République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 20 Août 1955 – Skikda
Faculté des Sciences
Departement d'informatique

جـامعة 20 أوت 1955- سكيكدة
كـــــــلية العلـــــوم
قسم الإعـــــلام الآلـــي

Mémoire De fin d'étude en vue de l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème :

Optimisation multi-objectif dans les centre de données Cloud sous contrainte d'énergie consommée

Réalisé par :

- Bouslama Belkis
- Djilani Manel

Dirigé par:

Mr. BOUAITA Riad
Mr. LAOUAR Walid

Année Universitaire 2022-2023

Remerciement

En premier lieu, nous remercions الله عزو جل de nous avoir donné la force et la patience nécessaire pour achever ce mémoire.

Nous tenons à remercier notre Encadrant Monsieur BOUAITA Riad, enseignant à l'Ecole Normale Supérieure de L'Enseignement Technologique Skikda -AZZABA pour son aide à réaliser notre mémoire ainsi que pour sa disponibilité et son soutien.

Nous remercions également Monsieur LAOUAR Walid, enseignant à l'université de SKIKDA, d'avoir Co-encadré notre travail, de son suivi et ses conseils.

Nous tenons également à adresser nos remerciements aux membres du jury de nous avoir fait l'honneur d'accepter de participer à notre jury de mémoire.

Nous tenons aussi à saluer toute notre promotion de Master 2 Réseaux et systèmes distribués et tous nos amis.

Enfin, que tous ceux qui ont participé de près ou de loin à la réalisation de ce travail, trouvent ici le témoignage de notre profonde reconnaissance.

Dédicaces

Du profond de mon cœur, je dédie ce travail a tous ceux qui me sont chers,

À Mon Cher papa (B.B)

Ce travail est dédié à mon père, qui il est décédé رحمك الله*, qui m'a toujours poussé et motivé dans mes études.*

J'espère que, du monde qui est sien maintenant, il apprécie cet humble geste comme preuve de reconnaissance de la part d'une fille a qui a toujours prié pour le salut de son âme. Puisse Dieu, le tout puissant, l'avoir en sa sainte misécorde !

À Ma chère Mère

Aucune dédicace ne saurait exprime mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et J'espère que votre bénédiction m'accompagne toujours.

À Ma chère sœurs

Merci énormément pour ton soutien plus que précieux. Merci pour ton grand cœur toutes vos qualités qui seraient trop longues à énumérer. Ma vie ne serait pas aussi magique sans ta présence et ton amour.

Belkis

Dédicaces

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que je dédie cette mémoire à :

À ma charmante mère :

Tu représente pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

À Mon très cher Père :

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail et le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

Mes chères sœurs et Mon très cher et unique frère :

Je leur souhaite du succès dans leur vie personnelle et académique.

A mes grands-parents paternels et maternels :

Qui sont décédés, en particulier ma grand-mère **« MAMA YAMINA** », qui est décédée récemment, que Dieu ait pitié de vous رحمكم الله

À Ma famille et Mes chers amis :

Pour tous les souvenirs éternels dans nos cœurs, je les remercie pour et pour leur soutien et pour m'accompagner.

A tous ceux dont l'oubli du nom n'est guère celui du cœur…

Manel

Résumé

Le cloud computing est un processus révolutionnaire qui a impacté sur la façon d'utiliser les réseaux. Il fournit aux clients des services et des ressources sur une base de paiement à l'utilisation. Les machines virtuelles (VM) exécutent de manière élastique les charges de travail sur les machines physiques (PM) dans les centres de données pour une plus grande flexibilité. Ces centres de données consomment d'énormes quantités d'énergie. Ce travail porte sur la problématique du placement des machines virtuelles dans le cloud. Le problème du placement des VMs étant un problème d'optimisation de complexité NP-complet avec de multiples contraintes, on a proposé une approche basée sur l'algorithme du Recuit Simulé. De plus, sachant qu'un placement optimal de VMs vise à satisfaire multiples objectifs, l'algorithme du recuit simulé a été étendu via l'intégration de l'optimum de Pareto dans lequel on cherche à obtenir un compromis entre différents objectifs contradictoires. A cet effet, deux objectifs sont mis en place dans ce travail : la consommation totale d'énergie et la violation des accords de niveaux de service (*Service Level Agreement* - *SLA*). La stratégie proposée a été implémentée sur le simulateur CloudSim Plus, l'un des simulateurs cloud les plus populaires. Les résultats de simulation montrent l'efficacité de l'algorithme utilisé notamment via l'obtention des solutions non dominées pour différents nombres de machines virtuelles.

Mots Clé : Cloud Computing, centre de données, placement de Machines Virtuelles, optimisation multi-objectif, Recuit simulé, optimum de Pareto, efficacité énergétique, contrat de niveau de service.

Abstract

Cloud computing is a revolutionary process that has impacted the way networks are used. It provides customers with services and resources on a pay-as-you-go basis. Virtual machines (VMs) elastically run workloads on physical machines (PMs) in data centers for high flexibility. These data centers consume huge amounts of energy. This work deals with the problem of virtual machine placement in the cloud. The VMs placement problem being an NP-complete complexity optimization problem with multiple constraints, we proposed an approach based on the Simulated Annealing algorithm. Moreover, knowing that an optimal placement of VMs aims to satisfy multiple objectives, the simulated annealing algorithm has been extended through the integration of the Pareto optimum in which we seek to obtain a compromise between different contradictory objectives. To this end, two objectives are implemented in this work: the total energy consumption and the violation of service level agreements (SLA). The proposed strategy was implemented on the CloudSim Plus simulator, one of the most popular cloud simulators. The simulation results show the efficiency of the algorithm used, in particular by obtaining non-dominated solutions for different numbers of virtual machines.

Keywords: Cloud Computing, Data Center, Virtual Machine Placement, Multi-Objective Optimization, Simulated Annealing, Pareto Optimum, Energy Efficiency, Service Level Agreement.

ملخص

الحوسبة السحابية هي عملية ثورية أثرت على طريقة استخدام الشبكات. هذه التقنية تزود العملاء بالخدمات والموارد على أساس الدفع حسب الاستخدام. تعمل الأجهزة الافتراضية (VMs) بمرونة على تشغيل أحمال العمل على الأجهزة المادية (PMs) في مراكز البيانات للحصول على قدر أكبر من المرونة. تستهلك مراكز البيانات هذه كميات هائلة من الطاقة. يتعامل هذا العمل مع مشكلة وضع الأجهزة الافتراضية في السحابة. نظرًا لكون مشكلة وضع VMs مشكلة تحسين ذات تعقيد كامل مع قيود متعددة ، فقد اقترحنا نهجًا يعتمد على خوارزمية محاكاة التلدين. علاوة على ذلك ، مع العلم أن الموضع الأمثل لأجهزة VM يهدف إلى تحقيق أهداف متعددة ، فقد تم توسيع خوارزمية التلدين المحاكاة من خلال دمج Pareto الأمثل الذي نسعى فيه للحصول على حل وسط بين الأهداف المتناقضة المختلفة. تحقيقا لهذه الغاية ، تم تحديد هدفين في هذا العمل: إجمالي استهلاك الطاقة وانتهاك اتفاقيات مستوى الخدمة (SLA). تم تنفيذ الإستراتيجية المقترحة على محاكي CloudSim Plus ، أحد أشهر محاكيات السحابة. تظهر نتائج المحاكاة كفاءة الخوارزمية المستخدمة ، لا سيما من خلال الحصول على حلول غير خاضعة للسيطرة لأعداد مختلفة من الأجهزة الافتراضية.

الكلمات الرئيسية: الحوسبة السحابية ، مركز البيانات ، وضع الآلة الافتراضية ، التحسين متعدد الأهداف ، التلدين المحاكي ، باريتو الأمثل ، كفاءة الطاقة ، اتفاقية مستوى الخدمة.

Table des matières

Contents

Liste des Figures

Chapitre 04

Liste des tableaux

Liste des abréviations

NIST	National Institute of Standards and Technologies
CISCO	Computer Information System Company
CIGREF	Club Informatique Des Grandes Entreprises Françaises
PDA	Personal Digital Assistant
SaaS	Software as a Service
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SLA	Service Level Agreement
QoS	Quality of Service
VPN	Virtual Private Network
VM	Virtual Machine
PM	Pysical Machine
KVM	Kilobyte Virtual Machine
VXLAN	Virtual Extended Local Area Network
VLAN	Virtual local area network
NVGRE	Network Virtualization using Generic Routing Encapsulation
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
IPC	Inter-process communication
OS	Operating System
CPU	Central Processing Unit

RAM	Random Access Memory
BW	BandWidth
E/S	Entrée /Sortie
PC	Personal Computer
NP-complets	Non déterministe Polynomial
HaaS	Hardware-as-a-Service
EPA	Engineering Personnel Authorization
DVFS	Dynamic voltage and frequency scaling
PL	Programmation Linéaire
AG	Algorithmes Génétiques
PSO	Particle Swarm Optimization
SLATAH	Service Level Agreement violation Time per Active Host
PDM	Performance Degradation Due to Migration
IDE	Integrated Development Environment
IHM	Interface Homme Machine
JDK	Java Development Kit
API	Application Programming Interface
JSP	Java server Page
JSTL	JavaServer Pages Standard Tag Library
JSF	Java Server Faces
XML	Extensible Markup Language
Mosa	Multi objective Simulated Annealing
PHP	Hypertext Preprocessor

Introduction générale

Au fil du temps, de plus en plus de nouvelles technologies sont introduites dans notre vie quotidienne sans les sentir venir et nous en prenons conscience en si peu de temps. Il en va de même pour le cloud computing, l'une des technologies les plus puissantes de ces dernières décennies.

L'une des technique clés est la virtualisation qui permet à une seule machine physique de prendre en charge plusieurs machines virtuelles, chacune pouvant exécuter différents types de services et d'applications qui représentent les besoins en ressources des utilisateurs.

Le placement optimal des machines virtuelles au sein des machines physiques garantit des performances élevées en termes d'utilisation des ressources et des applications, de sécurité, d'efficacité énergétique et de qualité de service (QoS) fiable, comme défini par les accords de niveau de service (SLA).

Les centres de données qui contiennent plusieurs éléments tels que des serveurs, des équipements de réseau et des systèmes de refroidissement consomment de grandes quantités d'énergie pour fournir des services efficaces et fiables à leurs utilisateurs.

Dans cette étude, nous proposons une approche polyvalente pour le placement des machines virtuelles Où nous avons utilisé deux critères contradictoires (énergie, SLA) en essayant de minimiser la consommation d'énergie et SLA qui il s'agit du pourcentage de temps moyen pendant lequel les hôtes actifs ont atteint 100 % d'utilisation du processeur a l'aide de métaheuristiques. Notre méthode se concentre sur l'algorithme du Récuit simulé multi-objectifs.

Ce mémoire est divisé en quatre chapitres et une conclusion générale comme suit :

Le chapitre 1 : « généralisés sur le cloud computing » nous présentons le cloud computing, ses caractéristiques, ses modèles de déploiements et de services ; nous ferons ensuite le pas vers la virtualisation et quelques notions qui y ont une relation.

Le chapitre 2 : nous avons présenté les problèmes du cloud computing et les différentes méthodes d'optimisation de placement des VMs (Les méthodes exactes, les méthodes

approchées) et nous avons passé sur les Indicateurs de performance et l'optimisation multi objectif d'une façon générale.

Le chapitre 3 : nous avons détaillé l'approche du Récuit simulé multi-objectif pour la résolution du problème de placement des machines virtuelles dans les centres de données du cloud computing.

Le chapitre 4 : nous allons rapporter tous ce que nous avions pu faire au niveau pratique, avec quel simulateur nous avons travaillé, le langage de programmation, et surtout ce que nous avons pu trouver comme algorithmes optimales existants pour la consommation d'énergie dans le Cloud.

Nous terminons ce mémoire par une conclusion générale, où nous résumons ce que nous avons fait.

Chapitre I :
Généralité sur le cloud Computing

1.1 Introduction

L'informatique est encore en train de se transformer, elle se dématérialise et devient cloud computing. La puissance informatique devient ainsi virtuelle et se consomme aux besoins des clients et devient extensible.

Comme le Cloud Computing devient de plus en plus accessible et populaire, une évolution naturelle consiste à étendre les différents concepts de services, plateformes et infrastructure au provisionnement à la demande afin d'assurer une certaine connectivité entre les ressources virtuelles et les services fournis par le Cloud.

