre minimum d'hôtes tout en minimisant la consommation d'énergie. Après avoir mentionné l'importance de réaliser un placement intelligent dans une infrastructure Cloud, on a utilisé l'approche du Recuit Simulé pour l'efficacité d'énergie et pour sauvegarder l'énergie. Cet algorithme tient compte des contraintes multidimensionnelles sur les ressources d'hôte, telles que le processeur, la mémoire vive et la bande passante.

Mots-clés : Cloud Computing, Centre De Donnée, Le Placement De Machines Virtuelles, La Consommation D'énergie, Le Recuit Simulé.

Abstract

Cloud Computing is a Computing paradigm that uses the Internet to deliver computing resources as a service. It provides customers with fast, on-demand access to resources. Cloud service providers are rapidly deploying data centers all over the world.

In this work, we will focus on the problem of placing virtual machines on data centers of a Cloud infrastructure. This problem consists in placing virtual machines on

a minimum number of hosts while minimizing energy consumption. After mentioning the importance of performing an intelligent placement in a Cloud infrastructure, the Simulated Annealing approach was used for energy efficiency and energy saving. This algorithm considers the server resources of multidimensional constraints like CPU, RAM, bandwidth etc.

Keywords: Cloud Computing, Data Center, Virtual Machine Placement, Energy Consumption, Simulated Annealing.

ملخص

الحوسبة السحابية هي نموذج حوسبي يستخدم الإنترنت لتقديم موارد الحوسبة كخدمة. يوفر للعملاء وصولاً سريعًا عند الطلب إلى موارد الحوسبة. يقوم مقدمو الخدمات السحابية بنشر مراكز البيانات بسرعة في جميع أنحاء العالم.

في هذا العمل، سنركز على مشكلة وضع الأجهزة الافتراضية على مراكز البيانات الخاصة بالبنية التحتية السحابية. تتمثل هذه المشكلة في وضع الأجهزة الافتراضية على أقل عدد ممكن من الأجهزة المضيفة مع تقليل استهلاك الطاقة. بعد الإشارة إلى أهمية إجراء التنسيب الذكي في البنية التحتية السحابية ، تم استخدام نهج التلدين المحاكى لكفاءة الطاقة وتوفير الطاقة. تأخذ هذه الخوارزمية في الاعتبار القيود متعددة الأبعاد على موارد المضيف مثل وحدة المعالجة المركزية وذاكرة الوصول العشوائي وعرض النطاق الترددي وما إلى ذلك.

الكلمات المفتاحية: الحوسبة السحابية ، مركز البيانات ، وضع الآلة الافتراضية ، استهلاك الطاقة ، التلدين المحاكي

Table des matières

Sommaire

Remerciement .....
Dédicace .....
Résumé .....
Table des matières .....
Listes des figures .....
Liste des tables .....
Liste des abréviations .....
Introduction Générale 1
CHAPITRE 1 .....
GENERALITES SUR LE CLOUD COMPUTING .....
1.1 Introduction 4
1.2 Définition du Cloud Computing 4
1.3 Technologie connexe 5
1.3.1 La grille informatique 6
1.3.2 L'informatique utilitaire 7
1.3.3 L'informatique autonome 7
1.3.4 La virtualisation 8
1.3.4.1 Les hyperviseurs 9
1.3.4.1.1 Hyperviseur de type 1 10
1.3.4.1.2 Hyperviseur de type 2 10
1.3.4.2 Avantages de la virtualisation 11
1.3.4.3 Inconvénients de la virtualisation 11
1.4 Caractéristiques du Cloud Computing 11
1.5 Les acteurs du Cloud Computing 12
1.6 Les modèles de déploiement du Cloud Computing 14
1.7 Les niveaux de services du Cloud Computing 15
1.8 Les avantages, inconvénients et challenges du Cloud 16
1.8.1 Les avantages 16
1.8.2 Les inconvénients 17
1.8.3 Les challenges 18

1.9 Conclusion 20
CHAPITRE 2 .....
PROBLEME DU PLACEMENT DES MACHINES VIRTUELLES DANS LE CLOUD .....
2.1 Introduction 22
2.2 Paramètres et Considérations 22
2.3 Placement de machines virtuelles dans le Cloud Computing 23
2.3.1 Placement statique 24
2.3.2 Placement dynamique 26
2.4 Stratégies de placement des VMs dans les datacenter du Cloud 28
2.4.1 Les méthodes exactes 29
2.4.2 Les méthodes approchées 30
2.4.2.1 Méthodes heuristiques 31
2.4.2.2 Méthodes méta-heuristiques 31
2.4.2.2.1 Méta-heuristiques à solution unique 32
2.4.2.2.1.1 Méthodes de recherche locale 32
2.4.2.2.1.2 Le recuit simulé 33
2.4.2.2.1.3 La recherche Taboue 33
2.4.2.2.2 Méta-heuristiques à base de population de solutions : 34
2.4.2.2.2.1 L'algorithme évolutionnaire 34
2.4.2.2.2.2 L'algorithme génétique 34
2.4.2.2.2.3 L'optimisation par essaims de particules (PSO) 35
2.4.2.2.2.4 L'algorithme de colonies de fourmis 35
2.4.2.2.2.5 L'algorithme de la recherche harmonique 36
2.5 Objectifs d'une solution de placement de VM dans un environnement Cloud 36
2.6 Indicateurs de performance pour un algorithme de placement de VMs 37
2.7 Conclusion 39
CHAPITRE 3 .....
RECUIT SIMULE AMELIORE POUR LE PLACEMENT DES VMS DANS LE CLOUD .....
3.1 Introduction 41
3.2 Le recuit simulé 41
3.2.1 Origines 41
3.2.2 Définition 42
3.2.3 Algorithme de Metropolis 43
3.2.4 Algorithme du recuit simulé 43

3.2.5 Avantages et Inconvénients du recuit simulé 46
3.2.5.1 Avantages 46
3.2.5.2 Inconvénients 46
3.2.6 Domaines d'applications 46
3.3 Le problème de placement de VMs et le recuit simulé 47
3.3.1 Formulation Du Problème 47
3.3.2 Algorithme du Recuit Simulé Amélioré pour le placement des VMs 49
3.3.3 Description De L'algorithme 51
3.4 Conclusion 54
CHAPITRE 4 .....
IMPLEMENTATION ET RESULTATS EXPERIMENTAUX .....
4.1 Introduction 57
4.2 Langage et environnement 57
4.2.1 Le langage de programmation Java 57
4.2.2 Environnement de développement NetBeans 59
4.2.3 Simulateur CloudSim Plus 59
4.2.3.1 Architecture générale de CloudSim Plus 60
4.2.3.2 Les classes de CloudSim Plus 63
4.3 Interface de l'application 65
4.4 Résultats expérimentaux 69
4.5 Conclusion 70
Conclusion générale 71
Bibliographies 72

Listes des figures

Figure 1.1 - Cloud Computing ……...5

Figure 1.2 - Grille Informatique "Grid Computintg"……..6

Figure 1.3 - Informatique Utilitaire "Utility Computing"……...7

Figure 1.4 - Informatique Autonome "Autonomic Computing" "……..8

Figure 1.5 - Virtualisation …….....9

Figure 1.6 - Types d'hyperviseurs ……10

Figure 1.7 - Caractéristiques du Cloud Computing…… 12

Figure 1.8 - Les modèles de déploiement du Cloud……..…14

Figure 2.1 - Placement statique de machines virtuelles sur les centres de donné…….24

Figure 2.2 - Placement statique des machines virtuelles sur des machines physiques……26

Figure 2.3 - Placement dynamique de machine virtuelle…….28

Figure 2.4 - Classes des méthodes de résolutions ……...29

Figure 2.5 - Classes des méta-heuristiques …….32

Figure 3.1 - Résultat du refroidissement du métal (rapide et lente) ……....43

Figure 3.2 - Fonctionnement de l'algorithme du Recuit Simulé……..45

Figure 3.3 - Inversion……...53

Figure 3.4 - Traduction…….53

Figure 3.5 - Commutation……....54

Figure 4.1 - Logo de langage Java……60

Figure 4.2 - Structure du package de l'API CloudSim Plus …….62

Figure 4.3 - Classes principales impliquées dans la création de simulations CloudSim Plus…...64

Figure 4.4 - Interface principale……....67

Figure 4.5 - Création des machines virtuelles……68

Figure 4.6 - Création des hôtes……..69

Figure 4.7 - Comparaison énergétique entre le RSA, PA et PRR…….……...70

Liste des tables

TABLEAU 1.1 - AVANTAGES ET INCONVENIENTS DES SERVICES CLOUD. 16
TABLEAU 1.2 - AVANTAGES DU CLOUD COMPUTING 17
TABLEAU 1.3 - INCONVENIENTS DU CLOUD COMPUTING 18

Liste des abréviations

AMDV	Advanced Micro Devices Virtualisation
API	Application Programming Interface
CPU	Central Processing Unit
DC	Data Center
E/S	Entrée Sortie
ESX	Elastic Sky X
GA	Genetic Algorithm
HTML	HyperText Markup Language
IaaS	Infrastructure AS A Service
IHM	Interface Homme Machine
IDE	Integrated DevelopmentEnvironment
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDK	Java Development Kit
JRE	Java Runtime Environment

JSP	Java Server Pages
MIDP	Mobile Information Devices Profile
NIST	National Institute of Standards and Technology
NP-Complet	Non déterministe Polynomial
OS	Operating System
PaaS	Platform As A Service
PC	Personal Computer
PDM	Performance Degradation due to Migration
PHP	Hypertext PreProcessor
PL	Programmation Linéaire
PM	Physical Machine
PSO	Particle Swarm Optimization
QEMU	Quick EMUlator
QOS	Quality Of Service
RAM	Random Access Memory
ROI	Return On Investment
RSA	Recuit Simulé Amélioré
SA	Simulated Annealing

SaaS	Software As A Service
SAN	Storage Area Network
SAVMP	Simulated Annealing Virtual Machine Placement
SLA	Service Level Agreement
SLAV	Service Level Agreement Violation
SoC	Separation of Concerns
SW	Software
VM	Virtual Machine
VMM	Virtual Machine Monitor
VMPP	Virtual Machine for Parallel Processing
VMs	Virtual Machines
VPN	Virtual Private Network
VT	Virtual Technology
XML	Extensible Markup Language

Introduction Générale

Ces dernières années, l'informatique en nuage (appelée Cloud Computing en anglais), s'est développée comme un paradigme majeur pour l'utilisation des ressources informatiques. Ces ressources fournies par les fournisseurs du Cloud sont accessibles via un réseau informatique, via des interfaces programmées (généralement via Internet). Le Cloud Computing rend de nombreuses ressources, telles que le processeur et le stockage, et les applications disponibles sous forme de services Internet à tout moment et depuis n'importe quel endroit.

Avec le développement du Cloud Computing, l'échelle des centres de données continue de s'étendre et un centre de données Cloud utilise généralement des milliers de machines physiques. La technologie de virtualisation permet à une machine physique de prendre en charge plusieurs machines virtuelles.

Le placement des machines virtuelles a un impact direct sur la stabilité du service Cloud, l'efficacité des ressources, la satisfaction des utilisateurs et les dépenses d'exploitation. Cela démontre l'importance de la recherche sur le problème de l'emplacement des machines virtuelles dans les centres de données du Cloud Computing.

Le problème du placement des VMs dans les centres de données du Cloud Computing est un problème NP-Complet, il ne peut pas être résolu en temps polynomial. Alors on a besoin d'un algorithme heuristique pour trouver la solution optimale.

Dans cette recherche, nous utilisons l'algorithme du recuit simulé qui s'inspire du processus de recuit métallurgie pour le nouveau problème de placement de machines virtuelles qui considère la consommation d'énergie dans les centres de données.

Les résultats expérimentaux montrent que l'algorithme fonctionne bien lorsque l'on s'attaque à des problèmes de test de différents types, et évolue bien lorsque la taille du problème augmente.

Essentiellement, ce mémoire est structuré autour de quatre chapitres :

- Le premier chapitre montre les concepts et les notions fondamentales du Cloud Computing.

- Le deuxième chapitre illustre le principe de placement, les concepts d'heuristiques et de méta-heuristiques, à savoir, celles à base de solution unique et celles à base de population de solution.

- Le troisième chapitre présente l'approche du Recuit Simulé pour la résolution du problème de placement des machines virtuelles dans les centres de donnés du Cloud Computing de manière détaillée.

- Le quatrième chapitre décrit la phase de validation de notre approche proposée pour le placement des machines virtuelle dans le Cloud Computing.

CHAPITRE 1

GENERALITES SUR LE CLOUD COMPUTING

1.1 Introduction

Dans un monde où l'information change constamment, nous recherchons la meilleure façon de stocker nos données. L'externalisation des données vers des centres de données est une solution qui devrait être étudiée pour voir si elle apporte un réel avantage. L'accès à ces données stockées est fourni par des services.

Le Cloud Computing est un terme général qui désigne un ensemble de technologies utilisées pour fournir des services. Son objectif est d'inciter les entreprises à externaliser et dématérialiser une partie de ses ressources numériques. Ces ressources, qui comprennent des capacités de stockage et de calcul, des logiciels de gestion de messagerie et d'autres services, sont mises à disposition par des entreprises tierces et sont accessibles via une connexion Internet.

Dans ce chapitre, nous allons présenter les notions fondamentales du Cloud Computing, dans un premier temps nous allons étudier le Cloud Computing de manière générale, en présentant ses définitions, caractéristiques et les technologies de virtualisation, puis nous allons étudier les trois services principaux, sur lesquels le Cloud Computing se repose: applicatif, plateforme et infrastructure, et qui ont donné naissance par la suite au fameux SaaS/PaaS/IaaS, et nous conclurons par quelques avantages, inconvénients et challenges du Cloud Computing.

1.2 Définition du Cloud Computing

Dans la littérature, plusieurs définitions du Cloud Computing ont été envisagées. La définition la plus utile et la plus complète est mise introduite par NIST (*National Institute of Standards and Technology*), qui a défini le Cloud Computing comme [1]:

Le Cloud Computing est un modèle informatique qui permet un accès facile et à la demande par le réseau internet à un ensemble partagé de ressources informatiques configurables. Ces ressources sont par exemple des réseaux, des serveurs, des espaces de stockage, des applications et des services. Elles peuvent être rapidement provisionnées et libérées avec un minimum d'efforts de gestion ou d'interaction avec le fournisseur de services.

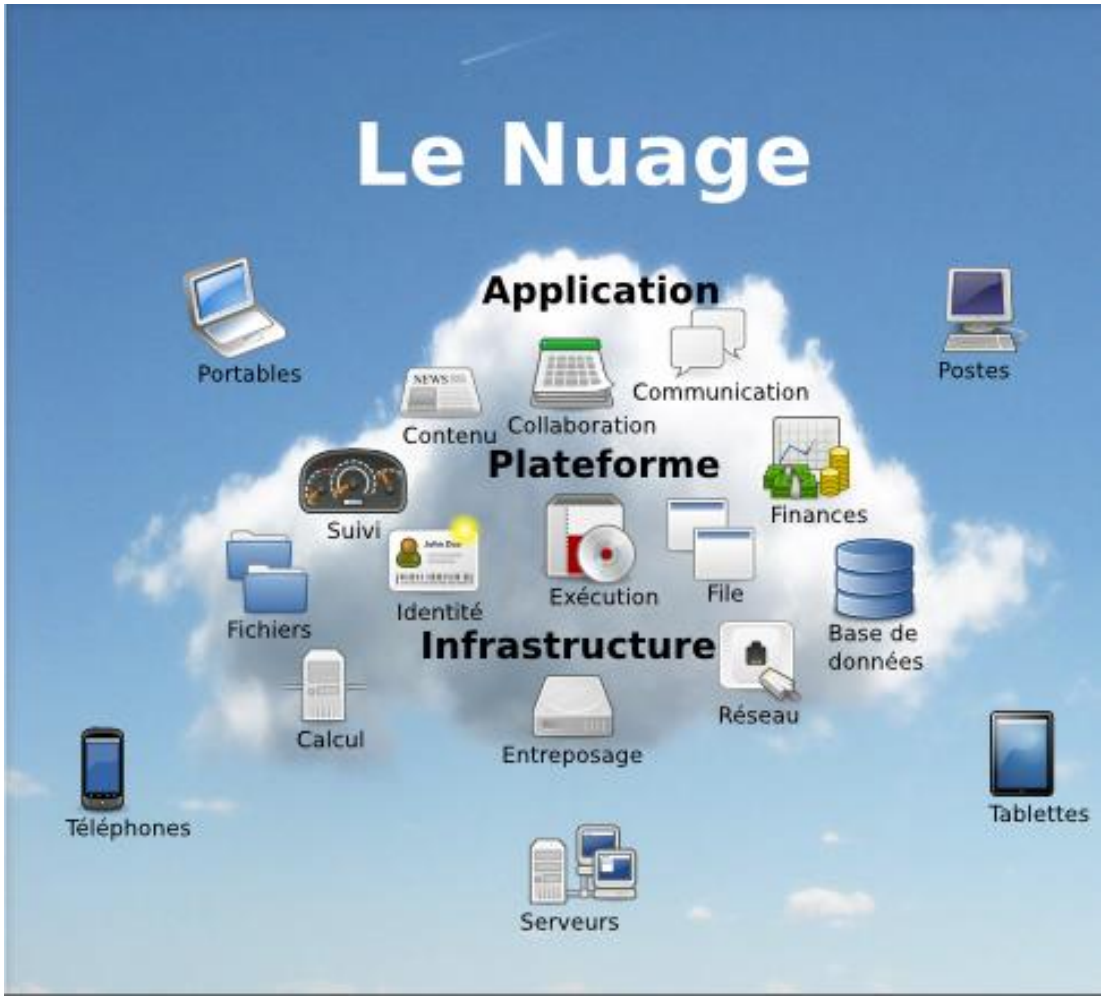


Figure 1.1 - Cloud Computing.

Selon Amazon [2], le Cloud Computing est la fourniture à la demande de ressources informatiques sur internet avec tarification à l'utilisation. Au lieu d'acquérir, et d'entretenir des infrastructures gigantesques tel que les centres de données physiques et les serveurs, on peut accéder à des services technologiques, tels que la puissance de calcul, le stockage et les bases de données, selon vos besoins auprès d'un fournisseur de Cloud comme Amazon Web Services (AWS).

Selon le Syntec [3], le Cloud est une interconnexion et une coopération de ressources informatiques, localisées au sein d'une même entité ou dans différentes structures internes, externes ou mixtes, et dont le mode d'accès s'appuie sur les protocoles et les standards Internet.

1.3 Technologie connexe

Le Cloud Computing a évolué en se basant sur diverses technologies modernes dont il a hérité des aspects et des fonctionnalités, à savoir : la Grille Informatique, l'informatique utilitaire, la virtualisation et l'informatique autonome [4] :

1.3.1 La grille informatique "Grid Computing"

Le calcul en grille est un paradigme de calcul distribué qui coordonne des ressources autonomes et géographiquement distribuées pour atteindre un objectif de calcul commun. Le grid est basé sur le partage dynamique des ressources entre personnes, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications scientifiques, qui sont généralement des calculs intensifs ou des traitements de très gros volumes de données. Le Cloud Computing et le grid Computing sont dans la mesure où ils utilisent tous deux le concept de fourniture de ressources sous forme de services. Cependant, ils diffèrent dans les buts qu'ils visent .Bien que la technologie des grilles vise essentiellement à permettre à des groupes différents de partager l'accès en libre-service à leurs ressources, le Cloud a pour objectif de fournir aux utilisateurs des services « à la demande » basés sur le principe du « paiement à l'usage ». De plus, le Cloud exploite les technologies de virtualisation à plusieurs niveaux (matériel et plateforme d'application) pour permettre le partage et l'approvisionnement dynamique des ressources.