Dans ce chapitre, nous allons présenter les notions fondamentales du Cloud Computing. Dans un premier temps nous allons étudier le Cloud de manière générale, en présentant ses définitions, caractéristiques et ces architectures, nous parlerons également de la virtualisation dans le cloud, de son fonctionnement et des domaines de son utilisation, ainsi que quelques notions qui y ont une relation, et nous conclurons par les Allocation de ressources dans le Cloud 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 Technologie), 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.

Selon (CISCO) 2 : Le Cloud Computing est une plateforme de mutualisation informatique fournissant aux entreprises des services à la demande avec l'illusion d'une infinité de ressources. [2]

Selon le groupe de travail (CIGREF)3 : Le Cloud Computing est défini par les quatre points suivants :

- Un Cloud est toujours un espace virtuel.
- Contenants des informations qui sont fragmentées.
- Dont les fragments sont toujours dupliqués et répartis dans cet espace virtuel, lequel peut être sur un ou plusieurs supports physiques.
- Qui possède « une console (programme) de restitution « permettant de reconstituer l'information. [3]

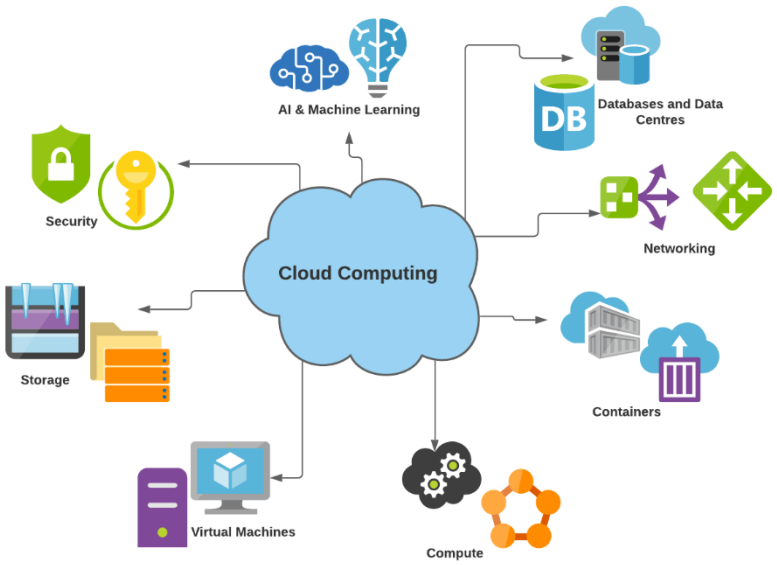

Figure 1.1 : Cloud Computing [1].

1.3 Architecture du cloud computing

L'architecture du cloud computing se réfère aux composants et sous-composants requis pour le cloud computing. Ils se composent généralement d'une plate-forme frontale ou front-end (gros client, client léger, appareil mobile), de plate-formes de back-end (serveurs, stockage), d'une livraison basée sur un cloud et d'un réseau (Internet, Intranet, Intercloud). Combinés, ces composants forment l'architecture du cloud computing [4].

1.3.1 L'extrémité avant (front-end)

Front End fait référence à la partie client du système de cloud computing. Il se compose d'interfaces et d'applications requises pour accéder aux plateformes de cloud computing, par exemple : Navigateur Web.

1.3.2 L'extrémité arrière (back-end)

Back-End fait référence au cloud lui-même. Il se compose de toutes les ressources requises pour fournir des services de cloud computing ; d'un vaste stockage de données, de machines virtuelles, de mécanismes de sécurité, de services, de modèles de déploiement, de serveurs, etc. Il incombe à l'arrière-plan de fournir des mécanismes de sécurité intégrés, des contrôles de trafic et des protocoles. Le serveur utilise certains protocoles, connus sous le nom de middleware, qui aide les périphériques connectés à communiquer les uns avec les autres.

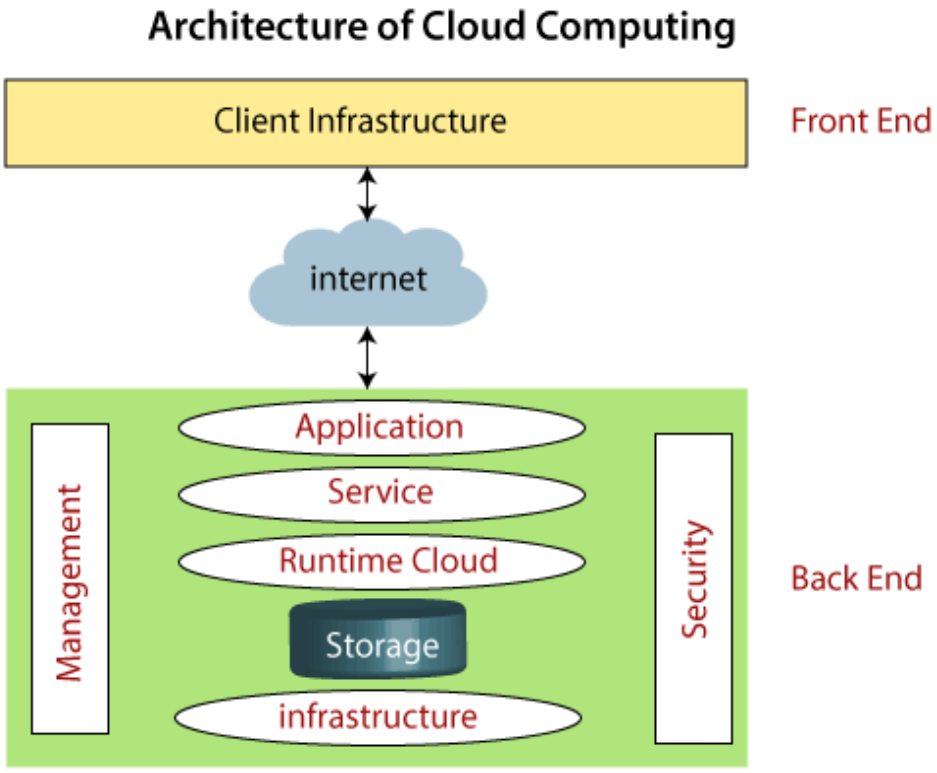

Figure 1.2 : Architecture du cloud computing [2].

1.4 Caractéristiques du Cloud Computing

Selon le NIST, Il existe cinq caractéristiques clés du cloud computing [1] :

1. Self-service, à la demande : Capacité à fournir une ressource automatiquement, sans requérir d'interaction humaine coté fournisseur. Cloud Computing permet aux utilisateurs d'utiliser les services Web et les ressources sur demande. On peut se connecter à un site Web à tout moment et les utiliser.

2. Bande passante élevée et accès réseau universel : les ressources mises à disposition des utilisateurs sont accessibles via Internet de n'importe quel endroit et en utilisant des plateformes hétérogènes (PC, téléphone, mobile, PDA, ordinateur portable, etc.) grâce à la capacité actuelle des réseaux.

3. Ressource partagées, mises en commun (pooling) : Cloud Computing permet à plusieurs locataires de partager un pool de ressources. On peut partager une instance physique, de base de données et d'infrastructure de base.

4. Elasticité rapide : possibilité de faire évoluer très rapidement la capacité fournie, que ce soit en plus ou en moins.

5. Mesurable : capacité à mesurer le service fourni. L'utilisation des ressources peut être surveillée, contrôlée et signalée, offrant la transparence pour le fournisseur et le consommateur du service utilisé. Cette caractéristique permet en particulier de payer le service à l'usage « pay as you go ».

Le pay as you go : les utilisateurs paient les ressources qu'ils utilisent en fonction de leur consommation réelle et précise.

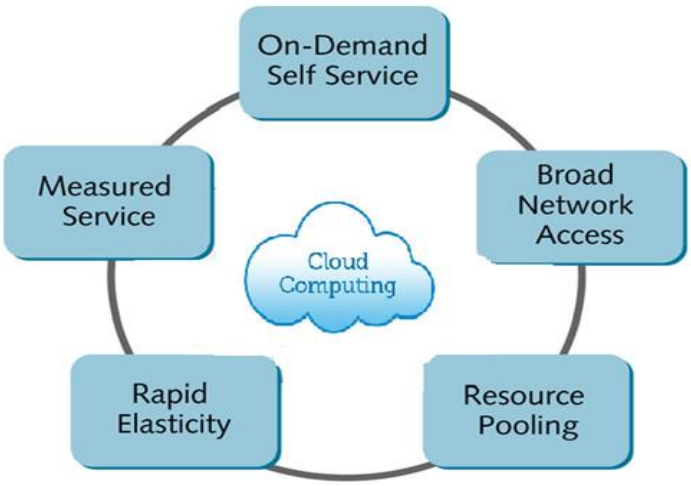

Figure 1. 3 : Caractéristiques du Cloud Computing [54].

1.5 Les niveaux de services du Cloud Computing

Les modèles de service sont les modèles de référence sur lesquels le Cloud Computing est basée. Ceux-ci peuvent être catégorisés en trois modèles de services de base, comme indiqué ci-dessous [5] :

- Logiciel en tant que service (SaaS)
- Plate-forme en tant que service (PaaS)
- Infrastructure en tant que service (IaaS)

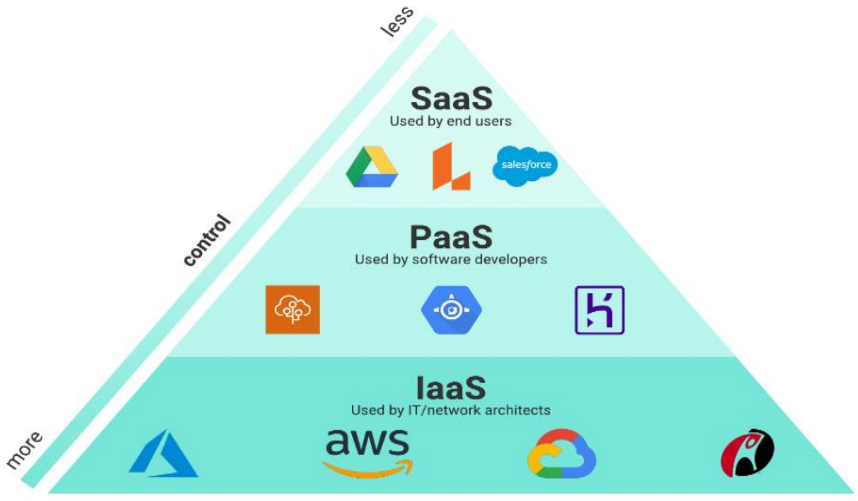

Figure 1.4: Les différentes niveaux des services du cloud computing [54].

1.5.1 SaaS (Software as a service)

C'est-à-dire que 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 [6].

Par Exemple : messagerie, sauvegarde en ligne.

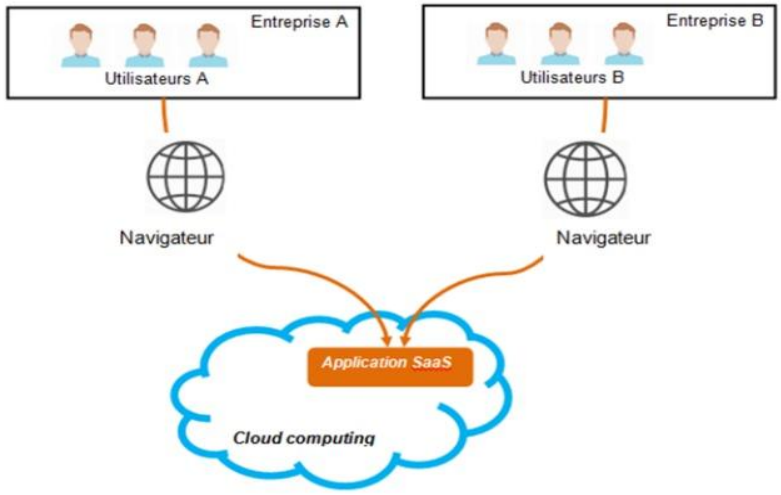

Figure 1.5 : Vision générale des SaaS [55].

1.5.2 Platform as a Service (PaaS)

PaaS fournit l`environnement d'exécution pour les applications, les outils de développement Cette plateforme peut être utilisée pour exécuter des sites web, des SaaS ou tout développement et l`architecture de la plateforme PaaS c'est-à-dire les applications déployées sur l`infrastructure du provider sont basées sur des langages de programmation et des outils supportés par Les développeurs utilisent les outils du fournisseur pour créer leurs propres applications, ils peuvent souvent créer des applications web sans avoir à installer le moindre outil sur leur poste de Ils ont ensuite la possibilité de déployer leurs applications sans pour autant avoir besoin de entreprises de déployer des applications web sans devoir supporter la complexité et les coûts (Serveurs, Stockage, systèmes d`exploitation, réseaux, etc.) mais il a le contrôle sur les applications [6] .