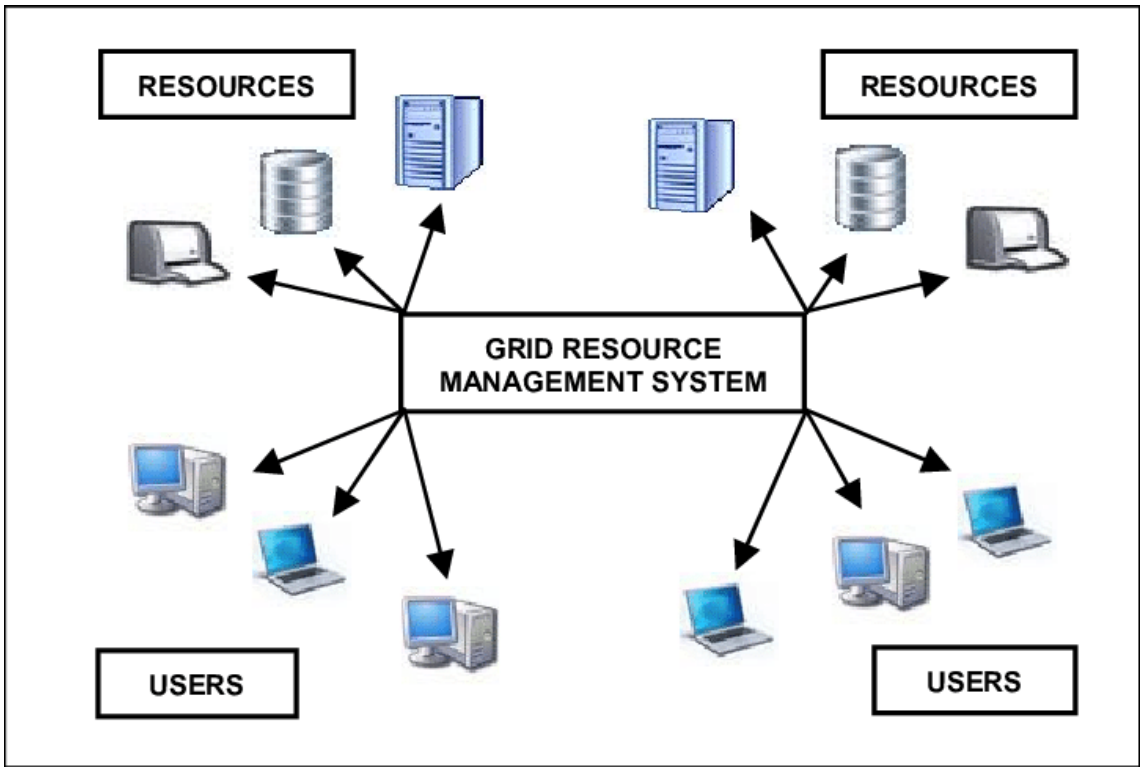


Figure 1.2 – Grille Informatique "Grid Computintg" [5].

1.3.2 L'informatique utilitaire "Utility Computing"

Est une délocalisation d'un système de calcul ou de stockage. Il présente un modèle d'affectation des ressources à la demande et de facturation des utilisateurs en fonction de leur utilisation. Le Cloud Computing peut être considéré comme une réalisation de l'informatique utilitaire. Il adopte un système de tarification basé sur les services publics uniquement pour des raisons économiques. Avec l'approvisionnement des ressources à la demande et le paiement à l'usage, les fournisseurs de services peuvent augmenter l'utilisation des ressources et minimiser leurs coûts d'exploitation.

Figure 1.3 – Informatique Utilitaire "Utility Computing" [6].

1.3.3 L'informatique autonome "Autonomic Computing"

L'informatique autonome vise à créer des systèmes informatiques capables de s'auto-administrer en s'adaptant à des changements internes et externes sans intervention humaine. Le but de l'informatique autonome est de surmonter la complexité de la gestion des systèmes informatiques d'aujourd'hui. Bien que le Cloud Computing présente certaines caractéristiques autonomes comme le provisionnement automatique des ressources, son objectif est de réduire le coût des ressources plutôt que de réduire la complexité du système.

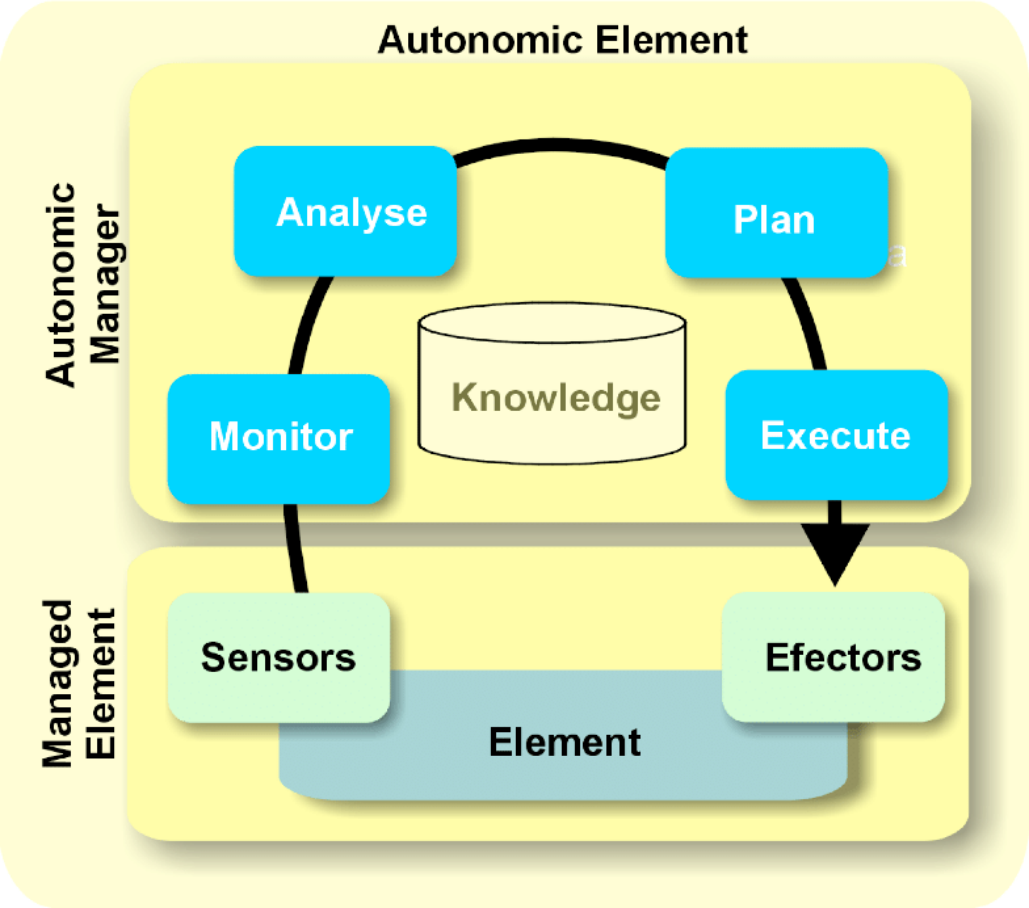


Figure 1.4 – Informatique Autonome "Autonomic Computing" [7].

1.3.4 La virtualisation

La virtualisation est l'un des composants majeurs du Cloud Computing qui a contribué à son développement .

Les principaux composants de la virtualisation dans le Cloud sont les machines virtuelles, car tous les systèmes d'exploitation et les applications sont à l'intérieur même si elles partagent le même hôte physique, elles sont comme des conteneurs qui sont isolés et séparés les uns des autres [8].

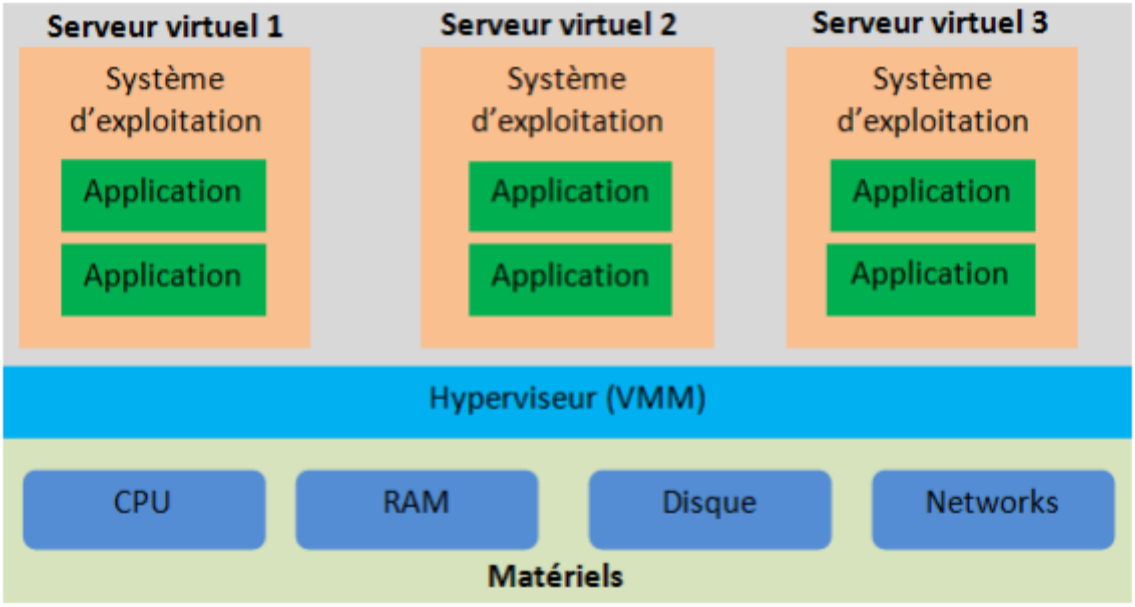


Figure 1.5 - Virtualisation [9].

1.3.4.1 Les hyperviseurs

Un hyperviseur ou VMM (Virtual Machine Manager) est un plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de fonctionner simultanément sur la même machine physique [8].

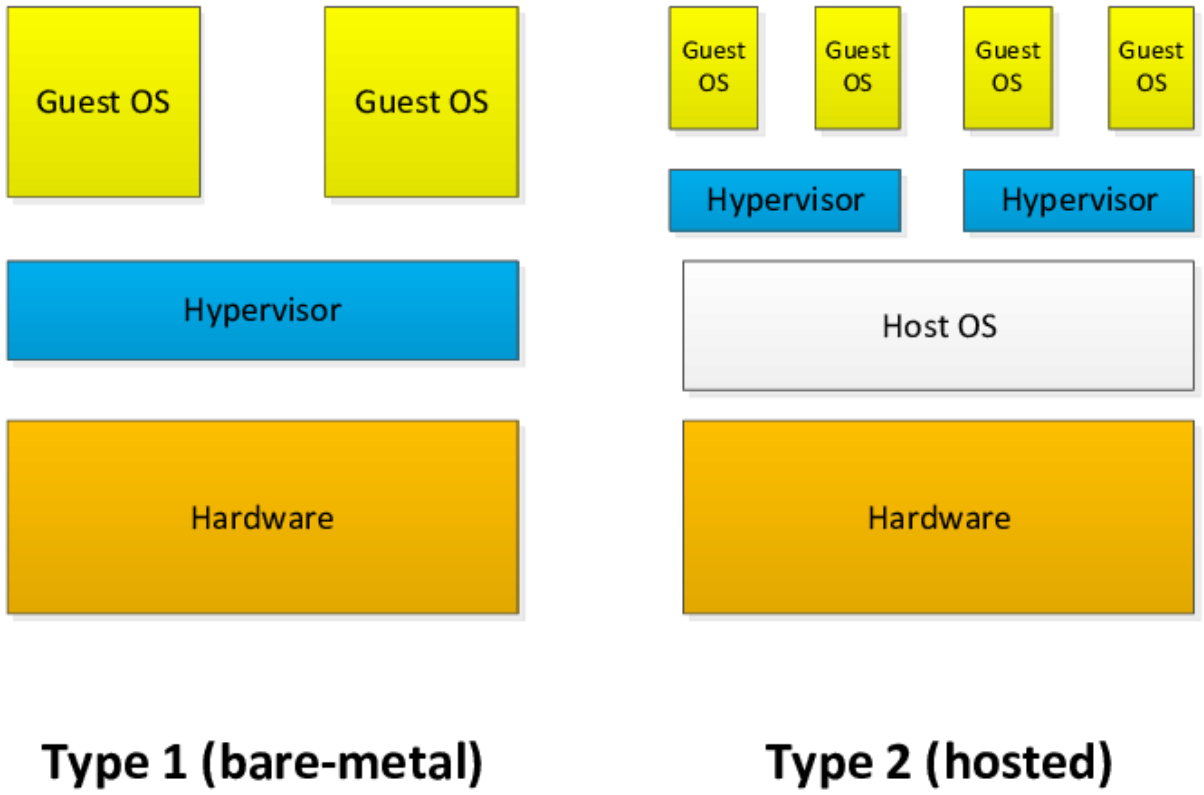


Figure 1.6 - Types d'hyperviseurs [10].

1.3.4.1.1 Hyperviseur de type 1

Un hyperviseur bare metal (ou de type 1), est un logiciel qui s'exécute directement sur une plateforme matérielle (machine physique). Il est considéré comme un noyau hôte allégé et optimisé pour exécuter des noyaux de systèmes d'exploitation invités qui sont adaptés et optimisés pour cette architecture. Par conséquent, cette plateforme est considérée comme un outil de gestion et d'allocation de ressources physiques. Elle gère l'abstraction entre ces ressources et les machines virtuelles (les systèmes d'exploitation invités). Les machines virtuelles sont exécutées sur l'hyperviseur [11].

Quelques exemples de ces hyperviseurs sont Xen, ESX Server de Vmware.

1.3.4.1.2 Hyperviseur de type 2

Il est également connu sous le nom de VMM hébergé car l'hyperviseur s'exécute comme une application dans un système d'exploitation normal connu comme système d'exploitation hôte. Le système d'exploitation hôte n'a aucune connaissance de la VMM de type 2, donc il la traite comme tout autre processus [11].

1.3.4.2 Avantages de la virtualisation

La virtualisation des systèmes informatiques présente de nombreux avantages citant [12]:

- Déploiement rapide des applications,
- Niveaux de service supérieur et disponibilité accrue des applications,
- Évolutivité rapide et flexible,
- Réduction des coûts énergétiques, d'infrastructure et des installations,
- Diminution des frais de gestion,
- Accès aux applications et aux données des bureaux à tout moment,
- Sécurité informatique renforcée.

1.3.4.3 Inconvénients de la virtualisation

Malgré tous ces avantages, la virtualisation présente quelques inconvénients : [9]

- **Coût important** : pour qu'une architecture virtualisée fonctionne correctement, l'entreprise doit investir dans un serveur physique disposant de plusieurs processeurs et mémoires.
- **Pannes généralisées** : si la machine physique tombe en panne, les machines virtuelles tombent aussi en panne.
- **Vulnérabilité généralisée** : si l'hyperviseur est endommagé ou exposé à une faille de sécurité, les machines virtuelles sont également vulnérables et ne sont plus protégées. En augmentant le nombre de couches logicielles, la virtualisation a augmenté la surface d'attaque de l'entreprise.

1.4 Caractéristiques du Cloud Computing

Selon le NIST, le Cloud Computing doit présenter 5 caractéristiques essentielles [1] :

Libre-service à la demande : les ressources et les services offerts par les fournisseurs sont toujours disponibles à la demande des clients.

large accès au réseau : le client peut accéder aux ressources le plus naturellement possible, n'importe où, à tout instant et à partir de divers dispositifs (ordinateur portable, Smartphone, tablette…).

Mutualisation des ressources : les ressources informatiques du fournisseur sont mises en commun de manière à servir plusieurs clients. Un espace de stockage, du temps de calcul, de la bande passante réseau et des machines virtuelles sont autant d'exemples de ressources.

Elasticité et extensibilité rapide : Les capacités matérielles doivent être élastiques. Le client peut en ajouter ou en supprimer comme il le veut. Pour ce dernier, les ressources semblent illimitées et peuvent être affectées en n'importe quelles quantités à tout moment.

Service mesuré : Les systèmes du Cloud contrôlent et optimisent automatiquement l'utilisation des ressources. L'utilisation des ressources peut être surveillée, contrôlée et indiquée de manière à offrir une certaine transparence au fournisseur et au client du service et permettre ainsi au client de ne payer que ce qu'il consomme.

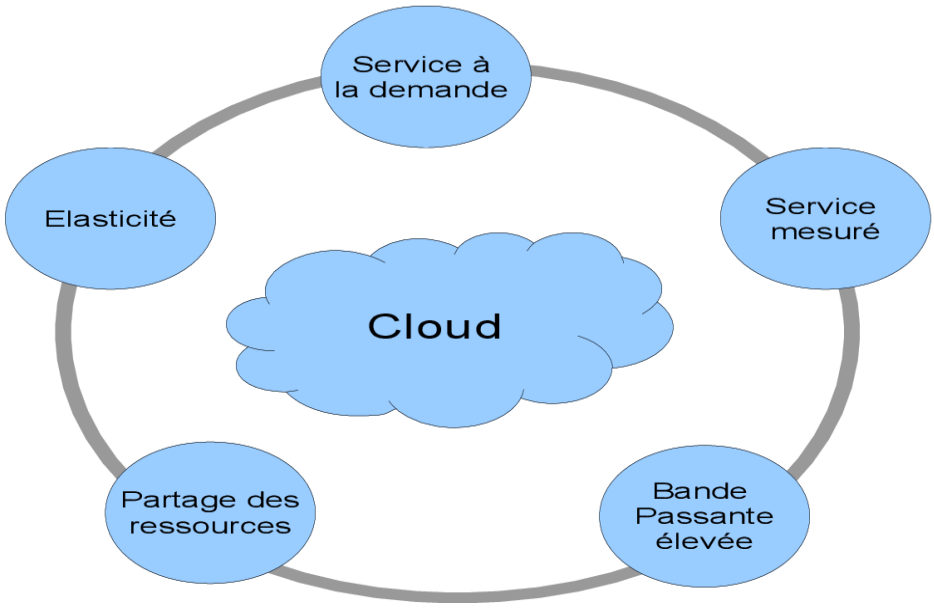


Figure 1.7 - Caractéristiques du Cloud Computing.

1.5 Les acteurs du Cloud Computing

L'écosystème du Cloud Computing est composé principalement par cinq acteurs majeurs [1] :

1. Le fournisseur du Cloud "Cloud Provider" : Il est responsable de fournir un service Cloud Computing qui satisfait les caractéristiques définies dans la précédente section, tout en respectant les accords de niveau services (SLA : Service Level Agreement) établies avec les autres acteurs (en particulier le Cloud Consumer). L'activité du fournisseur de services en nuage consiste à attribuer, organiser et gérer les ressources qu'il fournit tout en assurant le bon niveau de sécurité.

2. Le consommateur du Cloud "Cloud Consumer" : C'est l'utilisateur des ressources Cloud Computing. Selon le type de service Cloud utilisé, cet utilisateur peut être un utilisateur final ou un développeur. Cet utilisateur peut être une personne, un groupe de personnes, des petites et moyennes entreprises, des sociétés internationales ou des gouvernements.

3. Le transporteur "Cloud Carrier" : Le transporteur est l'intermédiaire qui assure principalement la connectivité entre les ressources Cloud Computing et la connectivité entre les acteurs de l'écosystème Cloud Computing (en particulier entre le Cloud Provider et le Cloud Consumer). Cet utilisateur peut jouer un simple rôle d'acheminements des paquets, comme il peut effectuer des tâches plus complexes en offrant des fonctionnalités avancées dans le réseau. Ces fonctionnalités reposent sur des SLAs établies avec les autres acteurs de l'écosystème.

4. Le courtier "Cloud Broker" : Le courtier Cloud est un intermédiaire qui négocie la relation entre les fournisseurs et les consommateurs du Cloud. Il peut offrir de nouveaux services qui facilitent les tâches de gestion du consommateur Cloud. Ce dernier peut demander des ressources Cloud Computing au courtier au lieu du fournisseur direct.

5. L'auditeur "Cloud Auditor" :L'auditeur Cloud est chargé de la vérification et l'audition des services Cloud Computing. Il évalue les services fournis par les fournisseurs Cloud, le transporteur et le courtier du point de vue performances et sécurité. L'objectif principal est de vérifier que les fournisseurs respectent bien les SLAs qu'ils proposent.

1.6 Les modèles de déploiement du Cloud Computing

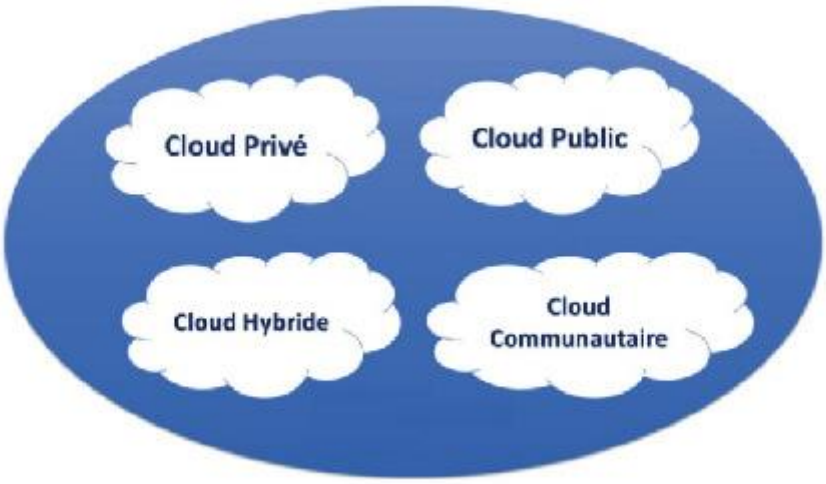


Figure 1.8 - Les modèles de déploiement du Cloud.

Le modèle de déploiement est généralement divisé en quatre catégories : Cloud public, privé, communautaire et hybride. Leurs descriptions ont été introduites par NIST [1] :

Cloud public : L'infrastructure d'un Cloud public est accessible à un large public et appartient à un fournisseur de services. Ce dernier facture les utilisateurs en fonction de leur consommation et garantit la disponibilité du service via des contrats SLA.

Cloud privé : L'ensemble des ressources d'un Cloud privé est exclusivement disponible pour une seule entreprise ou organisation. Le Cloud privé peut être géré par l'entreprise elle-même (Cloud privé interne) ou par une partie tierce (Cloud privé externe). Les ressources d'un Cloud privé sont souvent situées dans les locaux de l'entreprise ou bien chez un fournisseur de services. Dans ce dernier cas, l'infrastructure est entièrement dédiée à l'entreprise et y est accessible via un réseau sécurisé (de type VPN). L'utilisation d'un Cloud privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

Cloud communautaires : L'infrastructure d'un Cloud communautaire est partagée par plusieurs organisations indépendantes ayant des intérêts communs. L'infrastructure peut être gérée par les membres de l'organisation ou par un tiers. L'infrastructure peut se trouver soit au sein de ces organisations, soit chez un fournisseur de services.

Cloud hybride : L'infrastructure d'un Cloud hybride est une composition de plusieurs Clouds (privé, communautaire ou public). Les différents Clouds qui composent l'infrastructure restent des entités uniques, mais sont reliés par une technologie standard ou propriétaire permettant ainsi la portabilité des données ou des applications déployées sur les différents Clouds.

1.7 Les niveaux de services du Cloud Computing

Dans le Cloud Computing, les fonctionnalités sont offertes aux clients comme des services. Ces services peuvent être devisés selon la nature du service livré en trois grands groupes [13] :

Software as a Service (SaaS)

Les services SaaS du Cloud Computing permettent de mettre des applications prêtes à l'emploi à la disposition des utilisateurs. Concrètement, lorsqu'un client sollicite une solution SaaS, il se connecte simplement au service depuis un navigateur. Les SaaSs s'adressent donc aux utilisateurs finaux.
Exemple : messagerie, sauvegarde en ligne.

Platform as a Service (PaaS)

Les services du type PaaS fournissent des environnements spécialisés au développement d'applications comprenant les outils et les modules nécessaires pour ce type de travail, il s'agit d'environnements d'exécution qui permettent de gérer le cycle de vie des applications. Ce cycle de vie comprend notamment les phases de conception, de déploiement et plus largement d'administration des applications.
Exemple : hébergement des sites web.

Infrastructure as a Service (IaaS)

Les services IaaS du Cloud Computing mettent à disposition des utilisateurs des ressources matérielles (réseau, stockage, systèmes d'exploitation) accessibles sous format virtuelle.
Exemple : hébergement de serveurs virtuels.

Les avantages et les inconvénients de chaque service sont résumés dans le tableau 1.1 :

	avantage	**Inconvénient**
SaaS	- pas d'installation - plus de licence - migration	- logiciel limité - sécurité - dépendances des prestataires
PaaS	- pas d'infrastructure nécessaire - pas d'installation - environnement Hétérogène	- limitation des langages - pas de personnalisation dans la configuration des machines virtuelles
IaaS	- administration - personnalisation - flexibilité d'utilisation	-sécurité -besoin d'un administrateur système

Tableau 1.1 - Avantages et Inconvénients des services Cloud.

1.8 Les avantages, inconvénients et challenges du Cloud

Comme le Cloud Computing est une technologie émergente de plusieurs autres technologies connexes. aborder ce sujet nécessite de mettre en évidence les atouts offerts par le Cloud par rapports à ces technologies. D'une autre part De nombreux problèmes existants n'ont pas été entièrement résolus, tandis que de nouveaux challenges entravent l'adoption complète du Cloud Computing [14].

1.8.1 Les avantages

Catégorie	Avantages
Économie	- Économies d'échelle pour les fournisseurs - Réduction des coûts de formation - Réduction de la consommation d'électricité - Entrée facile pour les pays en développement - Accès facile pour les start-ups et les PME
Évolutivité	- Ressources informatiques à la demande
Performance	- Bonne qualité des services - Opérations simplifiées - Machines et services robustes offerts
Innovation	- Nouvelles applications et nouveaux services - Réduction des obstacles informatiques à l'innovation - Nouveaux marchés
Utilisation	- Accès facile pour les utilisateurs - Utilisation optimisée des ressources
Ubiquité	- Accès omniprésent aux données et aux services : partout, tout le temps, de toute manière

Tableau 1.2 - Avantages du Cloud Computing

1.8.2 Les inconvénients

Catégorie	Risques
Sécurité et confidentialité	- La confidentialité des données est un obstacle à l'informatique dématérialisée - Les données sensibles ne conviennent pas au Cloud - Attaques internes et externes - Perte potentielle de données
Conformité	- Réglementation et intégrité des lois - Emplacement des données critique
Intégration	- Résistance culturelle au changement - Intégration de nouvelles apps - Inadaptation à la migration de certaines applications existantes
Normalisation	- Personnalisation limitée - Compétitivité affectée
Fiabilité	- la disponibilité des serveurs - Offres de fournisseurs non fiables - Congestion - Bogues dans les grands systèmes distribués - Mauvaise connectivité à large bande
Compétences	- Manque de compétences et de formation - Ne pas comprendre comment utiliser les technologies Cloud

Tableau 1.3 - Inconvénients du Cloud Computing

1.8.3 Les challenges

Le Cloud Computing est devenu une préoccupation majeure pour les systèmes informatiques de toutes les entreprises. Pour évaluer l'impact que le Cloud peut avoir sur les entreprises, il est nécessaire d'évaluer d'un point de vue critique les challenges de recherche dans le Cloud. Nous allons résumer dans cette section quelques enjeux de recherche dans le Cloud Computing [15] :

Migration des données entre les environnements non standards : La majorité des fournisseurs de services de Cloud Computing utilisent des applications propriétaires. Ces applications ne sont pas interopérables, ce qui rend extrêmement difficile pour les utilisateurs de faire passer leurs données à un autre fournisseur de Cloud ou de revenir à leurs machines.

Sécurité de données et confidentialité : Dans le Cloud Computing, les données doivent être transférées entre les dispositifs de l'utilisateur et les centres de données des fournisseurs de services de Cloud Computing, ce qui les rendra cible facile pour les pirates. La sécurité et la confidentialité des données doivent être assurées, que ce soit sur le réseau ou encore dans les centres de données de Cloud où elles seront stockées.

Migration de machines virtuelles : La virtualisation peut offrir des avantages importants dans le Cloud Computing en permettant la migration de machine virtuelle pour équilibrer la charge de travail entre les Datacenter. Elle permet un approvisionnement des datacenters très fiable et réactif. Les principaux avantages de la migration de VM est d'éviter les points chauds (hot spots). Cependant, cela n'est pas une tâche facile à accomplir. Actuellement, la détection de points chauds et l'initiation d'une migration manque de souplesse pour répondre aux changements soudains de charge de travail.

Consolidation de serveurs : Dans un environnement de Cloud Computing, la consolidation des serveurs est un moyen efficace de maximiser l'utilisation des ressources tout en réduisant la consommation d'énergie. La technologie de migration de VM est fréquemment utilisée pour consolider les machines virtuelles se trouvant sur plusieurs serveurs sous-utilisés sur un seul serveur, de sorte à mettre ces derniers en mode d'économie d'énergie. Le problème de la consolidation optimale des serveurs est un problème d'optimisation NP-complet. En ce qui concerne les ressources du serveur qui sont partagées entre les machines virtuelles, telles que la bande passante, la mémoire cache et les E/S disques, consolider au maximum un serveur peut entraîner la congestion des ressources lorsqu'une machine virtuelle modifie son utilisation de ressource sur le serveur. Par conséquent, il est parfois important d'observer les fluctuations des traces de VM et d'utiliser cette information pour une consolidation efficace de serveurs.

Gestion de l'énergie : L'amélioration de l'efficacité énergétique est un autre enjeu majeur dans le Cloud Computing. Il a été estimé que le coût d'alimentation et du refroidissement représentent 53% des dépenses de fonctionnement total des Datacenter. En 2006, les Datacenter aux États-Unis ont consommé plus de 1,5% de l'énergie totale produite dans l'année, et le pourcentage devrait croître de

18% par an. Ainsi, les fournisseurs d'infrastructure doivent prendre des mesures pour réduire la consommation d'énergie. Le but est non seulement de réduire le coût de l'énergie
dans les Datacenter, mais aussi pour répondre aux réglementations gouvernementales et aux normes environnementales.

Ordonnancement : L'ordonnancement est un enjeu important qui influence considérablement les performances de l'environnement de Cloud Computing. Il existe plusieurs niveaux d'ordonnancement dans le Cloud, notamment l'ordonnancement au niveau application et l'ordonnancement au niveau infrastructure. Le premier consiste en l'ordonnancement (affectation) des tâches qui constituent les applications des utilisateurs sur les services IaaS ou HaaS du Cloud, tandis que le deuxième niveau concerne l'affectation de machines virtuelles sur les infrastructures physiques (machines physiques) du Cloud. Les deux niveaux d'ordonnancement sont des problèmes complexes.

1.9 Conclusion

Au cours de ce premier chapitre, nous avons fourni une base théorique sur le Cloud Computing, en présentant ses caractéristiques, ses types, ses modèles de services (IaaS, PaaS,SaaS), ses avantages, ses inconvénients et ses challenges, afin d'appliquer ces concepts à notre contexte.

Dans le chapitre qui suit, nous présenterons les stratégies de placement des machines virtuelles dans le Cloud Computing.

CHAPITRE 2

PROBLEME DU PLACEMENT DES MACHINES VIRTUELLES DANS LE CLOUD

2.1 Introduction

La résolution de différents types de problèmes rencontrés dans la vie quotidienne a incité les chercheurs à proposer des méthodes de résolution et à faire des efforts importants pour améliorer leurs performances en termes de temps de calcul et de qualité de la solution proposée. Au fil des années, de nombreuses méthodes de résolution de problèmes de différents niveaux de complexité ont été proposées. Le domaine d'optimisation combinatoire est essentiel, car il se situe au carrefour de la recherche opérationnelle, mathématique et informatique. La grande difficulté que posent les problèmes d'optimisation explique leur importance. L'optimisation combinatoire consiste à parcourir un espace de recherche afin d'extraire la meilleure solution parmi un ensemble d'options.

Le placement de la machine virtuelle est une tentative de déterminer l'hôte le plus approprié pour la machine virtuelle, ainsi, qu'il s'agisse d'un placement initial de VM, une méthode de placement de VM cherche à trouver le mappage VM à PM le plus optimal. Dans un environnement de Cloud Computing, la sélection d'un bon hôte est essentielle pour augmenter l'efficacité énergétique, l'utilisation des ressources et la qualité de service (QoS).

Dans ce chapitre nous allons présenter le problème de placement statique et dynamique de machines virtuelles dans le Cloud Computing, les stratégies de placement : méthodes exactes et méthodes approchées, les objectifs d'une solution de placement et on conclure avec les indicateurs de performance pour un algorithme de placement de VMs.

2.2 Paramètres et Considérations

Pour décider où et quand déployer des ressources dans un environnement Cloud, il faut tenir compte de nombreux paramètres et aspects. Les points suivants sont les plus importants [16] :

- **Performance** : Pour gérer un grand nombre d'applications fonctionnant en même temps, les centres de données ont de plus en plus recours à la virtualisation et à la consolidation. Les différents algorithmes de planification des machines virtuelles se traduisent par des performances très différentes. Lorsque de nombreux fournisseurs de Cloud sont impliqués, les performances deviennent beaucoup plus problématiques.

- **Efficacité énergétique** : Avec l'intérêt croissant des technologies d'écoefficacité, l'efficacité totale en termes d'utilisation, de puissance et de coût est devenue un problème majeur.

  Cet objectif est en contradiction avec d'autres préoccupations, telles que la performance.

- **Coûts** : Au début, les prix fixes dominaient le modèle de tarification, mais les tendances actuelles révèlent que les systèmes de tarification dynamiques deviennent plus populaires. Les coûts implicites internes pour l'ordonnancement des VM, tels que les interférences et les frais généraux causés par une VM sur d'autres VM sur le même hôte, doivent également être pris en compte.

- **Localité**: Pour des raisons d'utilité et d'accessibilité, les machines virtuelles doivent être positionnées à proximité des utilisateurs. D'un autre côté, la localisation peut devenir une contrainte pour une programmation optimale en raison de certains facteurs tels que les questions juridiques et les problèmes de sécurité. Cela s'applique à la fois aux fournisseurs de services qui utilisent les ressources de plusieurs fournisseurs de Clouds et aux fournisseurs de Clouds dont les centres de données sont répartis dans le monde entier.

- **Fiabilité et disponibilité** : L'un des principaux objectifs de la planification des VM est de garantir la fiabilité et la disponibilité du service. Pour ce faire, les VM sont dupliquées sur plusieurs sites. Des facteurs tels que la pertinence du service enveloppé dans les VM, sa fréquence d'utilisation prévue et la résilience du centre de données doivent tous être pris en compte.

2.3 Placement de machines virtuelles dans le Cloud Computing

Il existe deux types de placement de machines virtuelles dans les machines physiques [17] :

- Placement statique (initial).
- Placement dynamique.

2.3.1 Placement statique

Le placement statique des machines virtuelles est effectué lorsque le système est en mode hors ligne ou lors du démarrage du système. Il s'agit du placement initial des VM dans l'environnement du Cloud Computing. Aucune correspondance préalable des machines virtuelles n'est trouvée. Ce type de placement de machine virtuelle ne prend pas en compte les états des VM, les états des hôtes et le taux de requêtes de VM.

La figure 2.1 représente le modèle centralisé de placement statique des machines virtuelle suivant l'approche à deux niveaux.

D'abord le placement des VM sur les centres de données, puis le placement des requêtes sur les hôtes dans un centre de données particulier.

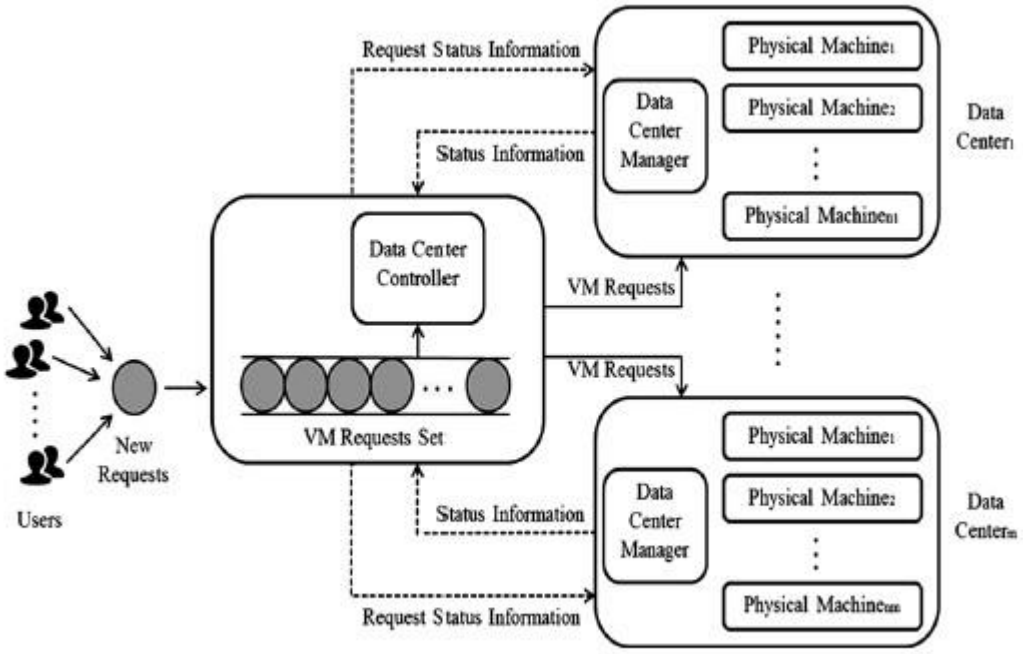


Figure 2.1 - Placement statique de machines virtuelles sur les centres de donné.

Les clients Cloud font des demandes et utilisent les services fournis par le fournisseur de services Cloud. Étant donné que chaque requête sera adressée à une machine virtuelle au-dessus de la machine réelle, ces requêtes se présentent sous la forme de requêtes de machine virtuelle. Chaque centre de données contient plusieurs machines physiques ainsi qu'un gestionnaire de centre de données qui supervise tous les hôtes. Un contrôleur de centre de données est utilisé pour contrôler tous les centres de données.

Les requêtes VM définies sont reçues par le contrôleur du centre de données. Il demande ensuite à tous les gestionnaires de centres de données disposant d'informations provenant de leurs propres centres de données des informations d'état. Les requêtes de VM contiennent des informations sur les ressources (CPU, RAM, réseau, etc.) nécessaires à l'exécution de la demande.

Le contrôleur reçoit des informations des gestionnaires de centre de données concernant les ressources disponibles. Le contrôleur du centre de données distribue de manière optimale les requêtes VM aux gestionnaires du centre de données suivant un algorithme heuristique. Cette approche de distribution des demandes de machines virtuelles est une approche centralisée, où la décision de planifier les demandes dépend du contrôleur central.