Par Exemple : hébergement des sites web.

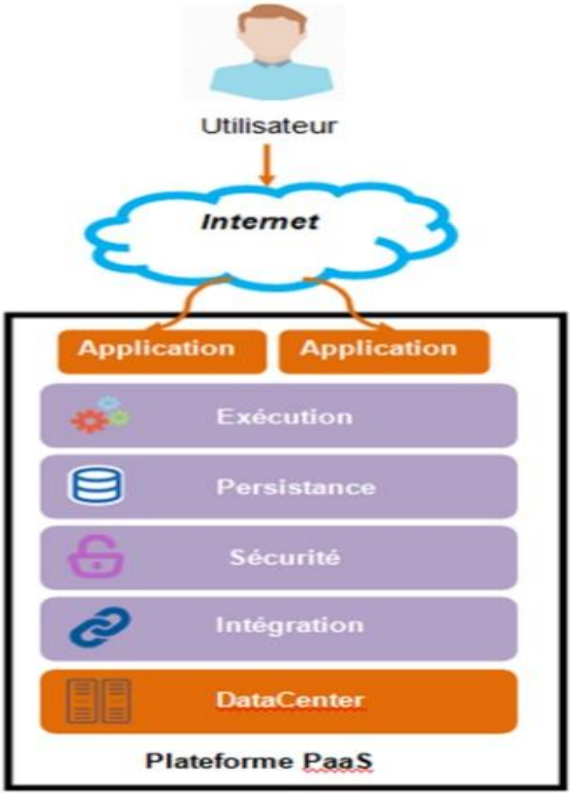

Figure 1.7 : Les plateformes PaaS [55].

1.5.3 Infrastructure as a Service (IaaS)

C'est-à-dire que Les services IaaS du Cloud Computing mettent à disposition des Utilisateurs des ressources matérielles (réseau, stockage, systèmes d'exploitation) accessible sous format virtuelle [6].

Par Exemple : hébergement de serveurs virtuels.

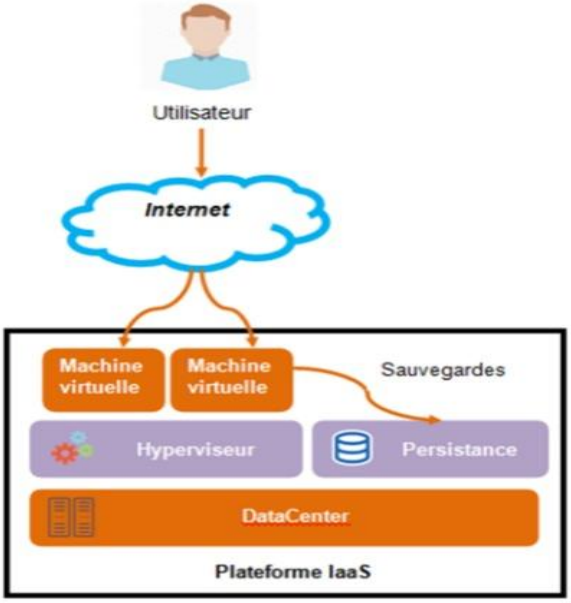

Figure 1.6 : les infrastructures IaaS [55].

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

	Avantage	Inconvénient
Saas	• Pas d'installation • Plus de licence • Migration	• Logiciel limité • Sécurité • Dépendances Des prestataires
Iaas	• Administration • Personnalisation • Flexibilité d'utilisation	• Sécurité • Besoin d'un administrateur système
Paas	• Pas d'infrastructure nécessaire • Pas d'installation • Environnement Hétérogène	• Limitation des langages • Pas de personnalisation dans la configuration des machines virtuelles

Tableau 1.1 - Avantages et Inconvénients des niveaux de services Cloud.

1.6 Les modèles de déploiement du Cloud Computing

Selon NIST, Le modèle de déploiement est généralement divisé en quatre catégories [1] :

a) **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.

b) **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.

c) **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.

d) **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.

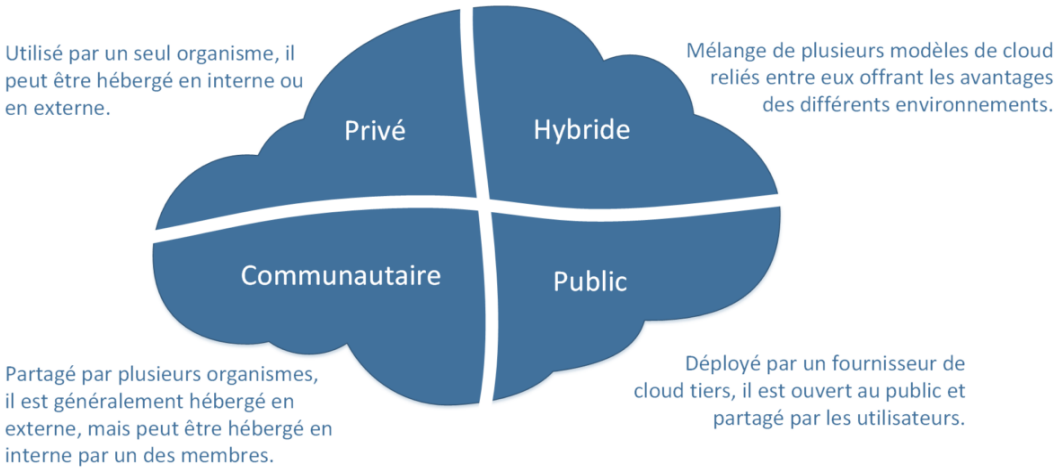

Figure 1.8 : Les modèles de déploiement du Cloud [1].

1.7 Cloud Computing et Virtualisation

1.7.1 Définition

La virtualisation est un concept qui s'est imposé depuis la deuxième moitié de la décennie 2000 à 2010 et qui a rendu crédible, au même titre qu'Internet, la notion même du Cloud. En effet, la virtualisation est totalement indissociable de la notion d'élasticité et de montée en charge [7].

La virtualisation est l'abstraction du matériel physique pour produire des ressources virtuelles permettant d'exécuter plusieurs machines virtuelles (VM) sur une seule machine physique. L'utilisation de ces différents OS donne à l'utilisateur l'illusion d'être en présence de plusieurs machines distinctes.

Comme illustré dans la figure 1.9, la virtualisation se compose de trois composantes principales [7] :

1) Le système d'exploitation principal installé sur un ordinateur physique, appelé système hôte car il agit comme hôte pour d'autres systèmes d'exploitation. Dans certains types d'hyperviseur (type 1), le système d'exploitation hôte n'existe pas d'où l'hyperviseur est installé directement sur le niveau matériel.

2) Un hyperviseur : est un outil de virtualisation installé sur un système hôte qui fournit un environnement dans lequel s'exécutent diverses machines virtuelles.

3) Système d'exploitation installé sur une machine virtuelle, appelé invité, qui s'exécute indépendamment des autres systèmes invités sur d'autres machines virtuelles.

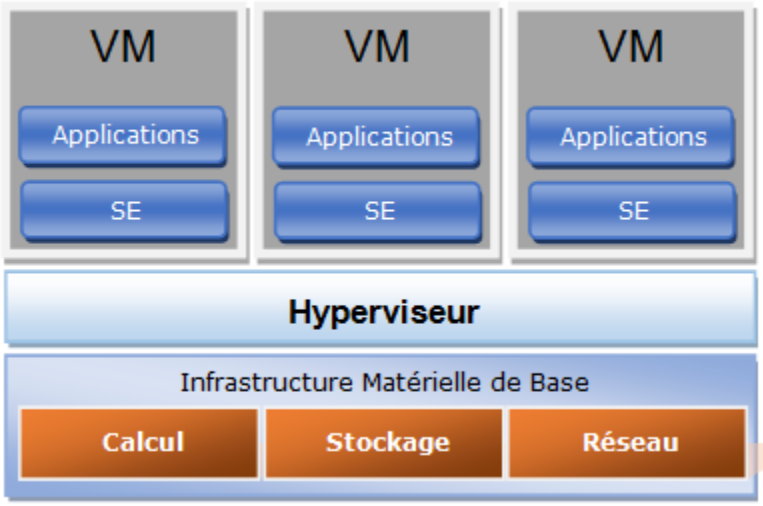

Figure 1.9 : la virtualisation [56].

1.7.2 Hyper viseurs

Un hyper viseur est une couche logicielle qui permet à de nombreux systèmes d'exploitation d'être installés et exécutés simultanément sur la même machine physique. Cette couche légère contrôle le processeur et les ressources du système hôte sur lequel elle est installée et les alloue en tant que ressources virtuelles aux systèmes invités, tout en fournissant une variété d'autres fonctions [8].

Il existe différents hyper viseurs payants et gratuits. Ils peuvent être divisés en deux types :

- Hyperviseur de type 1 appelé Native ou Bare Metal
- Hyperviseur de Type 2 appelé Hyperviseur hébergé

a) **Hyper viseur Natif (Type 1)** : est installé directement sur le niveau matériel du serveur. Logiciel qui s'exécute sur la plate-forme. Ce dernier sert d'outil de gestion des systèmes d'exploitation hébergés. Les exemples incluent Hyper-V, Xen Server et VMware [8].

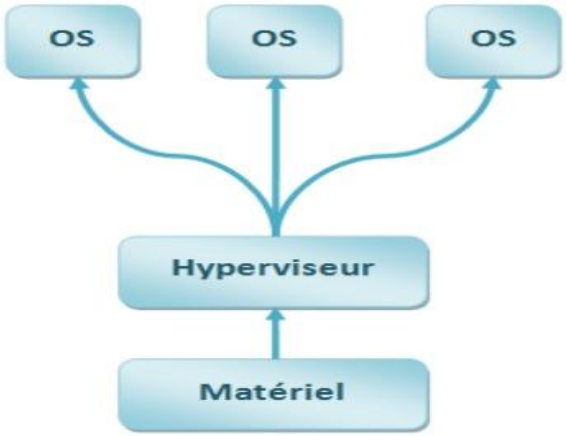

Figure 1.10 : hyperviseurs type 1 [56].

b) **Hyper viseur Hébergé (Type 2)** : est installé avec un système d'exploitation installé sur l'ordinateur hôte. Cela commence par une simple application. La machine virtuelle hébergée est en cours d'exécution Complètement indépendant du système d'exploitation hôte. Les exemples incluent KVM, Virtual Box [

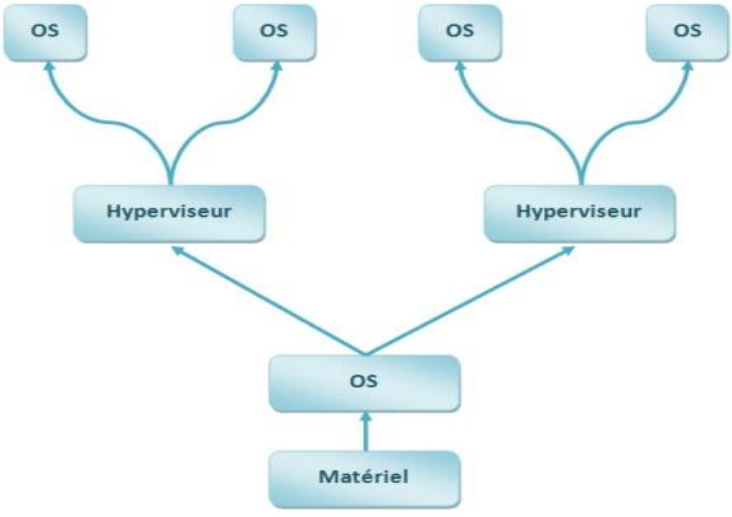

Figure 1.11 : hyperviseurs type 2 [56].

1.7.3 Types de virtualisation

Il existe six virtualisations du cloud computing, comme indiqué dans la figure 1.12 :

1.7.3.1Virtualisation de serveur

La virtualisation de serveur est le masquage des ressources du serveur, y compris le nombre et l'identité des serveurs physiques individuels, des processeurs et des systèmes d'exploitation -- des utilisateurs du serveur. L'intention est d'épargner à l'utilisateur d'avoir à comprendre et à gérer des détails compliqués des ressources du serveur tout en augmentant le partage et l'utilisation des ressources et en maintenant la capacité à s'étendre plus tard [9].

La couche logicielle qui permet cette abstraction est souvent appelée l'hyper viseur. L'hyper viseur le plus courant Type1 est conçu pour s'asseoir directement sur métal et offrent la possibilité de virtualité la plate-forme matérielle à utiliser par le virtuel machines (VM). La virtualisation KVM est un hyper viseur de virtualisation basé sur le noyau Linux qui offre des avantages de virtualisation de type 1 similaires à ceux des autres hyper viseurs. KVM est sous licence Open source. Un hyper viseur de type 2 nécessite un système d'exploitation hôte et est plus souvent utilisé pour tests/laboratoires.

1.7.3.2 Virtualisation des applications

La virtualisation des applications est une technologie logicielle qui améliore la portabilité et la compatibilité des applications en isolant une application du système d'exploitation qui l'exécute. Elle consiste à encapsuler une application et son contexte d'exécution système dans un environnement partitionné. La virtualisation des applications nécessite l'ajout d'une couche logicielle entre un programme donné et le système d'exploitation. Son but est d'intercepter toutes les opérations d'accès ou de modification de fichier ou de registre et de les rediriger vers un emplacement virtuel (généralement un fichier) de manière totalement transparente. Cette opération est transparente, l'application n'a donc pas connaissance de son état virtuel. Le terme de virtualisation d'application est trompeur car il ne s'agit pas de virtualisation d'application, mais du contexte dans lequel l'application s'exécute (registres du processeur, système de fichiers, etc.) [10].

1.7.3.3 Virtualisation du réseau

La virtualisation du réseau consiste à déplacer toutes les fonctions de l'appareil, telles que la commutation, le routage et le VPN, dans un logiciel. Par conséquent, tous les clients Cloud ont

des capacités de mise en réseau en plus des capacités de serveur et de stockage. Même si cela lui reste majoritairement transparent.

Ce réseau est appelé "superposition", ce qui signifie qu'une couche de logiciel est placée et contrôlée au-dessus du matériel. Utilisez ensuite la technologie de tunnel pour connecter ces réseaux virtuels entre eux. Vlan (limité à 4096 réseaux) et plus récemment VXLAN ou NVGRE [11].

1.7.3.4Virtualisation de bureau

La virtualisation de bureau consiste à virtualiser une charge de poste de travail plutôt qu'un serveur. Ce permet à l'utilisateur d'accéder au bureau à distance, généralement à l'aide d'un client léger au bureau. Depuis le poste de travail s'exécute essentiellement dans un serveur de centre de données, l'accès à celui-ci peut être à la fois plus sécurisé et portable. La licence du système d'exploitation doit également être prise en compte que les infrastructures [9].

1.7.3.5 Virtualisation du stockage

La virtualisation du stockage est le processus de séparation de la représentation logique de l'espace de stockage de la réalité physique. Son but est de limiter l'impact des changements structurels sur l'architecture de stockage, en ignorant les périphériques de stockage utilisés et leurs interfaces associées (SATA, SCSI, etc.).

Il s'agit d'une couche logicielle qui combine plusieurs zones de stockage appelées volumes physiques et vous permet de diviser cette zone de stockage globale en partitions virtuelles appelées volumes logiques selon vos besoins [10].

1.7.3.6 Virtualisation des données

La virtualisation des données fait abstraction des détails techniques traditionnels des données et des données gestion, comme l'emplacement, la performance ou le format, en faveur d'un accès plus large et plus résilience liée aux besoins de l’entreprise [9].

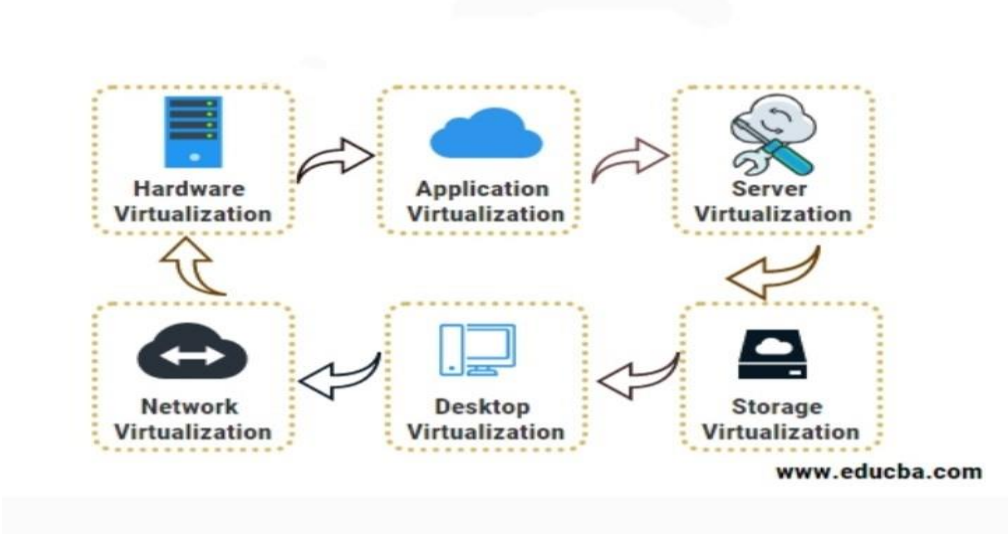

Figure 1.12 : Types de virtualisation [57].

1.7.4 Machine virtuelle vs Conteneur

Le concept de conteneurs est souvent proche de celui de virtualisation. Mais il y a des différences ici aussi. Comme son nom l'indique, une machine virtuelle est une imitation virtuelle d'un appareil informatique, créée dans le cadre de la virtualisation à l'aide d'un logiciel hyper viseur et équipée d'un système d'exploitation (ou OS) complet.

La virtualisation avec conteneurisation, quant à elle, consiste en un partitionnement direct au niveau du système d'exploitation. Ainsi, chaque conteneur exécute son environnement mais utilise le même système d'exploitation hôte. Pour cette raison, les conteneurs sont généralement utilisés pour virtualiser des programmes plutôt que des serveurs entiers [12].

Le tableau 1.2 est une comparaison de différents aspects entre les machines virtuelles et les conteneurs :

	Machines virtuelles	**Conteneurs**
Système d'exploitation	Chaque machine virtuelle s'exécute sur du matériel virtualisé et possède son propre noyau chargé en mémoire	Toutes les applications partagent le même système d'exploitation et le même noyau
Communication	Se produit sur les appareils Ethernet	Mécanismes IPC standard comme les signaux, les tuyaux

Performance	Les performances de la machine virtuelle sont dégradées car les instructions doivent être traduites pour le système hôte	Les conteneurs offrent des performances presque natives, car ils n'ont pas besoin de traduire les instructions pour le système hôte
Temps de démarrage	Le démarrage de la machine virtuelle prendra quelques minutes	Le système d'exploitation hôte est déjà opérationnel, de sorte que le conteneur peut démarrer en quelques secondes
Isolation	Il n'est pas vraiment possible de partager des bibliothèques et des fichiers entre les invités et les hôtes invités	Les sous-répertoires peuvent être montés et libérés de manière transparente
Sécurité	Dépend de la mise en œuvre de l'hyper viseur	Vous pouvez utiliser le contrôle d'accès obligatoire

Tableau 1.2 - Comparaison entre les machines virtuelles et les conteneurs.

1.8 Allocation de ressources dans le Cloud

Nous avons identifié définition des ressources de Cloud [13] :

« *Les Ressources de Cloud peuvent être vus comme n'importe quelle ressource (physique ou virtuel) que les utilisateurs peuvent demander du Cloud, Par exemple, les utilisateurs peuvent demandées des exigences de réseau, telles que la bande passante et les délais, et des exigences computationnelle, telles que le processeur, la mémoire et le stockage. En général, les ressources sont situées dans un centre de donné es qui est partagé par plusieurs clients, et doivent être attribués et ajustés dynamiquement en fonction de la demande* ».

L'environnement de Cloud computing se compose de deux acteurs principaux : les utilisateurs et les fournisseurs. Un utilisateur dispose d'applications avec différentes charges de travail exécutées dans des centres de données Cloud gérés par un fournisseur. Les fournisseurs maintiennent de grandes quantités de ressources informatiques dans de grands centres de données (centres de données) et allouent ces ressources aux utilisateurs [14].

L'utilisateur envoie une requête (Request VM) contenant ces besoins en ressources au fournisseur. Les requêtes se font généralement via des réservations (CPU, RAM, espace de stockage, bande passante, etc.). Lorsque ce dernier reçoit la demande, il cherche des ressources

pour satisfaire la demande et alloue ces derniers à l'utilisateur demandeur, généralement sous forme de machines virtuelles (VM), le fournisseur propose Différents modes de réservation [15] :

a. **Réservation immédiate** : La demande de ressources doit être satisfaite immédiatement.

b. **Réservation planifiée** (également appelée plan de ressources côté fournisseur) : Le client précise la quantité de ressources souhaitée, la date à laquelle les ressources seront disponibles et leur durée de vie.

Ensuite les utilisateurs exécutent des applications à l'aide des ressources qui leur sont attribuées. Ainsi, lorsque la réservation est déclenchée, la machine virtuelle est démarrée, effectue toutes les opérations de configuration et devient disponible pour que les utilisateurs puissent y accéder et l'utiliser à distance. Les ressources sont libérées à la fin de la réservation. Cependant, les clients peuvent signer des accords dits de SLA perpétuels avec leurs fournisseurs, afin qu'ils puissent accéder aux ressources immédiatement quand ils en ont besoin.

De plus, le mécanisme d'allocation des ressources vise à réduire le gaspillage des ressources, à réduire le temps de répartition des tâches et à réduire la satisfaction des utilisateurs.

Premièrement, nous définissons le problème de placer une machine virtuelle donnée sur une machine physique donnée. Surtout sur le modèle de demande de réservation avancée.

1.8.1 L’allocation efficace des ressources

L'allocation des ressources disponibles aux consommateurs de Cloud est une tâche difficile. Le provisioning des ressources Cloud sont proposées avec un accord de niveau de service (SLA) à l'esprit. Une allocation efficace des ressources devrait éviter les situations suivantes [16] :

a. **Sur-provisioning :** se pose lorsque le fournisseur alloue pour le consommateur des ressources plus que la demande.

b. **Sous-provisioning** : se produit lorsque l’allocation des ressources est moins que la demande.

c. **Situation conflictuelle de ressource :** se pose lorsque deux applications tentent d'accéder à la même ressource en même temps.

d. **Le manque en ressources :** se pose lorsque les ressources sont limitées.

e. **La fragmentation des ressources** : cette situation se pose lorsque les ressources sont isolées. [Il y a suffisamment de ressources, mais l'allocation est impossible.]

1.9 Conclusion

Dans ce chapitre, nous avons présenté une vue d'ensemble du Cloud computing, Son architecture, ses caractéristiques, ses modèles de déploiement et de services. Nous avons défini la virtualisation, ses types, la définition de la notion d'allocation de ressources dans le Cloud.

Dans le chapitre suivant, nous présentons Les problèmes d'optimisation dans le cloud computing.

Chapitre II :

Les problèmes d'optimisation dans le cloud sous contraintes énergie

2.1 Introduction

Comme le Cloud Computing devient de plus en plus accessible et populaire, une évolution naturelle consiste à étendre les différents concepts de services, plateformes et infrastructure au provisionnement à la demande afin d'assurer une certaine connectivité entre les ressources virtuelles et les services fournis par le Cloud.

Dans ce chapitre, nous commençons par donner les différents problèmes d'optimisation d'une manière générale dans le cloud puis nous allons présenter le problème de placement de machines virtuelles, les stratégies de placement : méthodes exactes et méthodes approchées, et on conclure avec les indicateurs de performance pour un algorithme de placement de VMs et l'approche d'optimisation multi-objectif.

2.2 Les problèmes d'optimisation dans le cloud

Le Cloud computing n'est pas une révolution technologique en soi, mais une orientation vers la gestion des ressources informatiques. Cependant, l'idée d'héberger plusieurs applications de différents utilisateurs pose certain problème d'optimisation dans le cloud parmi eux :

2.2.1 Ordonnancement

Le problème d'ordonnancement consiste à organiser dans le temps l'exécution d'un ensemble de tâches, compte tenu des contraintes temporelle tel que le délai et d'autres contraintes portant sur l'utilisation et la disponibilité des ressources requises tout en satisfaisant un ou plusieurs objectifs (coût, qualité, délai, énergie, etc…) Dans un problème d'ordonnancement plusieurs notions fondamentales interviennent. Parmi ces notions on trouve : les tâches, les ressources, les contraintes et les objectifs [17].

- **Tâches :**

Les tâches ou les travaux sont triés par date de début et Une date de fin, et sa réalisation nécessite une durée préalablement définie. il se compose d'un Ensemble d'opérations dont l'exécution nécessite certaines ressources et qui nécessite Programmez de manière à optimiser certains objectifs.