La figure 2.2 illustre le placement statique des demandes de machines virtuelles dans les machines physiques d'un seul centre de données.

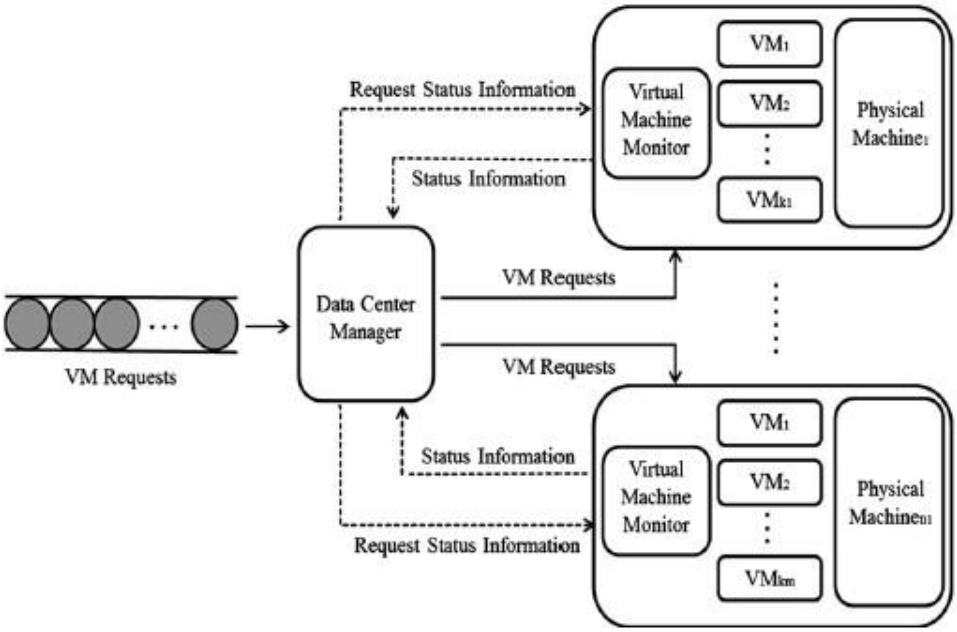


Figure 2.2- Placement statique des machines virtuelles sur des machines physiques.

Le Virtual Machine Monitor [VMM] (hyperviseur) est installé sur de nombreuses machines réelles dans un centre de données. La virtualisation PM est gérée par le VMM. Le gestionnaire du centre de données envoie des demandes aux VMM pour qu'ils fournissent des informations sur l'état de toutes les machines physiques. Le gestionnaire du centre de données dispose déjà des informations relatives aux demandes des VM. A l'arrivée des informations d'état des VMMs, le gestionnaire du centre de données place les demandes de VM sur les machines physiques individuelles afin de les traiter et renvoie la réponse aux utilisateurs du cloud. Le placement des demandes de VM sur les machines physiques est également une approche centralisée car la décision est prise par une autorité centrale, en l'occurrence le gestionnaire de centre de données.

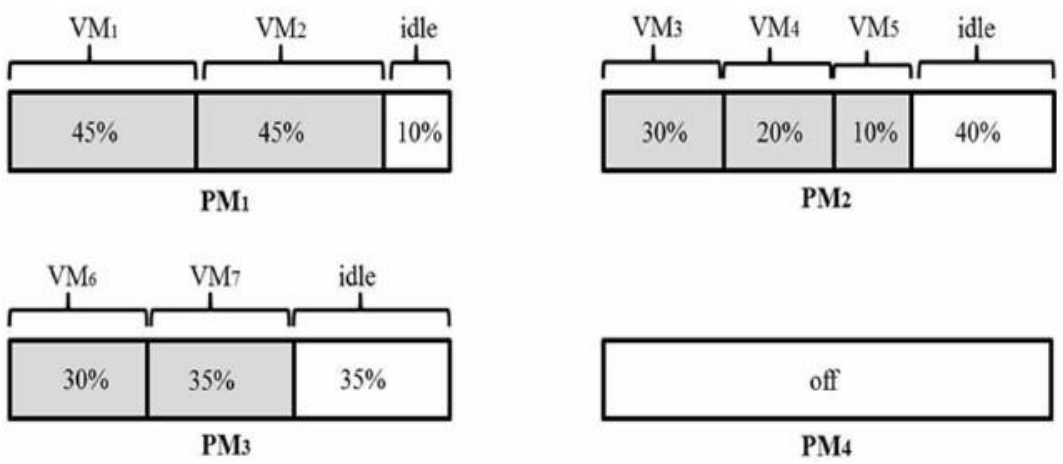
2.3.2 Placement dynamique

Le placement dynamique des machines virtuelles est effectué lorsque le système est en mode en ligne.

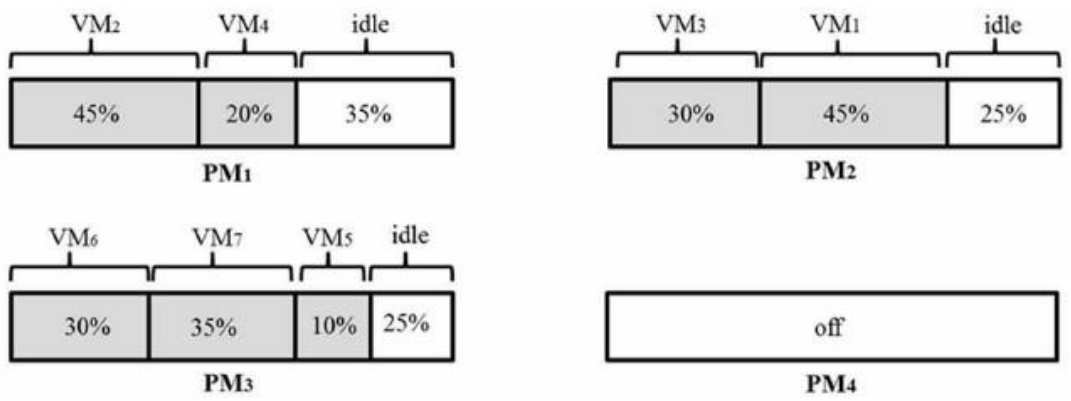
Le placement dynamique des machines virtuelles les place en fonction d'un mappage existant, ce qui contraste avec le placement statique des machines virtuelles qui commence sans mise en correspondance.

Le positionnement dynamique des machines virtuelles a pour objectif d'obtenir des solutions optimales à partir du mappage existant à un coût minimum. Cela ne causera ni n'arrêtera une machine virtuelle déjà en cours d'exécution. La solution de placement doit donc fournir une liste des migrations en direct à exécuter afin d'obtenir le meilleur état possible à partir de l'état existant. Tout le processus est en contraste avec le placement statique des machines virtuelles où les machines virtuelles peuvent être arrêtés et redémarrés, ce qui augmente la consommation d'énergie et réduit les performances du système complet. Pour effectuer un placement dynamique des machines virtuelles, les états des machines physiques devraient également être pris en compte.

La figure 2.3 montre un exemple de placement dynamique.



a) Solution Initial i.



b) Solution finale s1.

Figure 2.3-Placement dynamique de machine virtuelle.

La figure 2.3-a montre la solution initiale (i) pour le problème de placement dynamique des VM. Sept VM sont placées sur quatre PM. Le PM4 est en état d'arrêt car aucune VM ne s'exécute. En supposant que tous les PMs ont la même quantité de ressources, les VM doivent être distribuées dynamiquement sur différents PM afin de réduire la consommation d'énergie.

Une telle solution est illustrée à la figure 2.3-b Pour atteindre la solution s1 à partir de (i), des migrations en direct doivent être effectuées. La liste des migrations est la suivante VM5 de PM2 à PM3, migration de VM1 de PM1 à PM2 et enfin migration de VM4 de PM2 à PM1.

2.4 Stratégies de placement des VMs dans les datacenters du Cloud

Les méthodes d'optimisation peuvent être divisées en deux catégories pour résoudre les problèmes :

- Les méthodes exactes.
- Les méthodes approchées.

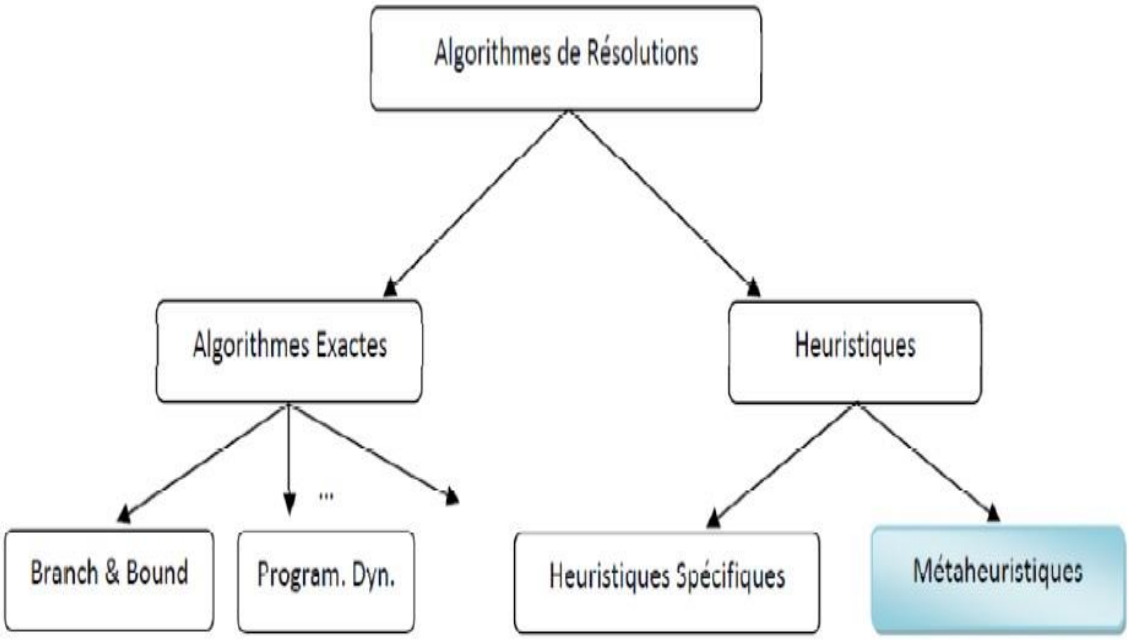


Figure 2.4- Classes des méthodes de résolutions [18].

2.4.1 Les méthodes exactes

Le principe des méthodes exactes est de trouver une solution, la meilleure solution ou l'ensemble des solutions à un problème, ce qui est souvent fait de manière implicite [19].

Ces méthodes fournissent des solutions optimales pour les petits problèmes, mais pour les grands problèmes, le temps de calcul devient irraisonnable [20].

Les méthodes exactes peuvent être divisées en quatre catégories [19] :

- La programmation dynamique.
- La programmation linéaire et non linéaire.
- Les méthodes de recherche arborescente (Branch & Bound).

➢ **La programmation dynamique :**

Est une méthode de résolution de problèmes qui consiste à déterminer la meilleure séquence de décisions à prendre. L'idée de base est qu'en combinant les meilleures solutions d'une série de sous-problèmes, on peut trouver la meilleure solution à un problème. Cela implique de

sélectionner des séquences de décision plus courtes. Les solutions aux problèmes sont calculées dans un ordre ascendant, ce qui signifie que l'on commence par les plus petits sous-problèmes et que l'on remonte vers les plus grands [21].

- **La programmation linéaire :**

La programmation linéaire (PL) est une branche d'optimisation qui peut être utilisée pour résoudre une variété de problèmes combinatoires. La programmation linéaire désigne la manière de résoudre les problèmes où la fonction objective et les contraintes sont toutes linéaires [22].

Si l'ensemble des solutions possibles S, est exprimé sous la forme d'un ensemble de variables ayant des valeurs dans l'ensemble des nombres réels R, qu'il existe des contraintes pour satisfaire les inégalités linéaires et que f est une fonction linéaire dans ces variables, nous avons un problème de programmation linéaire (PL). Plusieurs problèmes réels de recherche opérationnelle peuvent être exprimés comme un problème de PL. C'est pourquoi un grand nombre d'algorithmes permettant de résoudre d'autres problèmes d'optimisation sont basés sur la résolution de problèmes linéaires [19].

- **La méthode de branch and bound :**

Cette méthode consiste à fournir des solutions en explorant un arbre de recherche décrivant toutes les solutions possibles. Le but est de trouver la meilleure configuration afin d'élaguer les branches de l'arbre qui conduisent à de mauvaises solutions [20].

L'algorithme Branch and Bound effectue une recherche complète de l'espace des solutions d'un problème donné, pour trouver la meilleure solution. L'approche de l'algorithme:

1. Diviser l'espace de recherche en sous-espaces.

2. Recherche d'une borne minimale en termes de fonction objectif associée à chaque sous-espace de recherche.

3. Éliminer les mauvais sous-espaces.

4. Répétez les étapes précédentes jusqu'à l'obtention de l'optimum global.

2.4.2 Les méthodes approchées

Les méthodes approchées fournissent une solution approchée au problème posé.

Elles sont généralement conçues de manière à ce que la solution obtenue soit proche de la valeur optimale : de telles méthodes permettent d'obtenir des valeurs inférieures ou supérieures de la valeur optimale par exemple [23]:

- Méthodes Heuristiques.
- Méthodes Méta-heuristiques.

2.4.2.1 Méthodes heuristiques

Une heuristique est un algorithme approché qui fournit rapidement une solution réalisable, sans garanti d'optimalité pour un problème d'optimisation spécifique.

C'est une règle d'estimation, une stratégie, une méthode ou astuce utilisée pour améliorer l'efficacité d'un système qui tente de découvrir les solutions des problèmes complexes [24].

2.4.2.2 Méthodes méta-heuristiques

Une méta-heuristique est un processus itératif qui guide et modifie les opérations des heuristiques subordonnées afin d'obtenir des solutions optimales. Elle peut manipuler une solution unique complète (ou incomplète) ou une population de solutions à chaque itération. Les heuristiques subordonnées peuvent être des procédures de haut (ou bas) niveau, ou une simple recherche locale, ou simplement une recherche constructive [25].

Une méta-heuristique est un ensemble de concepts utilisés pour définir des méthodes heuristiques qui peuvent être appliquées à un large éventail de problèmes différents.

En d'autres termes, une méta-heuristique peut être considérée comme « boîte à outils » algorithmique utilisable pour résoudre différents problèmes d'optimisation, et ne nécessitant que des modifications mineures afin de s'adapter à un problème spécifique [26].

Une méta-heuristique est une méthode permettant de trouver une solution réalisable mais pas nécessairement optimale à un problème NP-Complet en un temps polynomial. Lorsqu'il s'agit du problème de la planification des machines virtuelles dans un centre de données, la détermination de la meilleure combinaison d'emplacements prendrait beaucoup de temps. De plus, en raison de la nature dynamique de l'utilisation des programmes, nous devrons refaire notre travail chaque fois qu'une nouvelle exigence sera ajoutée. Trouver une solution rapide devient alors le seul moyen

d'organiser un bon placement des machines virtuelles, permettant de profiter des services disponibles tout en conservant l'énergie [27].

Les méta-heuristiques peuvent être classifiés en deux catégories celles qui manipulent une population de solutions et celles qui n'utilisent qu'une seule solution [25].

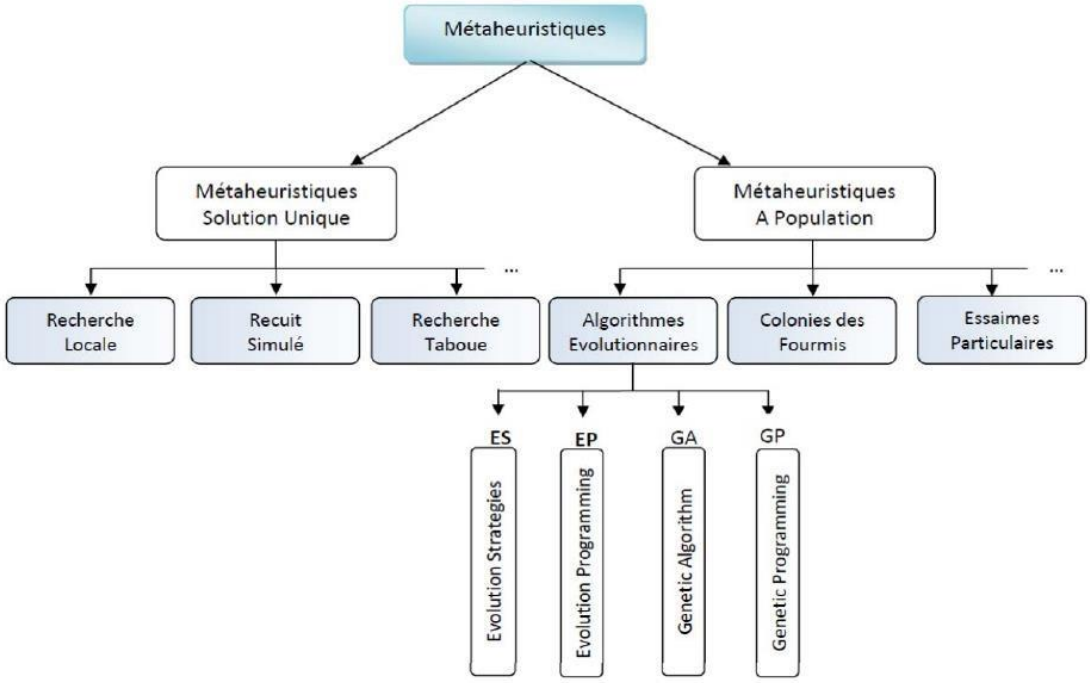


Figure 2.5 - Classes des méta-heuristiques [18].

2.4.2.2.1 Méta-heuristiques à solution unique

Consistent à manipuler et améliorer une seule solution, tant que cela est possible (telles que les algorithmes de recherche locale, de recherche tabou, de recuit simulé, etc…) :

2.4.2.2.1.1 Méthodes de recherche locale

La recherche locale «local search » est la plus ancienne des méthodes de résolution. Elle commence par une solution initiale, et après chaque itération, remplace la solution actuelle par le voisin qui optimise la fonction objectif. La recherche est terminée lorsqu'aucun des voisins

candidats n'est optimal par rapport à la solution courante, ce qui indique que l'optimum local a été trouvé [28].

Le LS peut être représenté comme une descente dans le graphe représentant l'espace de recherche.

D'une manière abstraite, une recherche locale peut être résumée de la manière suivante :

1. Commencer avec une solution.
2. Améliorer la solution.
3. Répétition du processus jusqu'à la satisfaction des critères d'arrêt.

2.4.2.2.1.2 Le recuit simulé

Le recuit simulé est une méta-heuristique inspirée d'une méthode utilisée dans le travail des métaux. S.Kirkpatrick, C.Gelatt, et M.Vecchi l'ont introduite pour la première fois dans [29]. Cette technique alterne des cycles de refroidissement et de réchauffement lents (recuit) afin de minimiser l'énergie du matériau. Son principe de base est que, à partir d'un état de départ, un nouvel état est créé sur la base de l'état précédent en y apportant une modification mineure. Si son énergie est plus faible, il est accepté avec une probabilité plus élevée, sinon il est accepté avec une probabilité plus faible. Cette probabilité est basée sur la température, un paramètre de contrôle critique. La température est élevée au début, l'algorithme accepte plus des mouvais états pour éviter le minimum local. La température baisse progressivement, et l'équilibre est perturbé.

2.4.2.2.1.3 La recherche Taboue

La méthode taboue est une autre technique développée par Glover [30.31].Elle est basée sur le concept de mouvements interdits (ou tabou). Chaque itération consiste à trouver le mouvement qui nous donne la meilleure solution dans le voisinage de la solution courante, tout en gardant à l'esprit que certains mouvements sont interdits. Parfois, on choisit une solution qui détériore légèrement la solution courante afin d'échapper au minimum local. A chaque répétition, l'inverse du mouvement est ajouté à une liste appelée liste taboue, qui contient les mouvements interdits. Initialement, la liste taboue est vide.

2.4.2.2.2 Méta-heuristiques à base de population de solutions :

Consistent à manipuler et améliorer un ensemble de solutions, nommé population, évolue en parallèle (comme les algorithmes évolutionnaires, algorithmes à essaim de particules, etc…) **:**

2.4.2.2.2.1 L'algorithme évolutionnaire

L'algorithme évolutionnaire s'inspire du principe de sélection naturelle de Charles Darwin. La théorie de la sélection naturelle s'applique aux individus d'une population et se repose sur trois principes [32] :

La variation : Les individus sont différents les uns des autres.

L'adaptation : Les individus qui sont mieux adaptés à leur environnement vivent plus longtemps et se reproduisent davantage.

L'héréditaire : fait référence au fait que les caractéristiques des individus sont transmises de génération en génération.

Le principe de la sélection naturelle est le suivant : Les meilleurs individus continueront à évoluer, leurs caractéristiques avantageuses seront transmises aux générations futures et avec le temps, ils deviendront dominants dans la population.

Les algorithmes évolutionnaires fait référence à une série de procédures stochastiques basées sur la simulation de processus d'adaptation naturels et connues sous les noms d'algorithmes génétiques, de stratégies évolutionnaires, de programmation évolutive et de programmation génétique [33].

2.4.2.2.2.2 L'algorithme génétique

L'AG est une technique évolutionnaire conçus et développés par Holland et plus tard par Goldberg et De Jong.

Les GA sont des stratégies de recherche basées sur des mécanismes spécifiques de la génétique et de la sélection naturelle, utilisant trois opérateurs de base : la sélection, le croisement et la mutation.

Pour chaque génération, la sélection est utilisée pour choisir les individus parents, en fonction de leur fonction de fitness. Après avoir sélectionné une paire de chromosomes parents, ils entrent dans l'étape de croisement pour générer deux individus. Le croisement est utile pour créer de nouveaux individus ou de nouvelles solutions qui héritent les bonnes caractéristiques des deux parents. Les individus nouvellement créés seront modifiés par des changements à petite échelle dans les gènes, en appliquant l'opérateur de mutation. Les mutations assurent l'introduction de "nouveauté" dans le matériel génétique. Après avoir complété la population de descendants, ceux-ci remplaceront les parents de la génération précédente et le processus de sélection-croisement-mutation reprendra pour une prochaine génération. Pour éviter de perdre la meilleure solution, [34] a proposé d'appliquer une procédure de remplacement spéciale appelée "élitisme" qui fait une copie du meilleur individu de la population actuelle et le transfère dans la génération suivante [35].

2.4.2.2.2.3 L'optimisation par essaims de particules (PSO)

Développée par Russel Eberhat et James Kennedy aux États-Unis en 1995, est une méthode d'optimisation par essais et erreurs [36]. Cette méthode est inspirée du comportement social d'animaux évoluant en laboratoire. L'exemple le plus fréquemment utilisé est le comportement des bancs de poissons [37]. En effet, on peut observer des dynamiques de mouvement assez sophistiquées chez ces animaux, malgré le fait que chaque individu a une intelligence limitée et une connaissance seulement locale de sa situation dans l'essaim. Un individu à l'essaim ne connaît que la position et la vitesse de ses plus proches voisins. Chaque individu utilise non seulement sa propre mémoire, mais aussi des informations locales sur ses plus proches voisins pour prendre des décisions concernant sa propre vie.

Kennedy et Eberhart se sont inspirés de ces comportements socio-psychologiquespour créer PSO.

2.4.2.2.2.4 L'algorithme de colonies de fourmis

L'algorithme de la colonie de fourmis simule la recherche de nourriture par les fourmis. Ce concept a été présenté pour la première fois par A.Colorni, M.Dorigo, et V.Maniezzo[38]. Une fourmi pose de la phéromone quand il se déplace. Les fourmis sont guidées de manière probabiliste en comptant la quantité de phéromone qui les entoure. Plus la quantité de phéromone indiquant un chemin est importante, plus les fourmis sont susceptibles de le suivre. Comme la phéromone s'évapore avec le temps, la décision probabiliste que les fourmis prend pour choisir un

chemin évolue dans le temps. Les fourmis peuvent trouver collectivement le chemin le plus court entre deux points grâce à l'utilisation d'une phéromone. L'algorithme de la colonie de fourmis est bien adapté à une variété de problèmes d'optimisation.

2.4.2.2.2.5 L'algorithme de la recherche harmonique

L'algorithme de la recherche harmonique a été développé par Geem et, al. en 2001 [39]. Il est basé sur l'approche de la performance musicale consistant à trouver les harmonies parfaites dans un orchestre où chaque musicien joue une note pour obtenir les meilleures harmonies. Dans une analogie inhabituelle, chaque variable de décision dans le processus d'optimisation a une valeur pour trouver la meilleure solution. Lorsqu'un musicien effectue un lancement improvisé, il suit généralement l'une des trois règles suivantes [40] :

- **Règle 1 :** effectuant un lancement de sa mémoire,
- **Règle 2 :** effectuant un lancement adjacent d'un lancement de sa mémoire,
- **Règle 3 :** effectuant le lancement totalement aléatoire de la gamme saine et possible.

2.5 Objectifs d'une solution de placement de VM dans un environnement Cloud

Les objectifs d'une technique idéale de placement des VM sont les suivants [16] :

- Augmenter l'évolutivité du centre de données.
- Maximiser l'utilisation des ressources.
- Réduire la consommation d'énergie.
- Fournir aux clients une qualité de service (QoS) garantie.
- Prévenir la congestion du réseau dans les centres de données.
- Excellentes performances.

- Assurer la sécurité.
- Augmenter le retour sur investissement (ROI).
- Réduire le nombre d'éléments de réseau actifs.
- Minimiser les violations de SLA.
- Diminuer les migrations de VM à l'avenir.

2.6 Indicateurs de performance pour un algorithme de placement de VMs

- **Consommation d'énergie**

Son objectif est d'évaluer la consommation d'énergie du centre de données. La consommation d'énergie augmente en proportion directe de l'utilisation du processeur. Le modèle d'énergie consommé dans un hôte peut être défini par l'équation suivante [16] :

$$\mathbf{P(u) = P_{idle} + (P_{busy} - P_{idle}) * u}$$

La consommation d'énergie prévue pour une utilisation u du CPU est P(u). Lorsque le serveur est inactif, P_{idle} est la quantité d'énergie qu'il utilise. P_{busy} est la quantité d'énergie utilisée lorsque le serveur est entièrement utilisé. Pendant le placement des VM, la consommation d'énergie devrait être moindre.

- **Temps total de migration**

Pendant la migration en direct, la mémoire est copiée.

$$\mathbf{T_{mig} = V_{mig} / b}$$

Le temps total de migration T_{mig} dépend de la quantité de mémoire V_{mig} à transférer de l'hôte vers la cible et de la bande passante du réseau,

b. Pour un bon algorithme de placement des VM, le temps total de migration doit être inférieur.

- **Temps d'arrêt**

Il y a un délai lors de la synchronisation des images du système d'exploitation hôte et cible lorsque la VM démarre sur le nouvel hôte pendant la migration en direct. C'est ce qu'on appelle le temps d'arrêt, et il est défini comme suit :

$$\mathbf{T_{down} = (d * l * t_n)/b}$$

Où d est le taux de salissure des pages,

l est la taille de la page,

t_n est la nième période de pré-copie

et b est la vitesse de la liaison.

Le temps d'arrêt doit être minimal pour un algorithme de placement de VM efficace.

- **Violation du SLA**

est le niveau de service attendu d'un fournisseur de services. La violation du SLA doit être pendant le placement de la VM afin de fournir les services garantis. La violation du SLA (SLAV) est définie comme

$$\mathbf{SLAV = SLATH * PDM}$$

Où SLATH est le nombre de violations du SLA par hôte actif.

$$\mathbf{SLATH} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum \mathbf{T_{si}/ T_{ai}}$$

N est le nombre d'hôtes,

T_{si} est le temps total pendant lequel l'hôte i a eu une utilisation de 100 % du CPU, ce qui a entraîné une violation de l'accord de niveau de service,

et T_{ai} est le temps total pendant lequel l'hôte i a été actif pour fournir des machines virtuelles.

PDM est une dégradation des performances due aux migrations.

$$\mathbf{PDM} = \frac{\mathbf{1}}{\boldsymbol{M}} \sum \mathbf{C}\text{di} / \text{Cri}$$

C_{rij} - capacité totale de l'unité centrale demandée par la VM j pendant sa durée de vie,

M - nombre total de machines virtuelles,

C_{di} - dégradation des performances de la VM j en raison des migrations.

2.7 Conclusion

Trouver un bon placement de machine virtuelle dans un environnement riche en informations est une tâche difficile. Pour la résolution du problème de placement des machines virtuelles, plusieurs approches ont été développées.

Notre objectif était de concevoir une approche heuristique pour résoudre le problème du placement des machines virtuelles, dans le but de réduire les coûts de placement tout en augmentant le nombre de machines virtuelles placées.

CHAPITRE 3

RECUIT SIMULE AMELIORE POUR LE PLACEMENT DES VMS DANS LE CLOUD

3.1 Introduction

Divers algorithmes et approches ont été discutés pour résoudre le problème du placement des machines virtuelles tout en tenant compte de la consommation d'énergie et des performances globales du système dans le domaine du Cloud Computing.

Le problème du placement des VM est un problème NP-Complet, il ne peut pas être résolu en temps polynomial. Pour découvrir la solution optimale dans le domaine réalisable d'un tel problème dans lequel la taille du problème croît et l'espace de solution croît également de manière exponentielle, nous avons utilisé l'approche du recuit simulé (SA).

Le présent chapitre présentera un algorithme du recuit simulé amélioré traitant le problème de placement de machine virtuelle dans le Cloud. Les détails de l'algorithme ainsi que les principaux paramètres de modélisation du problème sont également présentés.

3.2 Le recuit simulé

3.2.1 Origines

La méthode du recuit simulé est une généralisation de la méthode Monte Carlo, dont le but est de trouver la meilleure solution à un problème donné. Elle a été développée par trois chercheurs d'IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi, en 1983, et indépendamment par V. Cerny en 1985 à partir de l'algorithme de Metropolis qui permet de décrire l'évolution d'un système thermodynamique.

La méthode du recuit simulé est basée sur un processus courant dans la métallurgie appelé "le recuit", qui est utilisé pour obtenir un alliage sans défaut.

Pour commencer, on chauffe le métal jusqu'à ce qu'il atteigne une température à laquelle il devient liquide (les atomes peuvent donc circuler librement). Après avoir atteint ce point, on diminue très lentement la température afin d'obtenir un solide. Si la température baisse rapidement, on obtient du verre, si la température baisse lentement (en laissant aux atomes le temps d'atteindre l'équilibre

statistique) (recuit), on obtient des structures plus régulières, jusqu'à atteindre un minimum d'énergie qui correspond à la structure parfaite d'un cristal, on dit alors que le système est «gelé».

Si la baisse de température ne se fait pas assez lentement (trempe), des défauts peuvent apparaître. Il est donc nécessaire de les corriger en réchauffant de nouveau doucement le matériau pour permettre aux atomes de retrouver leur liberté de mouvement, ce qui permet une éventuelle réorganisation conduisant à une structure plus stable [41].

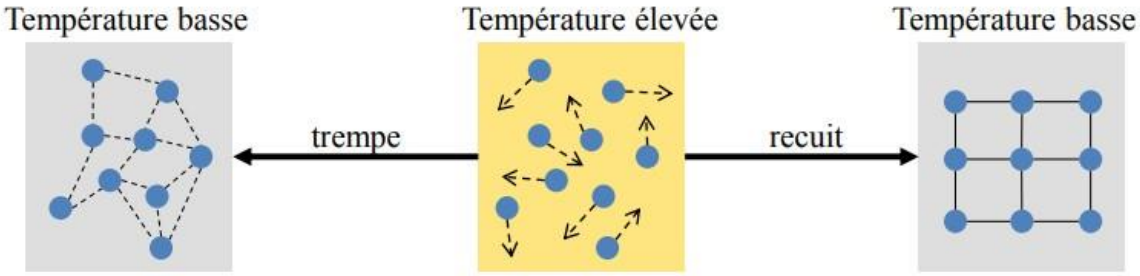


Figure 3.1- Résultat du refroidissement du métal (rapide et lente) [42].

3.2.2 Définition

L'idée principale du recuit simulé tel qu'il a été proposé par Metropolis en 1953, est de simuler le comportement du matériau dans le procédé de recuit largement utilisé en métallurgie. L'objectif est d'atteindre un équilibre thermodynamique, cet équilibre (où l'énergie est minimale) est la meilleure solution à un problème dans la méthode du recuit simulé. L'énergie du système sera calculée à l'aide d'une fonction de coût (ou fonction objectif) qui est unique à chaque problème. La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objective, pour cela, un paramètre fictif de température a été ajouté par Kirkpatrick, Gelatt et Vecchi.

En général, le principe consiste à générer des configurations consécutives à partir d'une solution initiale S0 et d'une température initiale T0 qui diminue tout au long du processus jusqu'à ce qu'une température finale ou un équilibre soit atteint (optimum global) [41].

3.2.3 Algorithme de Metropolis

Dans l'algorithme de Metropolis, on part d'une configuration donnée, et on lui fait subir une modification aléatoire. Si cette modification fait diminuer la fonction objectif (ou énergie du système), elle est immédiatement acceptée, dans le cas contraire elle n'est acceptée qu'avec une probabilité égale à ($\Delta E/T$) (avec E=énergie, et T=température),

ΔE = Ek+1- Ek : C'est la différence entre l'énergie précédente et l'énergie actuelle.

Cette règle est appelée critère de Metropolis [43].

3.2.4 Algorithme du recuit simulé

Le recuit simulé applique itérativement l'algorithme de Metropolis, pour engendrer une séquence de configurations qui tendent vers l'équilibre thermodynamique [41] :

Algorithme 1 : Recuit simulé

---

1. Engendrer une configuration initiale faisable S_0 de S, S ← S_0
2. Initialiser la température T en fonction du schéma de refroidissement
3. Engendrer aléatoirement une configuration voisine faisable S' de S
4. Calculer ΔE = f(S') − f(S)
5. Si ΔE ≤ 0 alors S ← S'
6. Sinon accepter S' comme la nouvelle solution avec la probabilité P(E, T) = $exp^{-\Delta E /T}$
7. Fin si
8. Décrémenter la température (réduire la température)
9. Si condition d'arrêt n'est pas satisfait aller à l'étape 3

10. Retourner la meilleure configuration trouvée

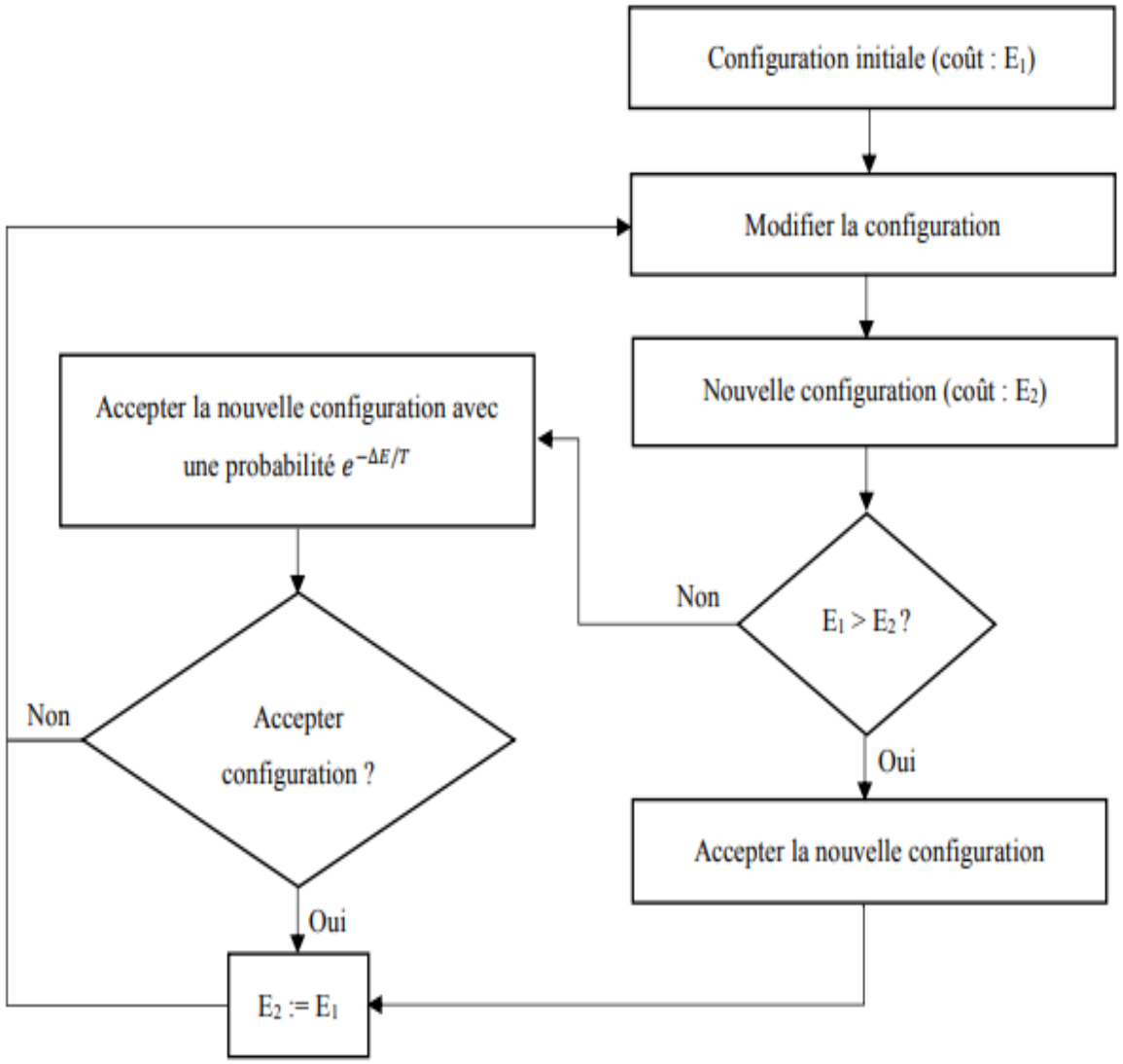


Figure 3.2 – Fonctionnement de l'algorithme du Recuit Simulé.

Au début, comme T est généralement fixé à une valeur élevée, de nombreuses solutions, y compris celles qui dégradent la valeur de f sont acceptées, et la méthode équivaut à une marche aléatoire dans l'espace des solutions. Cependant, lorsque la température baisse, la plupart des solutions d'économie d'énergie sont rejetées, et le système passe par défaut à une amélioration itérative classique.

À une température intermédiaire, l'algorithme effectue des modifications qui dégradent la fonction objective au fil du temps. Il laisse ainsi une chance au système de s'extraire d'un minima local.

La solution initiale peut être trouvée par hasard dans l'espace des solutions possibles, ou elle peut être générée à l'aide d'une heuristique classique. La température initiale doit être assez élevée, car c'est elle qui détermine l'acceptation ou le rejet des solutions défavorables à l'optimisation de la fonction f.