- **Resources :**

La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche et elle est disponible en quantité limitée.

- **Contraintes :**

Les contraintes représentent les limitations imposées par l'environnement ou les ressources, ou les utilisateurs.

• Exemple : le budget, l'échéance.

- **Objectif :**

Ce sont les critères à optimiser, c'est un ou plusieurs fonctions à minimiser ou à maximiser dans l'étude.

• Exemple : temps total d'exécution, cout monétaire, l'énergie consommée

2.2.2 Migration de machines virtuelles

La virtualisation apporte des avantages significatifs au Cloud computing en permettant la migration des machines virtuelles et l'équilibrage des charges de travail entre les centres de données. Cela permet une alimentation robuste et très réactive dans votre centre de données. L'un des principaux avantages de la migration de VM est d'éviter les points chauds (hot spots). Cependant, y parvenir n'est pas facile. Aujourd'hui, nous manquons de flexibilité pour détecter le hot spot, initier des migrations et répondre aux changements soudains des charges de travail [18].

On distingue deux types de migration des machines virtuelles :

1. Migration à froid (STOP and COPY migration).

2. Migration à chaud (Livre migration).

- **Migration à froid (STOP and COPY migration)**

La migration à froid des VMs est la stratégie la plus basique, elle consiste à (i) Mettre la VM hors tension afin de copie son état sur le serveur hôte cible. (ii) Une fois que la mémoire soit transférée entièrement la VM reprend son activité dans le même état que lorsqu'elle s'est arrêtée. (iii) Le temps total de la migration est égal au temps d'arrêt de la machine virtuelle [19].

- **Migration à chaud (LIVE migration)**

La migration à chaud est une stratégie utilisée pour pallier les problèmes de migration à froid. Elle permet à un hyperviseur (i) de transférer une machine virtuelle en cours d'exécution d'un hôte à un autre sans interruption de service. Elle assure (ii) une durée d'interruption minimale de la machine virtuelle. Il existe deux principales approches de live migration [20] :

1. Migration pré-copie.
2. Migration post-copie.

2.2.3 Problème de placement des machines virtuelles

Le placement de machine virtuelle est une opération importante effectuée pour déterminer le meilleur MP ou serveur pour héberger une machine virtuelle. Le choix du bon hôte est essentiel pour améliorer l'efficacité énergétique, l'utilisation des ressources et la prise en charge de la qualité de service (QoS) dans votre environnement de Cloud computing. Le déploiement de MV dans le Cloud computing est très important, mais c'est une tâche très complexe, et pourquoi ?

Premièrement, le schéma d'arrivée des demandes d'instanciation MV peut être imprévisible. Deuxièmement, alors que la taille des centres de données est souvent importante pour une charge donnée, trouver des scénarios optimaux ou non optimaux est toujours NP-difficile. Car par exemple, si n est le nombre total de machines virtuelles et m est le total nombre de serveurs, alors le nombre de mappage possible peut être (m puissances n).

Le problème du placement des machines virtuelles peut être divisé en deux parties. La première partie consiste à accepter de nouvelles demandes de déploiements de machines virtuelles et à placer les machines virtuelles acceptées sur les hôtes (placement initial des machines virtuelles) et l'autre partie à optimiser le placement des machines virtuelles.

Il existe deux types de placement de la machine virtuelle dans les machines physique [21]. :

a) **Placement statique de MV :** s'exécuté lorsque le système est en mode hors ligne ou lors du démarrage du système. Cela constitue le placement statique initial dans l'environnement du cloud computing.

b) Placement dynamique de MV *:* Le placement initial peut changer en raison de changements dans les charges système spécifiques. De plus, les algorithmes de Placement dynamique de VM peuvent être classés en :

- **Placement réactive de MV** : Le déploiement initial après que le système a atteint un certain état n'est pas souhaitable. Des modifications peuvent être apportées en raison de problèmes de performances, de maintenance, d'alimentation ou de charge, ou de violations spécifiques de SLA.
- **Placement proactive de MV** : qui Modifier une machine virtuelle à partir d'une machine physique (MP) initialement placée avant que le système n'atteigne un certain état.

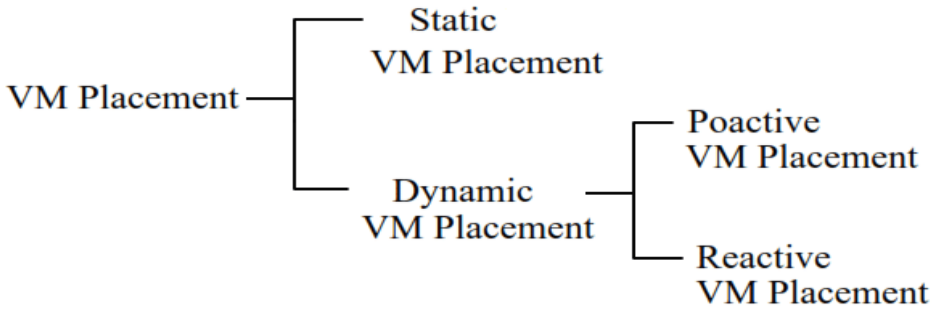

Figure 2.1: placement des Machines Virtuelles

2.3 Classification des algorithmes d'optimisation du problème de placement des VMs dans les datacenters du Cloud

Les méthodes d'optimisation pour le problème de placement des VMs sont des techniques utilisées pour trouver les valeurs optimales des paramètres d'un système ou d'une fonction, il existe deux types :

- ✓ Les méthodes exactes.
- ✓ Les méthodes approchées

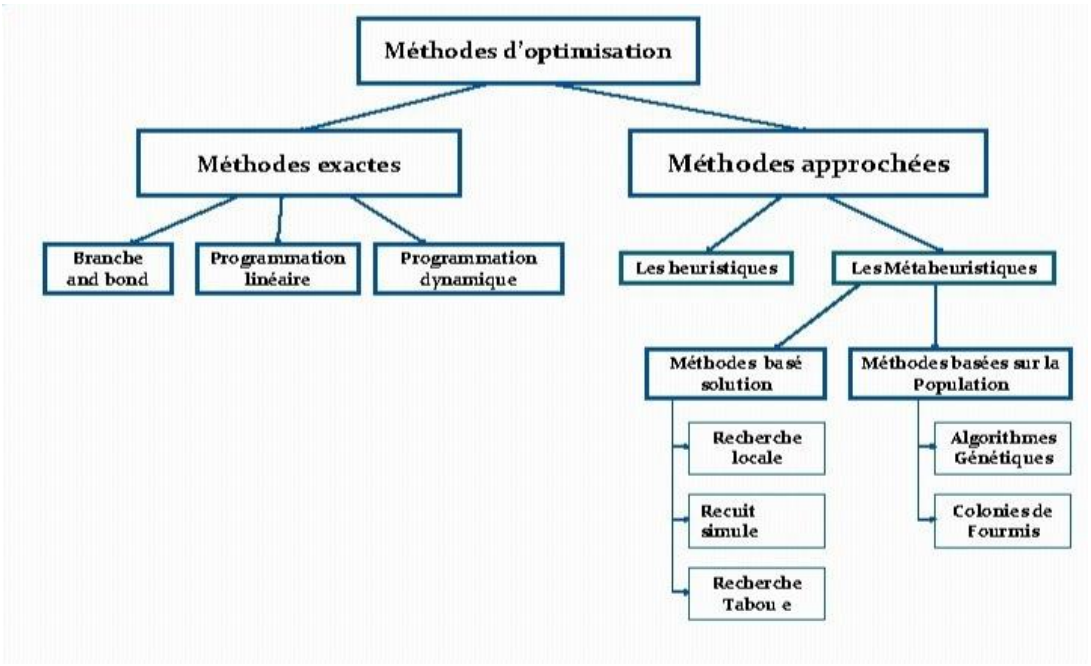

Figure 2.2 : classification de méthodes d'optimisation de placement des VMs [58].

2.3.1 Les méthodes exactes

Les méthodes exactes sont garanties que l'optimum global est trouvé, mais est souvent très gourmand en temps de calcul et en ressources pour des problèmes plus grands ou plus complexes. Parmi ces méthodes, on peut citer Branche & Bound et la programmation dynamique et la programmation linéaire [22].

a) La méthode de recherche arborescente (Branch &Bound) :

C'est l'un des moyens de fournir un mécanisme de recherche très intelligent, qui vous permet de faire bon usage de l'espace de recherche et d'atteindre la solution optimale plus rapidement que d'autres méthodes exactes. Branch &Bound est basé sur trois axes principaux (séparation, évaluation et la stratégie de parcours) [23].

b) La programmation dynamique :

Une technique de résolution de problèmes pour déterminer l'ordre optimal. Le contenu des décisions à prendre. L'idée de base est de combiner les meilleures solutions à partir d'un ensemble de solutions. Les sous-problèmes vous permettent de trouver la solution optimale

Votre problème. Ceci comprend Choisissez des séquences de décision courtes. La solution du problème est calculée Ordre croissant. Cela signifie commencer par le plus petit sous-problème. Que nous revenons au plus grand [24].

c) La programmation linéaire :

La programmation linéaire (PL) est un domaine d'optimisation qui peut être utilisé pour ce faire. Résoudre divers problèmes de combinaison. Qu'est-ce que la programmation linéaire ? Résoudre des problèmes où l'objectif et les contraintes sont tous linéaires [25].

Si l'ensemble des solutions possibles S'est exprimé comme un ensemble de variables Les valeurs de l'ensemble des nombres réels R ont des contraintes qui doivent être satisfaites Il existe des inégalités linéaires et le problème que f est une fonction linéaire dans ces variables. Programmation linéaire (PL). De nombreux problèmes réels peuvent survenir en recherche opérationnelle Exprimez-le comme un problème PL. Pour cette raison, il existe de nombreux algorithmes qui permettent cela. D'autres solutions de problèmes d'optimisation sont basées sur la résolution de problèmes linéaires [24].

2.3.2 Les méthodes approchées

Générer des solutions proches de l'optimum dans un délai raisonnable. Ils conviennent à la résolution d'instances de problèmes à grande échelle, mais ne sont pas garantis pour trouver un optimum global : les heuristiques, les méta-heuristiques [26].

2.3.2.1 Les méthodes heuristiques :

En optimisation combinatoire, une heuristique est un algorithme d'approximation qui permet d'identifier au moins une solution rapidement réalisable, pas nécessairement optimale, en temps polynomial. Une heuristique est généralement développée pour un problème spécifique en fonction de sa structure. Les heuristiques peuvent être divisées en deux catégories [27] :

- ✓ **Méthode constructive :**

Qui génère des solutions à partir d'une solution initiale et essaie d'ajouter des éléments petit à petit jusqu'à Ce qu'une solution complète soit atteinte

- ✓ **Méthodes de fouilles locales** :

Qui commence par une solution parfaite (probablement pas très intéressante), de manière répétitive ET continue d'essayer de l'améliorer en explorant les voisinages.

2.3.2.2 Les méta-heuristiques

Les métaheuristiques sont une famille d'algorithmes stochastiques permettant de résoudre des problèmes d'optimisation. La plupart des métaheuristiques utilisent des processus aléatoires et itératifs pour recueillir des informations, explorer des espaces de recherche et traiter des problèmes tels que l'explosion combinatoire.

Une caractéristique commune à de nombreuses métaheuristiques est qu'elles sont développées sur la base de processus se produisant dans la nature comme la biologie (algorithmes évolutionnaires et génétiques), la physique (recuit simulé) et aussi l'éthologie (algorithmes de colonies de fourmis).

Les métaheuristique peuvent être classées en deux catégories [27] :

A) Les Métaheuristiques À Solution Unique (S-méta-heuristiques) :

Toutes les méthodes itératives de solution unique sont basées sur des algorithmes de recherche voisinage commençant par la première solution et s'améliorant progressivement en choisissant de nouvelles solutions dans le voisinage. Voici quelques-unes des méthodes les plus couramment utilisées : le recuit simulée, la méthode de descente et la recherche Tabou.

1) Les méthodes de descente (Hill Climbing) :

Les méthodes de descente sont assez anciennes et doivent leur succès à leur rapidité et leur simplicité. A chaque pas de la recherche, cette méthode progresse vers une solution voisine de meilleure qualité. La descente s'arrête quand tous les voisins candidats sont moins bons que la solution courante c'est-à-dire lorsqu'un optimum local est atteint. On distingue différents types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente stochastique et la descente vers le premier meilleur [28].

2) La recherche Tabou (TabuSearch) :

La méthode taboue est une autre technique développée par Glover, 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 [29.30].