Décroissance de température :

Plusieurs lois de décroissance peuvent être utilisées, la plus courante est la suivante : $T_{i+1} = \alpha.T_i$ / $\alpha < 1$ (en général α=0.9 à 0.99). Le paramètre α est à choisir avec précaution ; En effet, s'il est choisi trop grand, la température baissera très rapidement et l'algorithme pourra être bloqué dans un minima local, en revanche s'il est choisi trop petit, la température baissera très lentement et le temps de calcul sera très grand [44].

Il existe deux approches pour décroitre la température :

· **Décroissance par paliers :** Pour chaque valeur de température, on itère l'algorithme jusqu'à l'atteinte d'un équilibre, puis on diminue la température. On parle alors de paliers de température.

· **Décroissance continue :** La température baisse de façon continue.

Dans notre implémentation, on a utilisé pour décroitre la température l'approche de la décroissance par palier (chapitre 4).

3.2.5 Avantages et Inconvénients du recuit simulé

3.2.5.1 Avantages

- Le principal avantage du Recuit simulé est donc de pouvoir sortir d'un minimum local, en fonction d'une probabilité d'acceptation liée à une fonction exponentielle, connue sous le nom de Gibbs-Boltzmann.
- La méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter.
- En comparaison avec les algorithmes de recherche traditionnels, elle produit généralement de bons résultats.
- Peut être utilisé dans la plupart des problèmes d'optimisation.
- Il converge vers un optimum global (lorsque le nombre d'itérations approche de l'infini), ce qui en fait une option intéressante pour les problèmes d'optimisation difficiles [43].

3.2.5.2 Inconvénients

Le principal inconvénient du recuit simulé est qu'une fois que l'algorithme est réglé sur une température basse à un minimum local, il est impossible d'en sortir. Plusieurs stratégies ont été proposées pour tenter de résoudre ce problème, comme accepter une augmentation rapide de la température dans le temps afin de relancer la recherche sur d'autres régions plus éloignées [43].

En dehors de cela, il y a quelques autres inconvénients à prendre en compte :

- La difficulté de déterminer la température initiale :
  Si elle est trop basse, la qualité de recherche sera mauvaise
  Si elle est trop élevée, le temps de calcul sera plus long
- L'impossibilité de déterminer si la solution trouvée est optimale.

3.2.6 Domaines d'applications

La méthode du recuit simulé, comme toute autre méta-heuristique, peut être utilisée pour résoudre une variété de problèmes d'optimisation [41] :

- Le routage des paquets dans les réseaux.
- Le problème du voyageur de commerce.

- Le problème du sac à dos.
- Le problème de placement de machines virtuelles dans le Cloud [45].

3.3 Le problème de placement de VMs et le recuit simulé

Le problème de placement de VM dans le Cloud Computing est un problème NP-Complet et pour trouver la solution optimale à ce problème, on utilise l'approche de recuit simulé.

3.3.1 Formulation Du Problème

Pour formuler le problème que nous traitons sous la forme mathématique, définissons :

N : nombre de machines virtuelles

M : nombre d'hôtes

V_i: la machine virtuelle numéro i

P_j: l'hôte numéro j

vp_{ij}: la valeur binaire indiquant si la machine virtuelle v_i est affectée au hôte p_j

V : l'ensemble de N machines virtuelles, à savoir v_1, v_2, ..., v_n

P : l'ensemble de M hôtes, à savoir p_1, p_2, ..., p_m

u_j: le pourcentage d'utilisation du CPU de p_j

e_j: la consommation d'énergie (en watt) de p_j.

e_{max}^{j}:la consommation d'énergie (en watt) de p_j lorsque u_j= 100 %.

e_{idle}^{j}:la consommation d'énergie (en watt) de p_j lorsque u_j= 0%.

v_{cpu}^{i}: la demande de CPU de v_i en MIPS.

v_{mem}^{i}: la demande de RAM de v_i en MO.

v_{net}^{i}:la demande de la bande passante du réseau de v_i en Mb/s

p_{cpu}^{i}:la capacité de l'unité centrale de p_j en MIPS.

p_{mem}^{i}:la capacité de la RAM de p_j en MO.

p_{net}^{i}: la capacité de la largeur de bande du réseau de p_j en Mb/s

VP est la matrice binaire de l'affectation de V à P, représentée par :

$$\begin{pmatrix} \text{vp11} & \cdots & \text{vp1M} \\ \vdots & \ddots & \vdots \\ \text{vpN1} & \cdots & \text{vpNM} \end{pmatrix} \quad (1)$$

Où

$$\text{vp}_{ij} = \begin{cases} 1, & si\ vi\ est\ affecté\ \text{à}\ pj \\ 0, & sinon \end{cases} \qquad 1 \leq \text{i} \leq \text{N}\ ,1 \leq \text{j} \leq \text{M} \qquad (2)$$

Pour une affectation VP donnée, l'utilisation CPU de p_j peut être calculée par

$$\text{u}_\text{j} = \frac{\sum_{i=1}^{N} v_{cpu}^{i} * vpij}{p_{cpu}^{j}} \qquad (3)$$

La consommation d'énergie de p_j est calculée par l'équation suivante [46][47][48] :

$$\text{e}_\text{j} = \begin{cases} 0, & si\ \sum_{i=1}^{N} vp\ \text{ij} = 0 \\ \left(e_{max}^{j} - e_{idle}^{j}\right) * \frac{uj}{100} + e_{idle}^{j}, & sinon \end{cases} \quad (4)$$

Lorsque $\sum_{i=1}^{N} vp_{\text{ij}} = 0$, cela signifie qu'aucune machine virtuelle n'est affectée à p_j, elle peut donc être éteinte et ne consomme pas d'énergie.

L'objectif de la recherche est de trouver une affectation VP qui minimise

$$\sum_{j=1}^{M} e_{\text{j}} \qquad (5)$$

Sous contraintes de placement et de capacités suivantes :

$$\forall i, \sum_{j=1}^{M} vp_{ij} = 1 \qquad (6)$$

$$\forall j, \sum_{i=1}^{N} v_{cpu}^{i} * \ vp_{ij} \leq p_{cpu}^{j} \qquad (7)$$

$$\forall j, \sum_{i=1}^{N} v_{mem}^{i} * vp_{ij} \leq p_{mem}^{j} \qquad (8)$$

$$\forall j, \sum_{i=1}^{N} v_{net}^{i} * vp_{ij} \leq p_{net}^{j} \qquad (9)$$

La contrainte 6 signifie qu'une machine virtuelle ne peut être affectée et ne doit être affectée qu'à une seule hôte, les contraintes 7, 8 et 9 exigent que le total des ressources de CPU, de mémoire ou de réseau affectées aux VMs d'une hôte ne puisse pas dépasser les capacités respectives de l'hôte. Ici, nous supposons implicitement que les ressources globales des hôtes sont suffisantes pour accueillir les machines virtuelles invitées, de sorte que nous pouvons garantir que chaque VM invitée peut avoir un hôte sur lequel fonctionner.

Si nous parcourions toutes les combinaisons de VMs avec hôtes, la complexité serait au moins N^M, il n'est donc pas possible d'utiliser un algorithme optimal exact pour s'attaquer au problème car l'espace de recherche augmente de façon exponentielle lorsque N ou M augmente. Par conséquent, nous allons traiter ce problème de manière heuristique.

3.3.2 Algorithme du Recuit Simulé Amélioré pour le placement des VMs

Afin d'éviter le problème du minimum local, l'algorithme du recuit simulé peut accepter avec une certaine probabilité ($\Delta E/T$) une solution moins performante que les autres solutions déjà générées (critère de Metropolis). Ce qui donne la possibilité de perdre la meilleure solution déjà obtenue surtout lorsque l'algorithme ne converge pas vers la meilleure solution.

Pour cela on a amélioré l'algorithme en sauvegardant, à chaque itération, la meilleure solution obtenue jusqu'à présent. La meilleure solution est obtenue en comparant la solution courante avec la dernière meilleure solution sauvegardée.

Les principales étapes de l'algorithme de placement de VMs par recuit simulé sont assez simples, comme le décrit l'algorithme 2 ci-dessous.

Algorithme 2 : Recuit simulé amélioré pour le placement des VMs (RSA)

Entrée : ensemble de VM, ensemble d'hôtes, demandes de VMs (RAM, BW, CPU)
Sortie : affectation (S) avec énergie minimale

01: Générer aléatoirement une affectation faisable S_0, solution courante $S=S_0$
02: $S_{best} \leftarrow S_0$
03: Initialiser la température
04: TQ température $\geq$ température final
05: Pour i = 1 à longueur_palier
06: Calculer l'énergie de S (E (S))
07: Générer une configuration voisine de S :S´
08: Calculer énergie E (S')
09: Si E (S´) $\leq$ E (S) alors
10: S $\leftarrow$ S´
11: $S_{best} \leftarrow$ S´
12: Sinon
13: X $\leftarrow$random [0,1] uniforme
14: Si X $\leq$ exp (-(E (S´) – E (S))/T)
15: S = S´

16:	Finsi
17:	Finsi
18:	Fin pour (palier)
19:	$T \leftarrow \alpha * T$
20:	Fin TQ
21:	Retourner la meilleure configuration avec énergie minimale : Min (S, S_{best})

Remarque : Faisable = Configuration qui respecte les contraintes 6, 7, 8, 9

3.3.3 Description De L'algorithme

Il y quatre composants dans l'algorithme:

1) la configuration du système (solution initiale, température initiale,..)

2) un générateur de la nouvelle configuration (perturbation de la solution initiale)

3) la fonction objective pour le problème d'optimisation

4) décroissance de la température.

Les quatre composants sont clairement définis pour suivre cette méthodologie :

- **Configuration du système**

La configuration est l'affectation initiale des VMs V aux hôtes P.

En utilisant la contrainte qu'une VM ne peut être assignée qu'à un seul hôte, et pour réduire le nombre de variables et la recherche, nous utilisons une HashMap Liste (structure de données qui permet une association clé – valeur)pour représenter l'affectation.

Une telle solution est représentée par une HashMap Liste ou la clé représente la Vm et la valeur représente l'hôte.

- **Générateur de nouvelles configurations (perturbation de la solution initiale)**

L'objectif de cette étape est la génération d'une nouvelle solution voisine, qui ne fournit pas forcement une amélioration par rapport à la solution précédente. Pour réaliser l'étape 2, trois techniques peuvent être mises en œuvre : inversion, translation et commutation [45].

Inversion : le processus d'inversion génère un nouvel emplacement en échangeant quelques positions de l'existante configuration de placement.

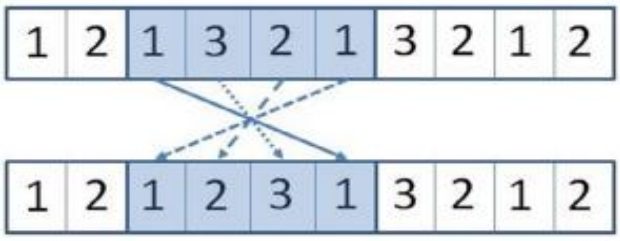


Figure 3.3 – Inversion [49].

Traduction : pour générer une nouvelle configuration, le processus de traduction supprime deux nœuds consécutifs ou plus de la configuration existante et les place entre deux nœuds consécutifs sélectionnés au hasard.

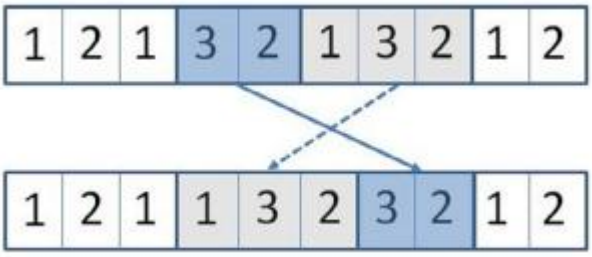


Figure 3.4 –Traduction [49].

Commutation : la technique de commutation sélectionne au hasard deux nœuds de la configuration existante et les permute pour obtenir une nouvelle configuration.

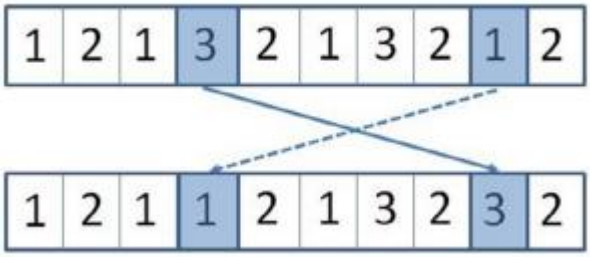


Figure 3.5 –Commutation [49].

Dans notre implémentation de l'algorithme, nous avons utilisé une technique de perturbation à côté de la technique de commutation. La technique de perturbation consiste à choisir au hasard quelques nœuds et de changer leurs numéros d'hôtes parmi l'ensemble des hôtes du Datacenter. Cela permet d'explorer l'espace de recherche d'une façon plus large que les 3 solutions cité précédemment qui sont limité par l'ensemble des hôtes générés initialement.

- **Critères d'acceptation des nouvelles configurations**

Bien que la génération de nouvelles configurations fournisse des configurations candidates, les critères d'acceptation décident quelle configuration devient le nouvel état suivant. Dans l'algorithme proposé, si une nouvelle configuration est faisable, c'est-à-dire qu'elle satisfait toutes les contraintes spécifiées par l'équation (6, 7, 8, 9), et qu'elle a une consommation d'énergie inférieure à celle de l'état précédent, elle sera acceptée comme nouvel état. Pour être simple, pendant le calcul de l'énergie, si une solution est jugée infaisable, la valeur énergétique se verra attribuer un nombre énorme, et dans ce cas elle sera ignorée et on régénère d'autre solutions jusqu'à l'obtention d'une solution faisable.

Dans le cas où une solution voisine n'améliore pas la fonction objective (énergie consommée), celle-ci pourra être acceptée selon le critère de Metropolis.

- **Décroissance de la température**

La décroissance de la température est un aspect très important de la méthodologie de recuit simulé, sinon la trempe rapide peut conduire à des résultats loin d'être optimaux.

Combien d'itérations sont autorisées à chaque température (longueur du palier) et la plage de température (température initiale, température finale et taux de décroissance), par exemple, sont les questions de l'ordonnancement de la température.

Cependant, la conception de décroissance de température n'est pas détaillée dans la méthodologie SA, mais expliquée de manière empirique.

3.4 Conclusion

Ce chapitre a été consacré à la présentation de notre approche «recuit simulé» pour la résolution du problème de placement des machines virtuelles dans un environnement de Cloud Computing. Comme nous l'avant déjà ce problème appartient à la classe NP difficiles, donc sa résolution n'est possible que d'une manière approchée.

Dans le chapitre qui suit, nous présenterons notre implémentation et les résultats obtenus.

CHAPITRE 4

IMPLEMENTATION ET RESULTATS EXPERIMENTAUX

4.1 Introduction

Pour bien finir notre travail, nous avons réalisé une simulation CloudSim Plus de notre projet. Les résultats de la simulation seront détaillés à la fin de ce chapitre.

Nous commençons ce chapitre par présenter les outils de simulations utilisées, à savoir, le langage java, l'IDE NetBeans et le simulateur CloudSim Plus. Nous présentons également l'IHM que nous avons développée ainsi que les résultats obtenus de simulation.

4.2 Langage et environnement

4.2.1 Le langage de programmation Java

Java est un langage de programmation à usage général, évolué et orienté objet, avec une syntaxe similaire à celle du C. Ses caractéristiques, ainsi que la diversité de son écosystème et de sa communauté, lui ont permis d'être largement utilisé dans le développement d'applications de types très disparates. Java est très populaire dans le développement d'applications d'entreprise et mobiles [50].

Les caractéristiques :

- Java est interprétable et portable : il fonctionne sur n'importe quelle plateforme.
- Java est un langage de programmation orienté objet.
- Java est simple, fortement typé.
- Java assure la gestion de la mémoire, sécurité, économe et multitâche.

Sun/Oracle met gratuitement à disposition un ensemble d'outils et d'API pour aider les programmeurs à créer des applications Java. Ce kit, connu sous le nom de JDK, peut être téléchargé gratuitement sur le site Web de Sun à l'adresse http://java.sun.com ou sur celui d'Oracle à l'adresse http://www.oracle.com/technetwork/java.

Le JRE (Java Runtime Environment) contient uniquement l'environnement d'exécution de programmes Java. Le JDK lui-même contient le JRE. Seul le JRE doit être installé sur les machines sur lesquelles les applications Java seront exécutées.

Sun a défini trois plateformes d'exécution (ou éditions) pour java pour des cibles distinctes selon les besoins des applications à développer :

-Java Standard Edition (J2SE / Java SE) : environnement d'exécution et ensemble complet d'API pour des applications de type desktop. Cette plate-forme sert de base en tout ou partie aux autres plateformes.

- Java Enterprise Edition (J2EE / Java EE) : un environnement de développement d'applications d'entreprises entièrement basé sur Java SE.

-Java Micro Edition (J2ME / Java ME) : un environnement d'exécution et une API pour le développement d'applications mobiles et embarquées dont les capacités ne permettent pas la mise en œuvre de Java SE.

La séparation en trois plateformes permet au développeur de mieux cibler l'environnement d'exécution et de faire évoluer les plateformes de façon plus indépendante.

Il existe de nombreuses et diverses sortes d'applications Java qui peuvent être développées :

- Applications desktop
- Applications Web (servlets/JSP, portlets, applets)
- Applications pour appareils mobiles (MIDP) : midlets
- Applications pour appareils embarqués (CDC) : Xlets
- Applications pour carte à puce (Javacard) : applets Javacard
- Applications temps réel

Figure 4.1- Logo de langage Java.

4.2.2 Environnement de développement NetBeans

NetBeans est un environnement de développement intégré (EDI) publié en open source par Sun en juin 2000. En plus de Java, NetBeans prend en charge une variété d'autres langages, notamment Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il possède toutes les fonctionnalités d'un IDE moderne (éditeur de couleurs, éditeur graphique d'interfaces utilisateur et de pages web, etc.).

NetBeans est une application conçu en Java qui s'exécute sur Windows, Linux, Mac OS X et d'autres systèmes d'exploitation, ainsi qu'une version indépendante (qui nécessite une machine virtuelle Java).

Un environnement Java Development Kit (JDK) est requis pour les développements en Java. NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)) et s'enrichit à l'aide de plugins [51].

4.2.3 Simulateur CloudSim Plus

CloudSim Plus est un cadre de simulation Java pour la modélisation et la simulation de divers services de Cloud Computing, y compris les couches d'infrastructure en tant que service (IaaS) et de logiciel en tant que service(SaaS). Il permet la création de scénarios de simulation pour tester, évaluer et valider des algorithmes à des fins diverses. Le cadre permet aux développeurs de spécifier les caractéristiques des différentes entités des fournisseurs de Cloud, notamment [52] :

- Les ressources physiques telles que les centres de données, les machines physiques (hôtes, serveurs ou simplement PM) et les réseau actifs ,

- Les ressources logiques telles que des réseaux de stockage (SAN), des topologies de réseau et des applications ,

-La couche de virtualisation, qui fournit des éléments tels que des machines virtuelles (VM) pour permettre la virtualisation des ressources physiques et logiques, et

- La gestion des exigences.

Il automatise la gestion de toutes ces ressources qui sont fournies en tant que service, en agissant comme des moniteurs de machines virtuelles (hyperviseurs ou simplement VMMs) qui effectuent des tâches d'administration de bas niveau telles que :

-La gestion du cycle de vie des VM (comme la création, le démarrage, l'arrêt, la destruction, le placement et la migration),

-La gestion des machines physiques actives pour réaliser des économies d'énergie,

-La planification de l'exécution des VM à l'intérieur des MP et de l'exécution des applications à l'intérieur des VM,

- L'allocation des VM.

4.2.3.1 Architecture générale de CloudSim Plus

CloudSim Plus est composé de différents modules.

Le module principal qui représente l'API du cadre de simulation est l'API CloudSim Plus. C'est le seul module nécessaire à la mise en place d'expériences de simulation de nuages.

La figure 4.2 montre une représentation simplifiée de la structure du paquet.

Les fonctionnalités de CloudSim Plus ne sont disponibles que dans les packages avec une teinte plus claire. Ces classes plus légères ont été établies uniquement pour améliorer l'organisation et la séparation des préoccupations (SoC), mais les classes contenues ont été refactorisées et repensées en profondeur.

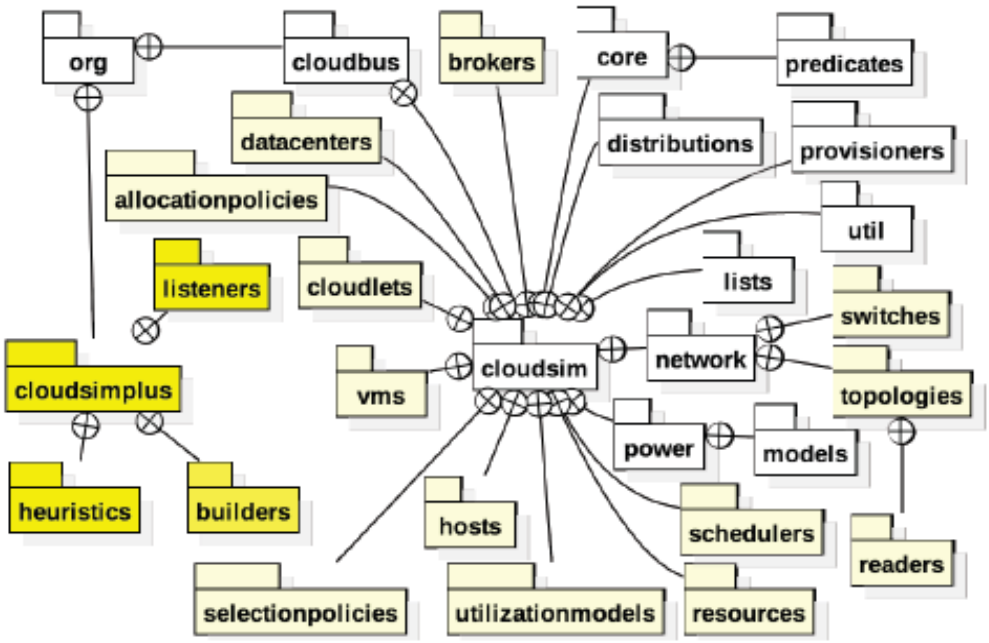


Figure 4.2- Structure du package de l'API CloudSim Plus [50].

Voici les packages CloudSim Plus les plus importants et les classes qu'ils contiennent :

• **distributions :** classes qui permettent à l'API de simulation de générer des nombres pseudo-aléatoires basés sur une variété de distributions statistiques. Ils peuvent également être utilisés par les développeurs qui créent leurs propres simulations.

• **réseau "network" :** classes pour permettre des simulations de réseau en établissant l'architecture réseau du centre de données,

• **puissance "power" :** classes pour permettre des simulations tenant compte de la consommation d'énergie en incluant des modèles de consommation d'énergie qui peuvent être améliorés par les développeurs créant leurs simulations.

Le projet CloudSim d'origine comprend des classes qui offrent des fonctionnalités de base pour implémenter rapidement des situations de simulation, telles que des modèles d'utilisation des ressources pour la RAM, le processeur et la bande passante. Il est livré avec un ensemble de classes de base et quelques interfaces que les utilisateurs doivent améliorer afin d'ajouter des fonctionnalités uniques à leurs simulations. Les packages légèrement indiqués dans la figure 1 ont été introduits pour améliorer l'organisation des classes et des interfaces dans CloudSim Plus, en regroupant les classes et

les interfaces avec des objectifs spécifiés dans des packages bien définis. Voici les nouveaux packages :

- **hosts, datacenters, virtual machines, and Cloudlets :** une collection de classes qui permettent des implémentations multiples de Datacenters, Hosts, Vms, et Cloudlets (applications), y compris des composants sensibles à la puissance et au réseau.

- **allocationPolicies:** classes qui permettent à un centre de données de choisir un hôte sur lequel installer ou migrer une machine virtuelle. Le cadre comprend une politique de VmAllocationPolicySimple qui place une Vm donnée sur l'hôte ayant le moins de cœurs de processeur disponibles.

- **brokers:** classes qui agissent au nom d'un client Cloud, répondant aux demandes de création et de destruction de Cloudlets et de Vms, et attribuant des Vms spécifiques à ces applications. Ces courtiers peuvent utiliser des algorithmes de prise de décision pour hiérarchiser les demandes d'applications Cloud, déterminer comment une Vm est choisie pour exécuter une application spécifique, etc. Il comporte un DatacenterBroker exclusif qui mappe les Cloudlets aux Vms à l'aide d'un algorithme.

- **schedulers :** classes permettant de planifier l'exécution de nombreux Cloudlets au sein d'une Vm, ainsi que de nombreuses Vms au sein d'un Host. Il dispose de planificateurs partagés dans le temps et dans l'espace, ainsi que d'une implémentation unique du planificateur complètement équitable vu dans les dernières versions du noyau Linux.
- **ressources :** classes qui représentent les ressources physiques et logiques du Cloud telles que les disques durs, les cœurs de processeur (traitement éléments ou simplement Pes), RAM, bande passante et fichiers utilisateur.
- **utilizationmodels :** classes qui modélisent l'utilisation de ressources telles que CPU, RAM et bande passante, définissant comment une ressource donnée est utilisée par une application au fil du temps.

Les packages exclusifs de CloudSim Plus sont mis en évidence par une couleur plus vive et décrits brièvement ci-dessous :

- **listeners:** Il s'agit de classes qui reçoivent des notifications pendant l'exécution de la simulation et qui surveillent ensuite le scénario de simulation.

Il est envisageable, par exemple, d'attribuer des Vms à la demande lorsqu'une condition donnée est remplie en utilisant de telles fonctionnalités de surveillance.

- **heuristics :** classes qui offrent la base pour l'implémentation d'heuristiques pour diverses raisons telles que le mappage Cloudlet vers Vm et les décisions de migration Vm .

- **builders :** classes qui implémentent le modèle de conception Builderqui fonctionnent comme des usines d'objets pour créer divers objets de simulation tels que Hosts, Vms et Cloudlets.

4.2.3.2 Les classes de CloudSim Plus

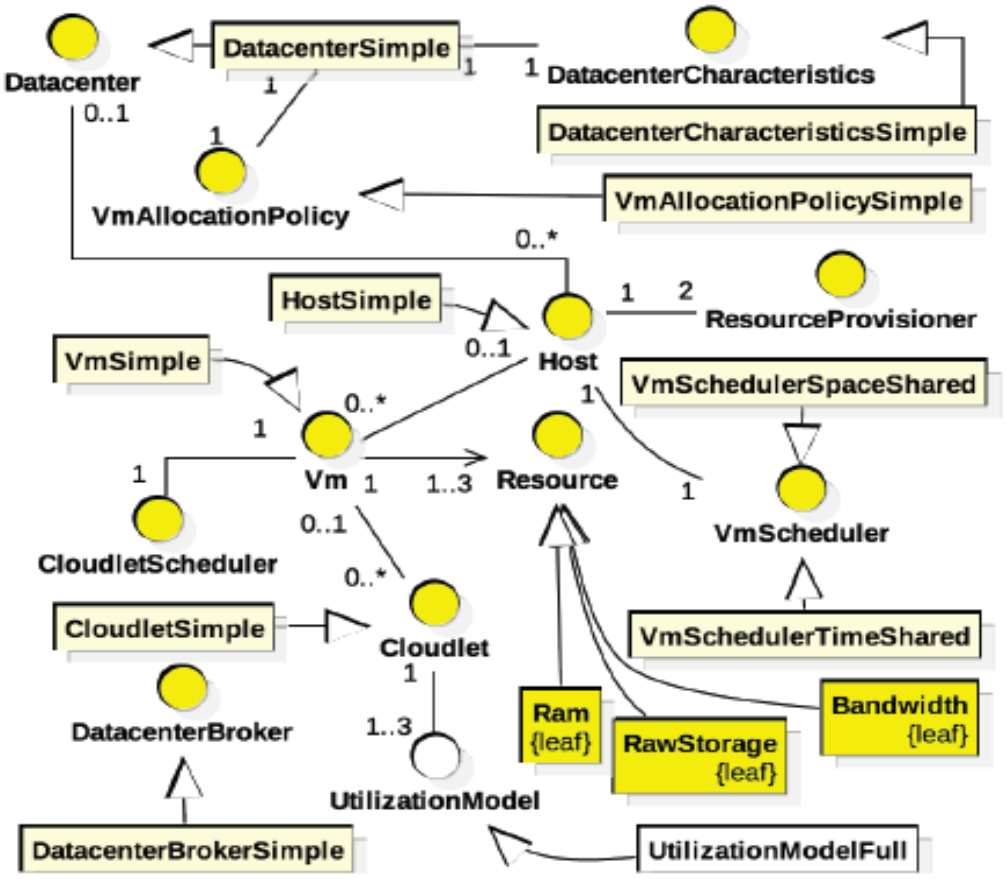


Figure 4.3 - Classes principales impliquées dans la création de simulations CloudSim Plus [50].

Les nouvelles classes, mises en évidence par une couleur plus foncée, indiquent la bande passante, le stockage et les ressources RAM d'un hôte ou d'une machine virtuelle. Toutes ces ressources ont des caractéristiques qui doivent être contrôlées, telles que la capacité et l'utilisation actuelle. En conséquence, l'interface des classes de ressources et de mise en œuvre ont été introduites pour supprimer tout code dupliqué utilisé pour gérer ces ressources.

La suppression du code dupliqué et l'inclusion de nouveaux cas de test garantissent également que les fonctionnalités fonctionnent de manière cohérente et comme prévu.

Vous trouverez ci-dessous une liste des classes principales qui entrent dans la réalisation d'une simulation.

- **DataCenter, DatacenterCharacteristics et VmAllocationPolicy** : un centre de données est un ensemble de machines physiques (hôtes, serveurs ou simplement PM) qui fonctionnent ensemble pour fournir la base de l'architecture Cloud. Chaque Datacenter a des caractéristiques qui le définissent, telles que les prix associés aux différentes ressources physiques de ses Hosts. Un objet DatacenterCharacteristics définit ces propriétés. Une instance VmAllocationPolicy doit être établie pour chaque centre de données créé. Cet objet détermine quel PM hébergera quelle machine virtuelle. L'implémentation de VmAllocationPolicySimple dans le cadre est une stratégie la moins adaptée qui alloue des Vms à l'hôte avec les cœurs de traitement (Pes) les plus disponibles.

- **Host, Pe et VmScheduler :** un hôte représente une machine physique (PM), et chaque PM nécessite la définition d'une liste d'éléments de traitement (Pes) (les cœurs CPU de la machine). Étant donné que le PM peut héberger des machines virtuelles, un mécanisme d'ordonnancement est nécessaire pour gérer l'exécution simultanée de plusieurs machines virtuelles dans le Pes hôte. Les VmSchedulers peuvent être utilisés de différentes manières, y compris en temps et en espace partagés.

- **DatacenterBroker** : Ce logiciel agit au nom d'un client Cloud, en recevant des demandes et en prenant les mesures nécessaires pour y répondre. Ces opérations comprennent la soumission de Vms à allouer au sein de l'hôte d'un centre de données, la soumission de Cloudlets (applications) à exécuter dans certains des Vms nouvellement créés, le choix de la VM sur laquelle déployer un Cloudlet spécifique, etc.

- **Vm et CloudletScheduler :** un objet Vm représente une machine virtuelle qui fonctionne dans un hôte et exécute des applications (Cloudlets). Un CloudletScheduler spécifie comment plusieurs applications sont programmées pour s'exécuter simultanément dans une machine virtuelle. Il suit la même logique que VmScheduler, avec les mêmes implémentations de base et un ordonnanceur complètement équitable.

- **Cloudlet et UtilizationModel :** un Cloudlet représente une application qui s'exécutera à l'intérieur d'une Vm, de manière abstraite défini en fonction de ses caractéristiques, telles que le nombre demillions d'instructions à exécuter, le nombre de Pes requis et les modèles d'utilisation du CPU, de la RAM et de la bande passante. Chaque objet UtilizationModel définit la manière dont une ressource donnée sera utilisée par le Cloudlet au fil du temps. Certaines implémentations de base de UtilizationModel sont fournies, telles que l'UtilizationModelFull, qui indique qu'une ressource disponible sera utilisée à 100% tout le temps.

4.3 Interface de l’application

Interface Principale

Le lancement de l'application fait apparaitre la page suivante :

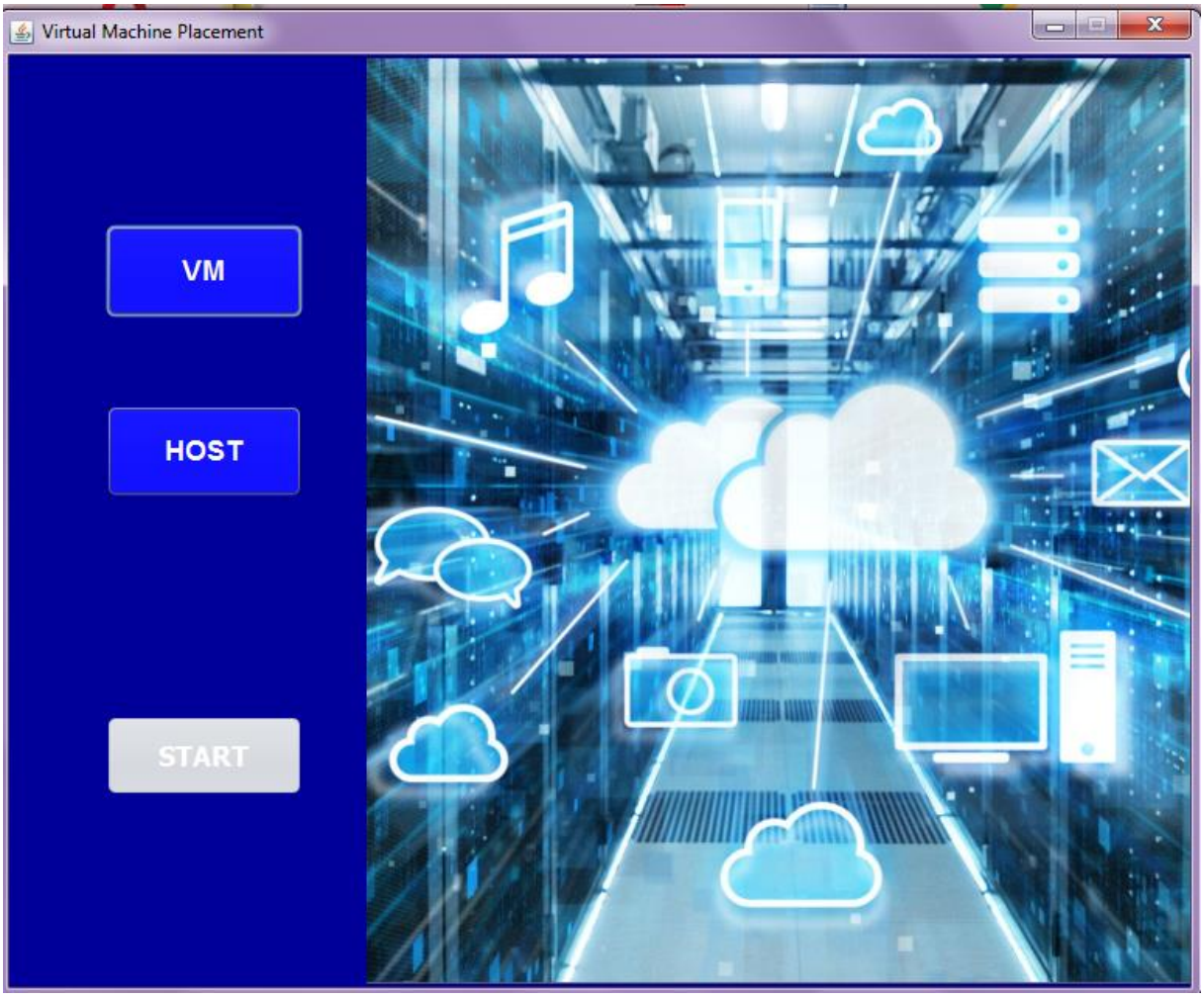


Figure 4.4 – Interface principale.

La première fenêtre sur la figure 4.4 comprend 3 boutons, les boutons VM et HOST qui vont lancer la configuration des VMs et Hôtes. En cliquant sur les deux boutons HOST et VM, les fenêtres figurant sur les images ci-dessous s'affichent. À la fin on devrait appuyer sur le bouton Start pour lancer l'exécution de notre application.

Configuration des VMs

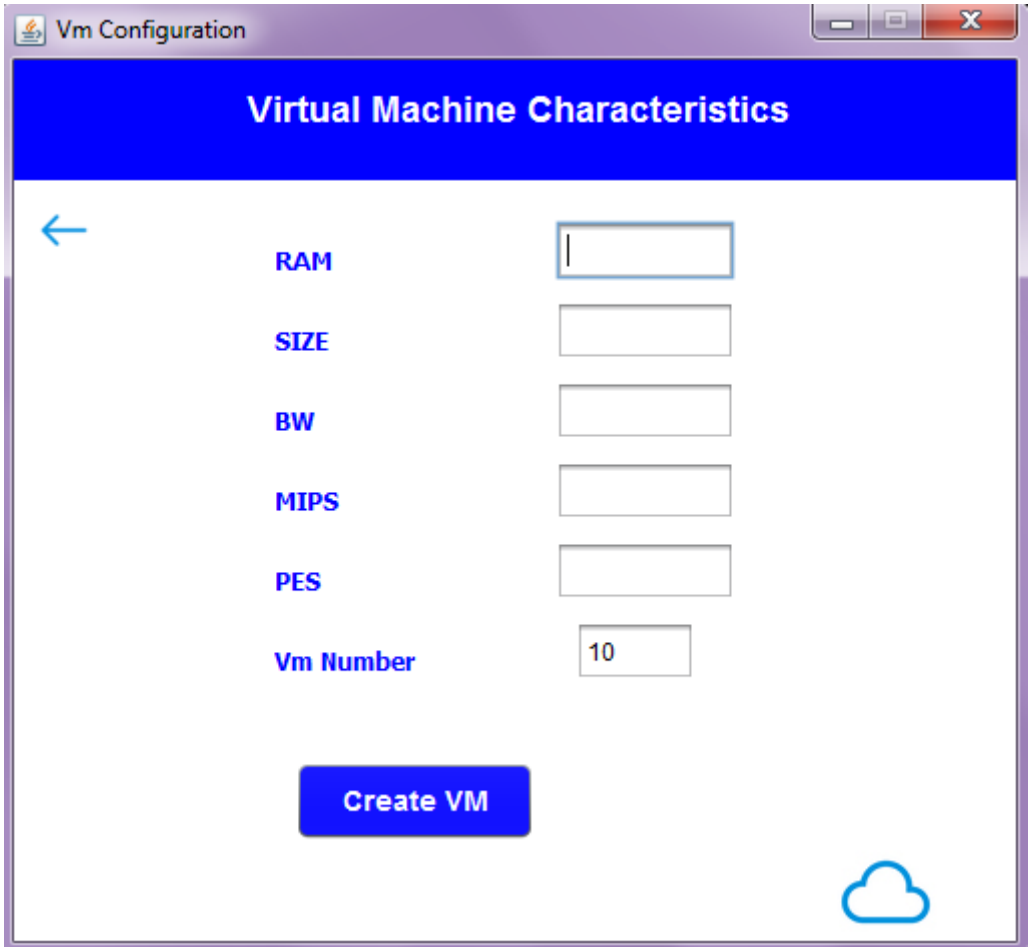


Figure 4.5 – Création des machines virtuelles.