B) Les métaheuristiques à population de solution (P-méta-heuristiques) :

La méthode d'optimisation de la population de solutions améliore un ensemble de solutions nommé population évolue en parallèle. Le but de ces méthodes est d'utiliser la population comme facteur de diversité comme l'algorithme de colonies de fourmis, l'algorithme génétique, Optimisation par Essaim Particulaire [27].

1) Les algorithmes de colonies de fourmis (Ants System) :

Les algorithmes de colonies de fourmis ont été proposés par Colonie, Dori go et Maniezô en 1992 et appliquées la première fois au problème du voyageur de commerce. Les fourmis déposent des substances odorantes appelées phéromones sur le sol, laissant derrière elles une traînée chimique. Les fourmis peuvent sentir ces phéromones, qui agissent comme des balises. Lorsque les fourmis choisissent une route, elles ont tendance à choisir la route avec la plus forte concentration de phéromones. De cette façon, ils peuvent retrouver leur chemin vers le nid à leur retour. Ailleurs, les odeurs peuvent être utilisées par d'autres fourmis pour localiser les sources de nourriture découvertes par d'autres fourmis [31].

2) Les algorithmes génétiques :

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique basée sur les mécanismes biologiques tels que les lois de Mendel et sur le principe fondamental

(Sélection) de Charles Darwin. Les algorithmes génétiques simulent le processus évolutif des populations. Travailler avec eux est très facile. On part d'une population de solutions initiales sélectionnées au hasard (chromosomes), on évalue leur fonction d'adaptation (Fitness). Sur la base de ces résultats, une nouvelle population de solutions potentielles est créée à l'aide d'opérateurs évolutifs simples de sélection, de croisement et de mutation [32,33].

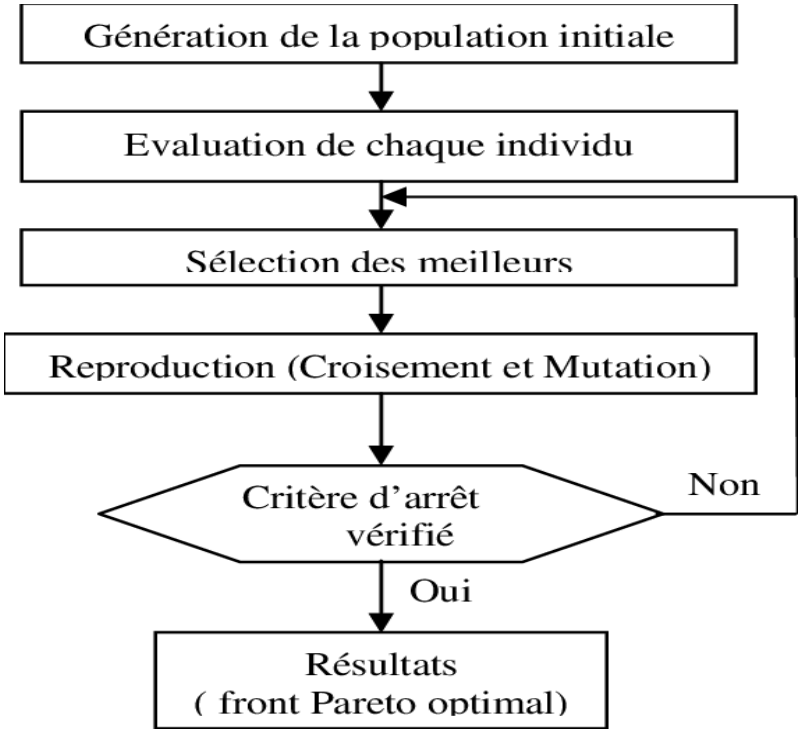

Figure 2.3: Fonctionnement d'un algorithme génétique[59].

Un algorithme génétique nécessite :

- **Population initiale :**

Initialement, on génère un nombre aléatoire d'individus afin de construire la population initiale.

- **Sélection :**

La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de converger vers l'optimum global.

Il existe plusieurs techniques de sélection. Voici les principales utilisées :

- **Sélection par rang *:*** Cette technique de sélection choisit toujours les individus possédant les meilleurs scores d'adaptation.

- **Probabilité de sélection proportionnelle à l'adaptation** : Technique de la roulette ou roue de la fortune, pour chaque individu, la probabilité d'être sélectionné est proportionnelle à son adaptation au problème.
- **Sélection par tournoi *:*** Cette technique utilise la sélection proportionnelle sur des paires d'individus, puis choisit parmi ces paires l'individu qui a le meilleur score d'adaptation.
- **Sélection uniforme *:*** La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation.

➢ **Croisement :**

Les opérateurs de croisement produisent un ou deux enfants à partir de deux parents, c'est le résultat obtenu lorsque deux chromosomes partagent leurs particularités.

Il existe deux méthodes de croissement : croissement point ou croissement multiples points.

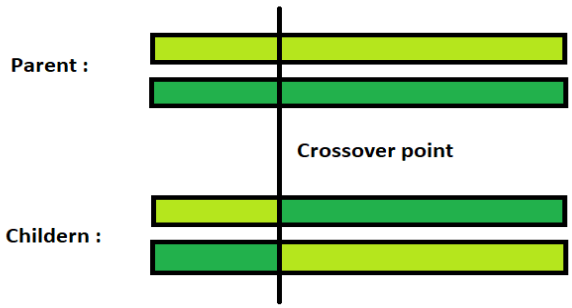

Figure 2.4 : croissement de point[60]

➢ **Mutation :** Une mutation consiste un processus qui applique de petits changements dans le code génétique aux individus pour introduire de la diversité et éviter de tomber dans des optima locaux.

➢ **Fonction d'adaptation (Fitness)** *:* Pour calculer le coût d'un point de l'espace de recherche, l'évaluation de la fonction d'adaptation est l'étape au cours de laquelle la performance de chaque individu est mesurée. Afin de pouvoir

évaluer les qualités individuelles et les comparer aux autres, il est nécessaire d'établir des normes communes d'évaluation.

3) L'optimisation par essaims de particules (PSO) :

Développée par Russel Eberhart et James Kennedy aux États-Unis en 1995, est une méthode d'optimisation par essais et erreurs [34]. 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 [35]. 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-psychologiques pour créer PSO.

2.4 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 [36] :

P(u) = Pidle+ (Pbusy– Pidle) * u

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

- **Violation du SLA :**

Le SLA est le niveau de service attendu d'un fournisseur de services. On note une violation des SLA par hôte actif par (SLATAH : **SLA** violation **T**ime per **A**ctive **H**ost). Il s'agit du pourcentage de temps moyen pendant lequel les hôtes actifs ont atteint 100 % d'utilisation du processeur, d'où :

$$\mathbf{SLATAH} = \frac{1}{N} \sum \frac{T_{si}}{T_{ai}}$$

N : est le nombre d'hôtes actifs

T_{si} : est le temps total de l'hôte i accompli l'utilisation du processeur à 100% conduisant à la violation du SLA

T_{ai} : est le temps total pendant lequel l'hôte i est dans l'état actif.

- **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 :

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

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.

- **Le coût :**

La plupart des fournisseurs de cloud ont fixé des prix pour l'utilisation de leurs services. Ils ont fixé le prix du transfert d'une unité de donnée entre deux services et le prix pour le traitement par unité de temps.

Le coût total d'exécution des tâches dans l'IaaS est défini dans l'équation (2.1) :

$$\boldsymbol{Coût} = \sum_{i,j=0}^{N,M} \mathbf{CoûtEx(i,j)} + \sum_{i,j=0}^{N,M} \mathbf{CoûtCom(i,j)} \quad \textbf{(2.1)}$$

Ou,

N est le nombre de tâches à exécuter

M est le nombre de DC

CoûtEx (i, j) est le coût d'exécution de la tâche i par le DC j

CoûtCom (i, j) est le coût de communication de la tâche i avec le DC j

$$\textbf{CoûtEx (i, j) = Temps}EX_{ij}\textbf{*Prix}U_{EX} \quad \textbf{(2.2)}$$

$$\textbf{CoûtCom (i, j) = Temps}Com_{ij}\textbf{ *Prix}U_{Com} \quad \textbf{(2.3)}$$

Ou,

TempsEx (i, j) est le temps d'exécution de la tâche i par le DC j

TempsCom (i, j) est le temps de communication de la tâche i avec le Datacenter j

- **L'assurance de la QoS des applications** :

La qualité est l'ensemble des fonctionnalités et des caractéristiques d'un produit ou d'un service qui porte sa capacité à satisfaire les besoins exprimés et implicites. Il existe de nombreux facteurs affectent la qualité des systèmes [37] :

- ✓ **Convivialité** : l'interface utilisateur du logiciel doit être facile à utiliser.
- ✓ **Sécurité :** Des politiques de sécurité telles que l'authentification et l'autorisation doivent être mises en œuvre.
- ✓ **La performance :** c'est le temps de réponse du logiciel. Robustesse, un logiciel est robuste s'il est disponible même en état de panne.
- ✓ **Compatibilité des plates-formes :** le logiciel de qualité doit fonctionner sur autant de plates-formes que possible afin que de nombreux utilisateurs puissent l'utiliser.

- ✓ **Maintenabilité :** le système doit être modifiable et capable de gérer la fonctionnalité sans détruire le système pour atteindre la qualité.

- ➢ **Le nombre de machines virtuelles allouées :**

Pour calculer le nombre de machines virtuelles attribuées dans le cloud computing, vous devez prendre en compte les éléments suivants :

- ✓ **Besoins en ressources :** déterminez les besoins en ressources pour chaque machine virtuelle, tels que la quantité de mémoire, la puissance de calcul (nombre de cœurs de CPU virtuels), l'espace de stockage et la bande passante du réseau. Ces exigences varient en fonction de la charge de travail spécifique exécutée sur la machine virtuelle
- ✓ **Types de machines virtuelles :** Les fournisseurs de services cloud proposent généralement différents types de machines virtuelles avec des configurations prédéfinies. Chaque type a ses propres spécifications en termes de ressources, de performances et de coûts.
- ✓ **Optimisation des ressources :** Certains fournisseurs de services cloud offrent des fonctionnalités d'optimisation des ressources, telles que l'évolutivité automatique où

  les groupes d'équilibrage de charge, qui peuvent ajuster dynamiquement le nombre de machines virtuelles allouées en fonction de la demande

2.5 Optimisation multi-objectif

De manière informelle, un problème multi-objectif peut être défini comme un problème d'optimisation dont le but est de déterminer une solution qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectives. La difficulté avec les problèmes multi-objectifs est qu'il n'y a pas de solution optimale. En fait, lorsque les objectifs considérés sont contradictoires, il n'existe généralement pas de solution optimale pour tous les objectifs. Ainsi, S'il est facile de dire qu'une solution est meilleure qu'une autre, vous ne pouvez pas dire avec certitude qu'une solution est meilleure que toutes les autres.

Par conséquent, le but de la résolution de problèmes multi-objectifs est de déterminer toutes les solutions satisfaisantes. Pour cette raison, les méthodes de résolution de ce type de problème sont appelées méthodes d'aide à la décision. C'est parce que le choix final se résume à la tâche. Aux décideurs (dans ce cas, les utilisateurs des ressources cloud) [38].

2.5.1 Formulation générale d'un problème d'optimisation multi-objectif

Un problème d'optimisation combinatoire multi-objectif, MOP (Multi Objectif Combinatoire Optimisation Problème) peut être défini par [38] :

(MOP) = {Min F(x)=(f1(x), ..., f_n (x)) s, c, x∈ X}

Où n : est le nombre d'objectifs (n ≥2)

Le vecteur X= (x1, …, X_K) ∈ X : représente le vecteur de k variables de décision, X représente l'ensemble de solutions réalisables dans l'espace décisionnel.

Dans le cas combinatoire : X est un ensemble discret. À chaque solution x ∈ X a associé un vecteur objectif sur la base d'un vecteur de fonctions f : X→ Z tel que :

Z= (z1,. . . . , Z_n)= f(x)= (f1(x), ..., f_n (x)). L'ensemble Z = f(X) représente les points réalisables dans l'espace objectif.

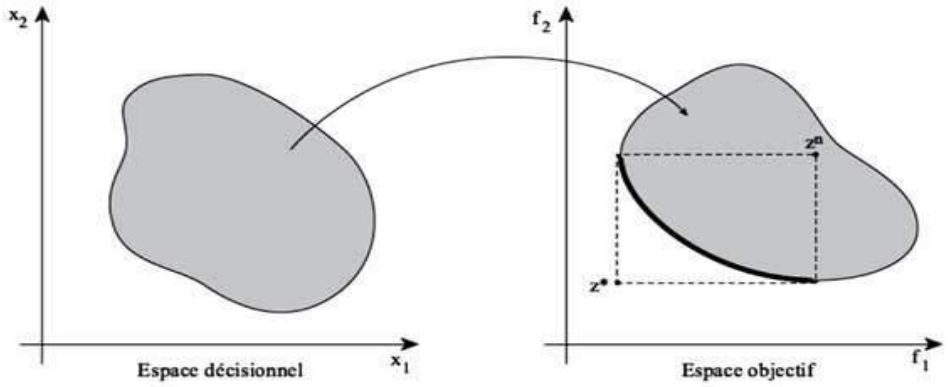

Figure 2.5: Espace décisionnel et espace objectif d'un MOP (cas de deux Variables de décision X1 et X2 et de deux fonctions objectif f1 et f2) [51].

2.5.2 Notions de dominance

- **Définition (Dominance de Pareto) :**

Un vecteur objectif z ∈ Z domine un vecteur objectif z′ ∈ Z si les deux conditions suivantes sont vérifiées [38] :

∀i ∈ {1, . . ., n}, , $Z_i \leq , Z_i$` **(1)**

$\exists j \in \{1, \ldots, n\}, , Z_i \leq , Z_i`$ **(2)**

Si z domine z′ cette relation sera notée z > z`. Par extension, une solution x ϵ X domine une solution x`∈ X, noté x > x`, ssi f(x) > f(x`).

- **Optimalité de Pareto :** La définition de la solution optimale de Pareto découle directement de la notion de dominance. Ce concept a été initialement développé par F.Y et Généralisé par Edge Worth et V. Pareto

  - **Définition (solution Pareto optimale) :**

Une solution x ∈ X est Pareto optimale (ou non dominante) si et seulement si ∀ x` ∈X, x`≻ x. Une solution optimale de Pareto peut donc être considérée comme optimale, car il est impossible d'améliorer la valeur cible sans dégradation.

Le concept d'optimalité de Pareto était à l'origine utilisé en économie et en sciences de gestion Cependant, il trouve ses racines dans les travaux d'Edgeworth (1881) et Pareto (1896). Le front de Pareto de cet ensemble dans l'espace objectif, noté z_N [38].

- **Définition (Ensemble Pareto optimal) :**

Étant donné un MOP (fx), l'ensemble Pareto optimal est défini comme suit [38] :

XE = {x ∈X| ∄ x`∈ X, x`≻ x}

- **Définition (Front Pareto) :**

Etant donné un MOP (f, X) et son ensemble Pareto optimalx_E XE, le front Pareto est

Défini comme suit : z_N= {f(x) | x∈ XE}

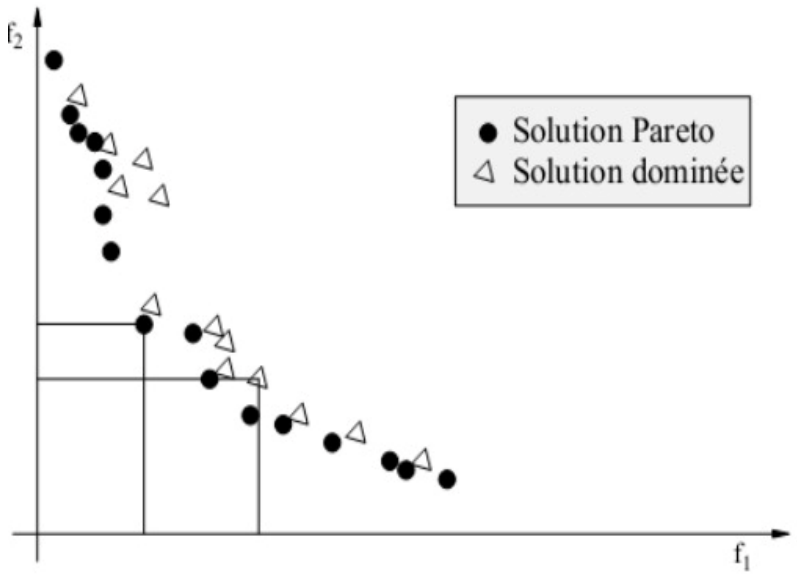

Figure 2.6: Front Pareto Optimal [51].

2.5.3 Optimisation multi-objectif et aide à la décision

La résolution d'un problème multi-objectif nécessite la détermination d'un ensemble de solutions optimales de Pareto. Maintenir la sélection finale des solutions nécessite une implication humaine à travers les décideurs. La première décision à prendre lorsqu'on s'attaque à des problèmes multi-objectifs est une combinaison de processus de recherche et de prise de décision. Cela peut être fait de l'une des trois manières suivantes [38] :

a. **A priori :** Dans ce cas, après le processus d'optimisation, le décideur intervient pour donner une priorité du problème à résoudre afin de soutenir le chemin de la solution dans sa recherche. En pratique, cela transforme le problème d'optimisation multi objectif original en un problème à objectif unique. Ce dernier peut être résolu par une méthode qui donne la solution souhaitée en une seule fois. Cette approche nécessite une connaissance a priori du problème. C'est rapide, mais il faut tenir compte du temps de modélisation des préférences et de la possibilité que le décideur ne soit pas satisfait de la solution trouvée et relance la recherche avec une formulation différente de la préférence.

b. **A posteriori** : La méthode de résolution tente de fournir au décideur un ensemble de solutions optimales de Pareto bien réparties. Parmi cet ensemble de solutions, il peut choisir celle qui, selon lui, correspond le mieux à ses goûts. Par conséquent, nous n'avons plus besoin de modéliser les préférences des décideurs, mais nous devons fournir un ensemble de solutions bien réparties, ce qui peut être difficile et coûteux en calculs.

c. **Interactive :** Cette approche implique une collaboration directe et itérative entre les décideurs et les solutions. Les décideurs interviennent pendant la recherche en ajustant les préférences pour guider la recherche. Cette approche peut tenir compte des préférences du décideur, mais nécessite la présence du décideur tout au long du processus de recherche.

2.4 Conclusion

Nous avons traité dans ce chapitre Les problèmes d'optimisation dans le Cloud, ou il nous traité le problème de placement des machines virtuelles dans le Cloud centres de données. Enfin nous avons passé sur les différents méthodes d'optimisation et l'optimisation multi objectif d'une façon générale.

Dans le chapitre suivant, nous présenterons une méthode approchée « le récuit simulé Multi-objectif » pour résoudre le problème de placement des VMs dans le cloud.

Chapitre III :

Recuit Simulé Multi-objectif Pour Le Placement Des VMS Dans Les centre de données Cloud

3.1 Introduction

Les problèmes d'optimisation combinatoire NP-complets sont caractérisés par une complexité exponentielle ou factorielle, par conséquent ; il est impossible d'énumérer toutes les solutions possibles car cela dépasse la capacité de calcul de n'importe quel ordinateur actuel. Il est donc très difficile de trouver la solution optimale par une méthode exacte.

Pour palier à ce problème, les chercheurs ont introduit des méthodes approchées appelées heuristiques, elles présentent l'avantage d'un temps de calcul réduit mais ne donnent aucune information sur la qualité de la solution trouvée, de plus elles ne sont en général applicables qu'a un seul type de problème [39].

Par exemple la méthode de la descente consiste à partir d'une solution S à choisir une solution S'dans un voisinage de S, telle que S'améliore la solution. La recherche s'arrête donc au premier minimum (ou maximum) local rencontré, c'est là son principal défaut. Pour améliorer les résultats, on peut relancer plusieurs fois l'algorithme mais la performance de cette technique décroît rapidement.

A cet effet, les chercheurs sont poussés à proposer de nouvelles méthodes d'utilisation générale (applicables à la plupart des problèmes d'optimisation) appelées métaheuristiques, y compris la méthode du recuit simulé, conçu pour rechercher un optimum global parmi plusieurs minimas (ou maximas) locaux.

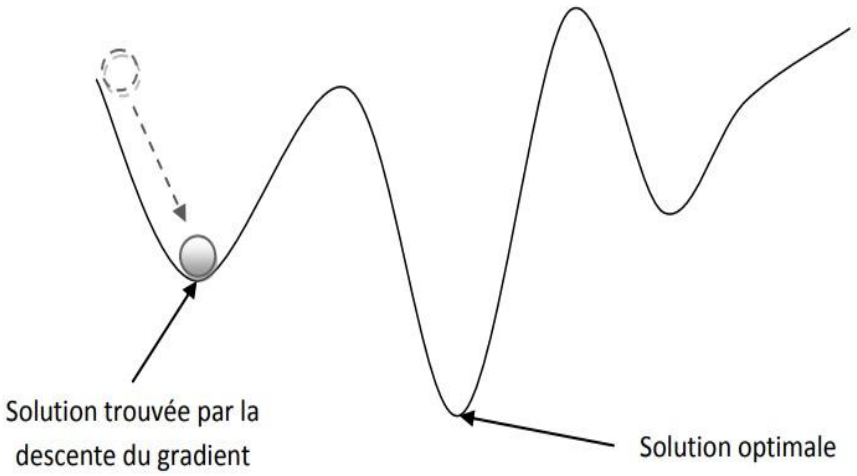

Figure 3. 1 : blocage d'une heuristique classique dans un minima local [45].

3.2 Motivations du choix de l'algorithme du recuit simulé

Dans la littérature, plusieurs métaheuristiques ont été mise œuvre pour résoudre les problèmes d'optimisation mono-objectif aussi bien que ceux multi-objectifs. Le choix d'un tel algorithme se base sur un ensemble de critères affectant particulièrement la qualité des solutions trouvées. En effet, le recuit simulé présente les caractéristiques principales suivantes [40,41] :

1) Exploration de l'espace de recherche : Le recuit simulé multi objectif permet d'explorer efficacement l'espace des solutions possibles d'un problème multi objectif. En utilisant des stratégies de refroidissement adaptées, il est capable de traverser des régions de l'espace de recherche contenant des solutions potentiellement intéressantes, même si elles ne sont pas optimales selon les critères évalués.

2) Recherche de compromis : Le recuit simulé multi objectif vise à trouver un ensemble de solutions qui représentent un compromis entre les différents objectifs.

3) Évitement des optima locaux : Comme le recuit simulé utilise un processus de recherche stochastique, il peut échapper aux optima locaux, c'est-à-dire des solutions qui sont localement optimales mais ne représentent pas la meilleure solution globale.

4) Adaptabilité aux contraintes : Le recuit simulé multi objectif peut être adapté pour prendre en compte des contraintes supplémentaires dans le problème d'optimisation.

3.3 Le recuit simulé (Simulated Annealing)

3.3.1 Les origines

La méthode du recuit simulé est une généralisation de la méthode Monte Carlo. Son but est de trouver une solution optimale pour un problème donné. Elle a été mise au point par trois chercheurs de la société 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 très utilisé en métallurgie pour obtenir un alliage sans défaut, ce processus est appelé « le recuit » [39].

On commence d'abord par chauffer le métal jusqu'à une certaine température où il devient liquide (les atomes peuvent donc circuler librement). Après avoir atteint ce stade, on abaisse la température très lentement de sorte à obtenir un solide.

Si cette baisse de température est brusque on obtient alors du verre ; si au contraire cette baisse de température est très lente (laissant aux atomes le temps d'atteindre l'équilibre statistique), nous obtiendrons des structures de plus en plus régulières, jusqu'à atteindre un état d'énergie minimale correspondant à la structure parfaite d'un Crystal, on dit alors que le système est « gelé ».

Au cas où cet abaissement de température ne se ferait pas assez lentement, il pourrait apparaitre des défauts. Il faudrait alors les corriger en réchauffant de nouveau légèrement la matière de façon à permettre aux atomes de retrouver la liberté de mouvement, leur facilitant ainsi un éventuel réarrangement conduisant à une structure plus stable.

3.3.2 Définition recuit simulé

Le recuit simule est issu d'une analogie avec le phénomène thermodynamique de recuit des métaux. La méthode imite une procédure utilisée depuis des millénaires par les métallurgistes qui, pour obtenir un alliage exempt de défauts, chauffent d'abord à blanc leur morceau de métal et laissent ensuite l'alliage se refroidir très lentement. En effet, lorsqu'on chauffe un métal solide il devient liquide a une certaine température, dans ce cas, les atomes qui le composent voient leur degré de liberté augmenter. Inversement, lorsqu'on baisse la température, le degré de liberté diminue et les atomes s'organisent en structure atomique parfaite, proche de l'état d'énergie minimale afin d'obtenir un solide. Cependant, lorsque la baisse progressive de la température est trop rapide (comme pour la Trempe), le solide présente alors des défauts, on tombe dans un minimum local d'énergie. La technique du recuit redonne de la liberté aux atomes pour tenter d'atteindre un nouvel état dynamique. Se basant sur révolution d'un système thermodynamique, le recuit simulé accepte, pour sortir d'un minimum local, une dégradation de la fonction de cout avec une certaine probabilité. Cet algorithme a été proposé par Kirkpatrick, Gelatt, et Vecchi en 1983 à partir de **la méthode de Metropolis** qui était utilisée pour modéliser les processus physiques, après avoir constaté que la recherche des minimas locaux était de même nature dans les deux cas (physique et informatique) [42].