Pour configurer les VMs, nous commençons par saisir les paramètres nécessaires à la création des machines virtuelles de telle sorte que le VM soit obligatoirement dans un hôte, un hôte pouvant contenir une ou plusieurs machines virtuelles. L'utilisateur doit saisir les informations suivantes : La rame (RAM), le Storage (size), bande width (BW), la vitesse de chaque processeur (MIPS), le nombre de processeur (PES) et le nombre de VM à placer. Après avoir saisir les données, il faut appuyer sur le bouton Create VM pour créer les VMs.

Si vous voulez revenir à la page principale, une flèche retour se trouve en haut à gauche de la fenêtre.

Configuration des Hôtes

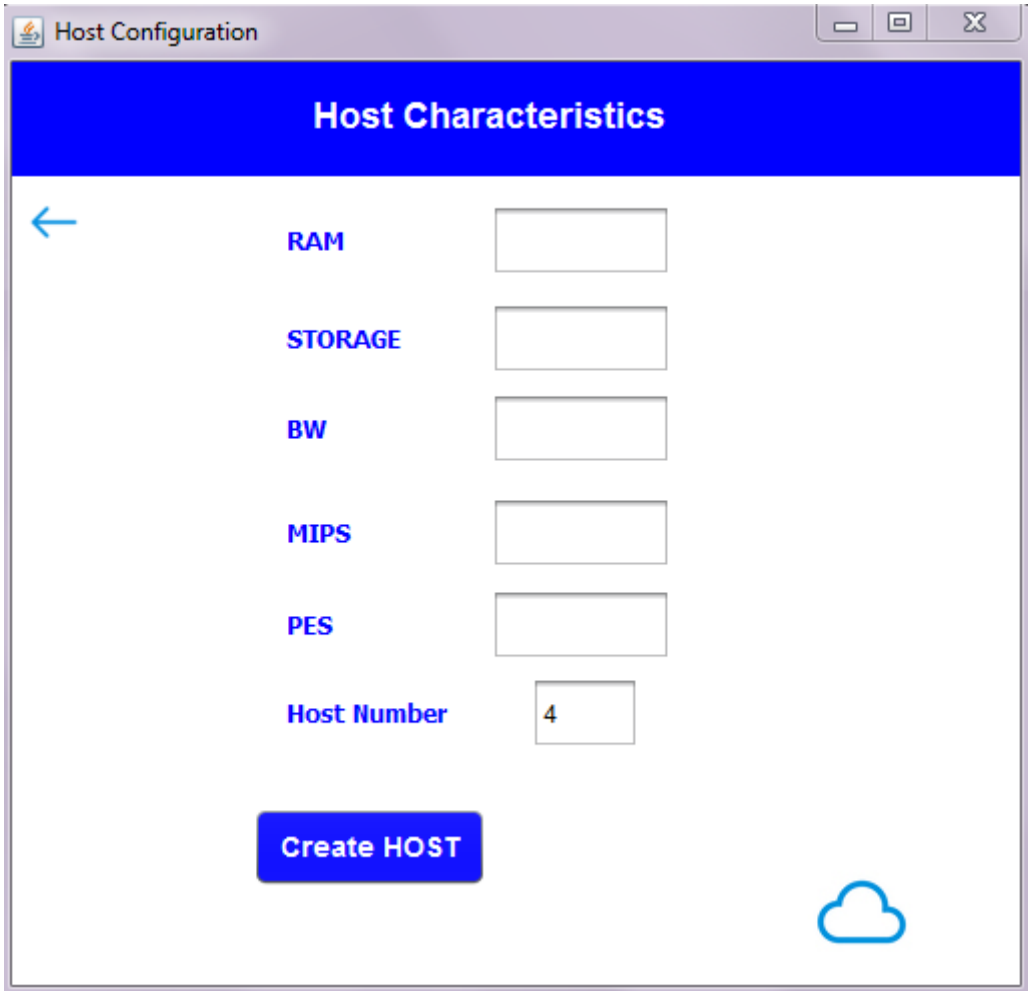


Figure 4.6 – Création des hôtes.

Lorsque l'étape de création de VM est achevée, on entame l'introduction des paramètres nécessaires pour créer les hôtes, il s'agit des mêmes caractéristiques citées ci-dessus qui sont : La rame (RAM), le Storage (size), bande width (BW), la vitesse de chaque processeur (MIPS), le nombre de processeur (PES) et le nombre d'hôtes. À la fin de la saisie des données on doit appuyer sur le bouton Create HOST pour créer les hôtes.

Si vous voulez revenir à la page principale, une flèche retour se trouve en haut à gauche de la fenêtre.

Une fois les phases de création des VM et Hôtes sont terminées, on clique sur le bouton Start pour que l'exécution se lance.

4.4 Résultats expérimentaux

Pour évaluer la technique proposée on a effectué des simulations sur une machine avec un processeur Intel Core-3 de fréquence 1.8 GHZ et avec une Ram de 4 Go.

Dans la simulation, on a commencé par des petits nombre de VM et on a ensuite augmenté ce nombre.

Les résultats de simulation de la technique du recuit simulé amélioré (RSA) sont comparées avec deux méthodes reconnais dans la littérature à savoir le Placement Aléatoire (PA), le placement selon la politique Round-Robin (placement PRR).

Cette simulation est mesurée par la consommation d'énergie, et réalisée avec les paramètres suivants : La ram (RAM), le Storage (size), bande width (BW), la vitesse de chaque processeur (MIPS), le nombre de processeur (PES) et le nombre de VM et Hôtes.

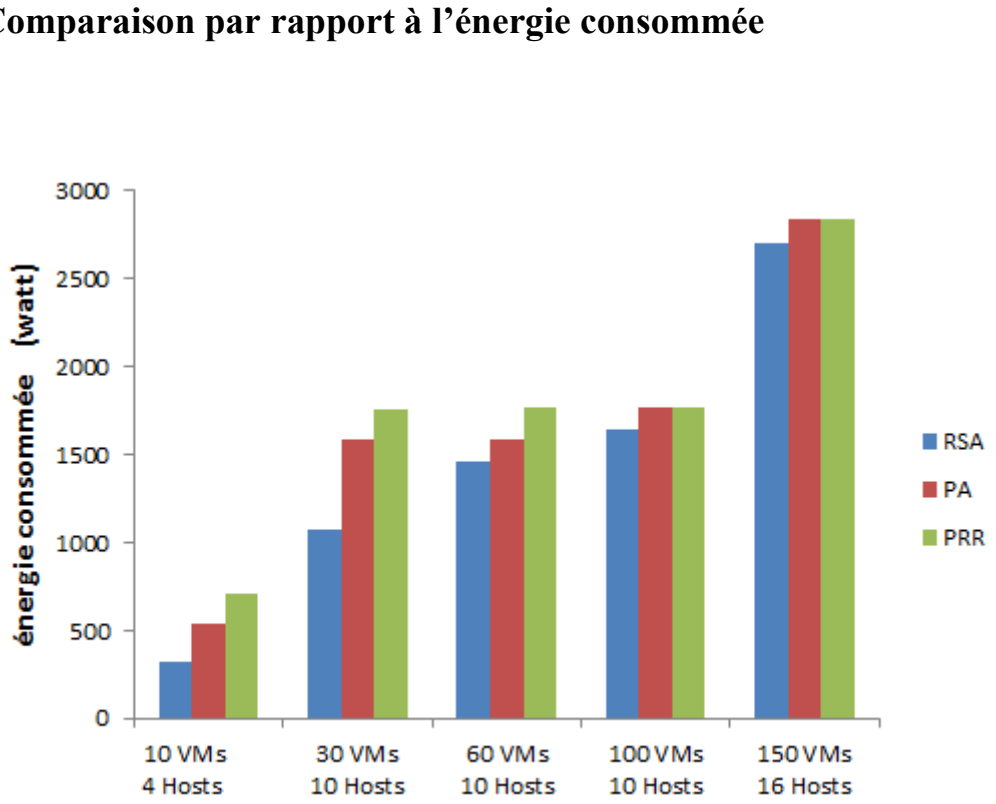


Figure 4.7 - Comparaison énergétique entre le RSA, PA et PRR.

La figure 4.7 Montre le résultat de la consommation d'énergie de notre algorithme proposé Recuit Simulé Amélioré avec les méthodes PA et PRR.

La barre à gauche illustrée sur la figure 4.7 représente la quantité totale d'énergie consommée par le système.

Comme le montre la figure 4.7, l'énergie consommée par le système augmente de façon linéaire pour tous les algorithmes lorsque le nombre de machines virtuelles augmente.

Dans notre comparaison, on fait varier les VMs entre 10 et 150, et également les hôtes.

D'après nos résultats, le recuit simulé est beaucoup plus économe en énergie que le placement aléatoire et le placement round-robin.

Le résultat de la simulation révèle que notre algorithme proposé Recuit Simulé a montré de meilleures performances par rapport aux autres méthodes discutées, en termes de consommation d'énergie.

Le recuit simulé donne de bons résultats par rapport aux méthodes PA et PRR.

Le recuit simulé nous a permis d'atteindre notre objectif principal qui est la diminution de la consommation d'énergie.

4.5 Conclusion

Au cours de cette dernière étape de notre travail, nous avons présenté l'implémentation ainsi que les outils qui ont été utilisés pour résoudre le problème de placement de VM dans un PM en utilisant l'algorithme de recuit simulé tout en tenant compte de divers objectifs et contraintes.

Conclusion générale :

Le Cloud Computing est une nouvelle technologie populaire dans le monde d'aujourd'hui science de l'informatique. Elle permet de fournir de multiples types de services aux clients en fonction de leurs besoins en termes de qualité de service.

Dans ce mémoire, nous avons proposé un algorithme pour résoudre le problème de placement de machines virtuelles dans les centres de données du Cloud Computing et appliqué une approche d'optimisation heuristique pour obtenir une solution adaptée.

Pour implémenter l'approche proposée, nous avons utilisé le simulateur CloudSim Plus qui a été choisi pour ses qualités de modélisation. La contrainte utilisée pour évaluer les performances de notre algorithme est la consommation d'énergie.

Après avoir analysé les résultats obtenus après l'exécution de notre programme pour résoudre le problème de placement des machines virtuelles dans les centres de données tout en minimisant la consommation d'énergie, nous avons obtenu des résultats très satisfaisants et logiques qui confirment les contraintes énoncées au préalable ainsi que notre fonction objective.

Bibliographies:

[1]: Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. NIST Cloud Computing reference architecture, volume 500. 2011.

[2]: https://aws.amazon.com/fr/what-is-cloud-computing, consulté le 04/02/2022, à 20 h.

[3]: Djillali Boukhelef, vers une methodologie d'optimisation du placement des objets dbaas dans un environnement de Cloud Computing, thèse Doctorat, 2019.

[4]: Zhang, Q., Cheng, L. & Boutaba, R. CloudComputing: state-of-the-art and research challenges. J Internet Serv Appl, 2010.

[5]: Ku Ruhana Ku-Mahamud, and Husna Jamal Abdul Nasir, Ant colony algorithm for job scheduling in grid computing. In Mathematical/Analytical Modelling and Computer Simulation (AMS), Fourth Asia International Conference on (pp. 40-45), 2010.

[6]:Juan Carlos Cuesta, Karita Luokkanen-Rabetino, and Katarina StanoevskaSlabeva, Grid Value Chains – What is a Grid Solution, 2010.

[7] : Ricardo Sanz, J. Bermejo, and Ignacio Lopez paniagua, A Rationale and Vision for Machine Consciousness in Complex Controllers, 2007.

[8] : Khajehei Kamyab, Role of virtualization in cloud computing, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 4, 2014.

[9] : Dad djouhra, Optimisation des performances des datacenters des Cloud sous contrainte d'énergie consomée, thèse Doctorat, 2016.

[10]: Khoa Dang Pham, Embedded Virtualization of a Hybrid ARM-FPGA Computing Platform, 2014.

[11]: Desai, Ankita, et al. Hypervisor: A survey on concepts and taxonomy. International Journal of Innovative Technology and Exploring Engineering 2.3, 2013.

[12]:https://fr.slideshare.net/Tsubichi/virtualisation-71519850, consulté le 06/02/2022, à 14 h.

[13] : C. d. R. d. e. d. Production, Cloud Computing Définitions et Concepts, Enquête et Analyse des Tendances, France, 2010.

[14] : Khalil Sabine, Implementing Cloud services in large french organizations: beyond heir IT governance, Business administration, Thèse de doctorat. Télécom ParisTech, 2017.

[15]:Sonia Yassa, Multi-constrained optimal allocation of workflows to Cloud Computing resources. Theses, Université de Cergy Pontoise, Juillet 2014.

[16] : Divya Bharathi.P, Prakash.P and Vamsee Krishna Kiran.M, "Virtual Machine Placement Strategies in Cloud Computing", International Conference on Innovations in Power and Advanced Computing Technologies (i-PACT2017).

[17] : Arnab Paul & Bibhudatta Sahoo, Dynamic Virtual Machine Placement in Cloud Computing, Chapter 6, janvier 2017.

[18]: Dr. LEMOUARI ALI, Introduction Aux Méta-heuristiques, cour, Université de Jijel, 2014.

[19]: LAYEB Abdeslam, Utilisation des Approches d'Optimisation Combinatoire pour La Vérification des Applications Temps Réel, Thèse Doctorat en informatique, Université Mentouri de Constantine, 2010.

[20] : Radhia Zaghdoud. Hybridation d'algorithme génétique pour les problèmes des véhicules intelligents autonomes : applications aux infrastructures portuaires de moyenne taille. Automatique. Ecole Centrale de Lille; Institut supérieur de gestion (Tunis), 2015.

[21]: SOPHIE Jacquin, hybridation des métaheuristiques et de la programmation dynamique pour les problèmes d'optimisation mono et multi-objectif : application à laproduction d'énergie, Thèse Doctorat en informatique, Université de Lille 1, 2015.

[22]: Karloff, Howard. *Linear programming*. Springer Science & Business Media, 2008.

[23]: DOUIRI Mohamed, Méthodes de Résolution Exactes Heuristiques et Métaheuristiques, Mémoire Master en informatique, Université d'Oran, 2008.

[24]: Romanycia, Marc HJ, & Francis Jeffry Pelletier. "What is a heuristic?." *Computational intelligence* 1.1 (1985): 47-58.

[25]: S. Voß, S. Martello, I.H. Osman and C. Roucairol (Eds), "MetaHeuristics - Advances and Trends in Local Search Paradigms for Optimization". Kluwer Academic Publishers, Dordrecht, The Netherlands, (1999).

[26]: metaheuristics.org, Consulté le 26/03/2022, à 21h.

[27]: mohamed mourad mamlouk, approche heuristique pour le placement des machines virtuelles dans un environnement infonuagique de grande taille, université de montréal, 2017.

[28]:SERIK mehdi rouan, implémentation de méthodes de recherches locales sur les architectures multi et many-cœurs, Application au problème d'affectation quadratique à 3 dimensions, Thèse Doctorat en informatique, Université d'Oran.

[29]: S.KIRKPATRICK, C.D. GELATT ET M.P. VECCHI. Optimization by Simulated Annealing. s.l. : Science, 1983.

[30]: GLOVER F., "Tabu Search - Part 1", ORSA Journal on Computing, vol. l, pp. 190-206,1989.

[31]: GLOVER F., "Tabu Search - Part II", ORSA Journal on Computing, vol. 2, pp. 4-32,1990.

[32]:CHARLES-EDMOND BICHOT, élaboration d'une nouvelle métaheuristique pour le partitionnement de graphe : la méthode de fusion-fission. Thèse de Doctorat, novembre 2007.

[33]:DEFFAS ZINEB, Métaheuristiques parallèles à solution unique pour la résolution du problème du Q3AP sur grille de calcul. Mémoire Magister en informatique, 2010.

[34]:K.A. De Jong, Genetic Algorithms: A 10 Year Perspective; Proceedings of the 1st InternalConf. onGAs and TheirAppl, pp. 169-177, 1985.

[35] : Mihai Gavrilas, Heuristic and metaheuristic optimization techniques with application to power systems, Proceedings of the 12th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering, Octobre 2010.

[36]: Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimization, Proceedings of the IEEE Conference on Neural Networks, IV, Piscataway, NJ, pp. 1942-1948.

[37]: Reynolds, C.W. (1987). Flocks, herds and schools: a distributed behavioral model, Computer Graphics, Vol. 21, N°4, pp.25-34, 1987.

[38]: A.COLORNI, M.DORIGO andV.MANIEZZO.Distributed Optimization by Ant Colonies.s.l. : Proceedings of the first European Conference on Artificial Life, 1992.

[39]: Z. W. Geem, J. H. Kim & G. V. Loganathan. "A new heuristic optimization algorithm :harmony search". simulation, 2001, vol. 76, n. 2, p. 60-68.

[40]: F.Z. Benayed, L. Abdelhakem-Koridak& M. Rahli. “Optimisation d’un dispatching economique de la production combinee d’electricite et de la chaleur par l’algorithme harmonysearch”. Mediamira Science Publisher, 2011, vol. 52.

[41]: https://rfia2012.files.wordpress.com/2011/12/amine_le_recuit_simulc3a9.pdf, consulté le 05/05/2022, à 9h.

[42]: http://dumas.perso.math.cnrs.fr/MINT-2018-TO6.pdf, consulté le 05/05/2022, à 10h.

[43]: AutinBaptist, Les métaheuristiques en optimisation combinatoire, Mémoire présenté en vue d’obtenir l’examen probatoire en informatique, Conservatoire National des Arts et Métiers, Paris, 2006.

[44]: mohammedbenhammouda, contribution a l’optimisation d’ordonnancement de workflows dans un environnement Cloud, thèse de doctorat en sciences, 2021.

[45]: Dubey, Kalka, SubhashChander Sharma, and Aida A. Nasr. "A SimulatedAnnealingbasedEnergy-Efficient VM Placement Policy in Cloud Computing." *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*.IEEE, 2020.

[46] : Young-Choon Lee&Albert Zomaya, Energy efficient utilization of resources in Cloud computingsystems, The Journal of Supercomputing, May 2010.

[47] :Rawas, Soha, Ahmed SherifZekri, and Ali El Zaart. "Power and Cost-aware Virtual Machine Placement in Geo-distributed Data Centers." CLOSER. 2018.

[48] :GAO, Yongqiang, GUAN, Haibing, QI, Zhengwei, et al, A multi-objective antcolony system algorithm for virtual machine placement in cloudcomputing, Journal of computer and system sciences, 2013, vol. 79, no 8, p. 1230-1242.

[49] : SouravKantiAddya, AshokKumarTuruk, BibhudattaSahoo, MahaswetaSarkar, Sanjay KumarBiswash, Simulatedannealingbased VM placement strategy to maximize the profit for Cloud Service Providers, Engineering Science and Technology, an International Journal 20 (2017) 1249-1259.

[50] : Jean Michel DOUDOUX, Développons en Java ,2011.

[51] : BECHAR Anissa, SADELLI Salim. Conception et Implémentation d'une Application Mobile pour les Services d'Aide aux Etudiants sous Android. Master en informatique. Université A. Mira-Bejaïa 2013.

[52] : Silva Filho, Manoel C., et al. "CloudSim Plus: A CloudComputing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness." 2017 IFIP/IEEE symposium on integrated network and service management (IM).IEEE, 2017.