3.3.3 Principe De La Méthode Du Recuit Simulé :

La méthode de recuit simulé part d'une solution initiale admissible et continue l'exploration de l'espace d'états en effectuant des perturbations mineures sur la solution courante. Si la nouvelle solution obtenue est améliorée alors elle est retenue. Si elle est détériorée par rapport au critère d'optimisation alors elle est retenue avec une probabilité inversement proportionnelle au nombre d'itérations. Le recuit simulé a l'avantage de couvrir un espace de recherche plus grand et d'éviter la convergence prématurée vers un optimum local. Plusieurs problèmes de job shop ont été traités par la méthode de recuit simulé [43].

3.3.4 Algorithme De Metropolis :

En 1953, Metropolis avait proposé un algorithme itératif qui permet d'atteindre l'état d'équilibre thermodynamique d'un système simulé à une température T.

Son principe consiste à itérer les deux étapes suivantes :

- Évaluer la variation d'énergie associée à une transition élémentaire aléatoire de l'état courant i, d'énergie E_i , vers un nouvel état j, d'énergie Ej : $\Delta E_{ij} = E_j - E_i$;

• accepter la transition vers le nouvel état avec une probabilité P_{ij} [44] où :

$$\boldsymbol{P_{IJ}\,(T) = \exp\left(-\frac{\Delta E}{T}\right)} \qquad \textbf{si } \boldsymbol{\Delta E_{IJ} > 0}$$

Dans un premier temps, T étant généralement choisi très grande, beaucoup de solutions - même celles dégradant la valeur de f - sont acceptées, et l'algorithme équivaut à une visite aléatoire de l'espace des solutions, mais à mesure que la température baisse, la plupart des solutions augmentant l'énergie sont refusés, et l'algorithme se ramène à une amélioration itérative classique.

3.3.5 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 [45] :

Algorithme 1 : Recuit simulé

1. Engendrer une configuration initiale faisable S0 de S, S ← S0
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) = $exp^{-i\mathbf{\Delta E}/t}$
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

A température intermédiaire, l'algorithme autorise de temps en temps des transformations qui dégradent la fonction objective. Il laisse ainsi une chance au système de s'extraire d'un minima local [46]

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.

A. Décroissance de température :

La décroissance de la température est un aspect très important de la méthodologie du 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.

Deux approches sont possibles pour décroitre la température :

- **Décroissance par paliers** : Pour chaque valeur de la température, on itère l'algorithme de Metropolis jusqu'à atteindre un équilibre statistique, puis on diminue la température.
- **Décroissance continue :** On fait baisser la température d'une façon continue, le plus courant est d'utiliser la loi suivante : $T_i+1 = \alpha . T_i / \alpha < 1$ (en général $\alpha = 0.9$ à 0.99) Remarque : Le paramètre α est à choisir avec précaution ; En effet, s'il est choisi trop petit, la température baissera très rapidement et l'algorithme pourra être bloqué dans un minima local ; Si au contraire il est choisi trop grand (proche de 1), la température baissera très lentement et le temps de calcul sera très grand.

B. 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, Traduction et commutation [45].

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

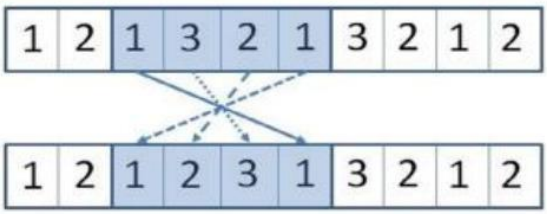

Figure 3.2 : Inversion [45].

- ✓ **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.

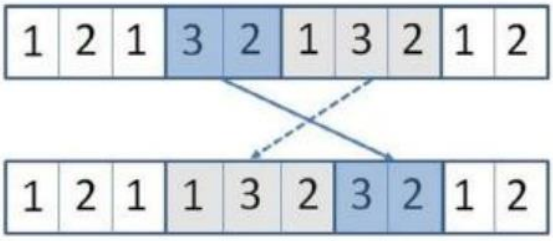

Figure 3.3 : traduction [45].

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

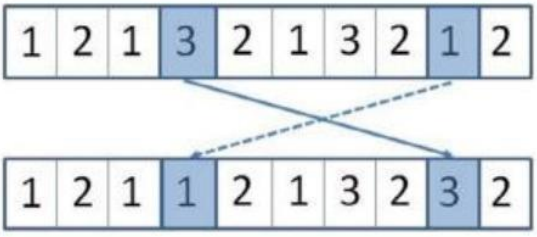

Figure 3.4 : Commutation [45].

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 citées précédemment qui sont limité par l'ensemble des hôtes générés initialement.

Le schéma ci-dessus (figure 3.2) représente la structure générale de l'algorithme de recuit simulé.

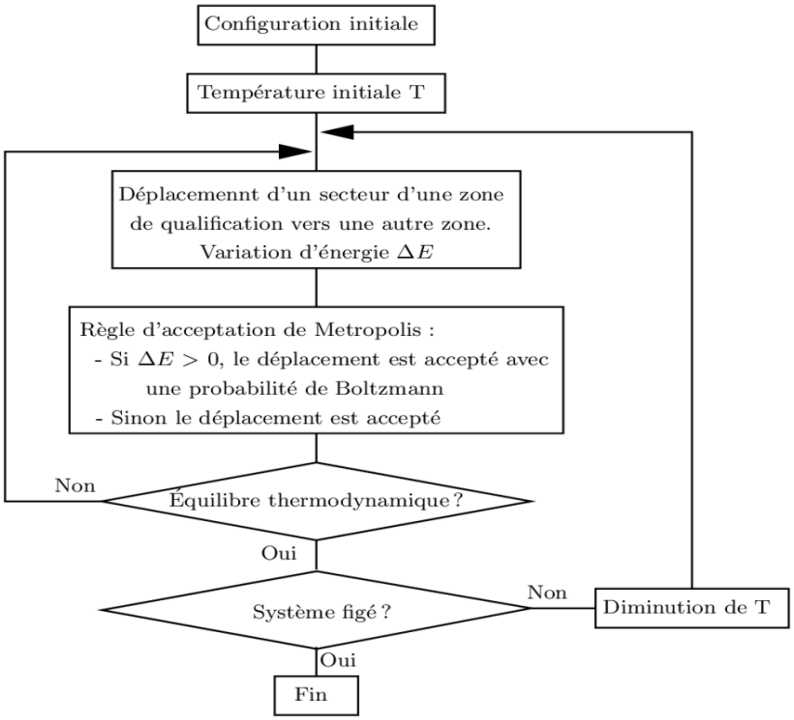

Figure3.5 *:* Organigramme général du recuit simulé [45]

3.4 Le recuit simulé multi-objectif pour le placement des VMs :

Le problème de placement de VMs dans le Cloud Computing étant un problème de complexité NP-Complet, on a utilisé une métaheuristique - le recuit simulé - dont l'objectif est de trouver une solution optimale ou quasi-optimale. L'objectif d'origine du recuit simulé était de résoudre les problèmes d'optimisation mono-objectif, et sachant qu'on a plusieurs objectifs à satisfaire, on a adopté cet algorithme pour résoudre plusieurs objectifs simultanément. Dans le présent travail, on a mis l'accent sur les deux principaux objectifs :

- Réduction de l'énergie consommées dans l'ensemble de l'infrastructure Cloud.
- Réduction du temps de violation des SLA par hôte actif (SLATAH).

3.4.1 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 vi est affectée à l'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

$SLATAH_i$: le pourcentage de temps moyen pendant lequel les hôtes actifs ont atteint 100 % d'utilisation du processeur :

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

$$\begin{pmatrix} \boldsymbol{vp11} & \cdots & \boldsymbol{vp1M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{vpN1} & \cdots & \boldsymbol{vpNM} \end{pmatrix} \qquad \textbf{(1)}$$

Où

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

Pour une affectation VP donnée, l'utilisation du CPU de P_j peut être calculée par :

$$u_j = \frac{\sum_{i=1}^{N} v_{cpu}^{i} * vp_{ij}}{p_{cpu}^{j}} \qquad (3)$$

Fonctions objectives :

Comme cité ci-dessus, deux fonctions (critères) objectives sont mises au point :

A- La consommation d'énergie : la consommation d'énergie d'un hôte P_j est calculée par l'équation suivante [47] [48] [49] :

$$e_j = \begin{cases} 0, & si\ \sum_{1}^{N} vp_{ij} = 0 \\ \left(e_{max}^{j} - e_{idle}^{j}\right) * \frac{u_j}{100} + e_{idle}^{j}, & sinon \end{cases} \qquad (4)$$

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

B- Réduction du temps de violation des SLA par hôte actif SLATAH :

$$\text{SLATAH} = \frac{1}{M} \sum \frac{T_{si}}{T_{ai}} \qquad (5)$$

M : est le nombre d'hôtes actifs

T_{si} : est le temps total de l'hôte i accompli l'utilisation du processeur à 100% conduisant à la violation du SLA

T_{ai} : est le temps total pendant lequel l'hôte i est dans l'état actif.

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

$$\sum_{J=1}^{M} e_j \qquad (6)$$

$$\text{SLATAH} = \frac{1}{M} \sum \frac{T_{si}}{T_{ai}} \qquad (5)$$

Sous contraintes de placement et de capacités suivantes :

$$\forall i, \sum_{J=1}^{M} vp_{ij} = 1 \qquad (8)$$

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

$$\forall j, \sum_{i=1}^{N} v_{mem}^{i} * vp_{ij} \leq p_{mem}^{j} \quad (10)$$

$$\forall j, \sum_{i=1}^{N} v_{net}^{i} * vp_{ij} \leq p_{net}^{j} \quad (11)$$

La contrainte 8 signifie qu'une machine virtuelle ne peut être affectée et ne doit être affectée qu'à un seul hôte, les contraintes 9, 10 et 11 exigent que le total des ressources de CPU, de mémoire ou de réseau affectées aux VMs d'un 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 fonctionne.

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.

3.4.2 Algorithme du Recuit Simulé multi-objectif

- **Méthodes MOSA (*Multi-objective Simulated Annealing*)**

La méthode (MOSA) est une classe d'extensions de recuit simulé à l'optimisation multi-objectif exploitant l'idée de construire un front de Pareto estimé en rassemblant des solutions non dominées trouvées lors de l'exploration du domaine réalisable.

Une archive M est considérée comme servant à la maintenance de ces solutions efficaces. En conséquence, de nombreux objectifs, multi-objectifs, paradigmes, de recuit simulé ont été proposés dans la littérature qui ont en commun la suggestion de considérant une variation d'une certaine énergie composée souvent comme une combinaison linéaire des fonctions objectifs considérées. Sous certains choix de probabilités d'acceptation, la convergence vers l'ensemble des solutions de Pareto a été prouvée.

L'algorithme 2, ci-dessous ; présente une extension du recuit simulé pour supporter un tel problème d'optimisation multi-objectifs [50] :

Algorithme 2 : Recuit simulé multi-objectif pour le placement des VMs

Entrée : Un programme de planification de refroidissement « calendrier de refroidissement » ; une température initiale « température initiale » T← $T0$; un échantillon initial de solutions générées S ; et une mémoire (archive) initiale M ←S

Sortie : L'archive M représentant une approximation de l'ensemble des solutions de Pareto

1 : **Répéter**

2 **:** **Pour chaque** élément x dans S **faire**

3 : **Répéter**

4 : Construire une solution voisine y

5 : **Si** y n'est pas dominée par x **alors**

6 : Mettre à jour M avec y

7 : Sélectionner le plus proche voisin z (s'il existe) de x

8 : Mettre à jour les poids des objectifs en fonction de la dominance partielle de xet y

9 : **Sinon**

10 : Accepter y avec une probabilité $\boldsymbol{P\ (Accept\ S_C)}$

11 : **Fin si**

12 : **Jusqu'à ce que** la condition d'équilibre soit atteinte

13 : **Fin pour**

14 : Diminuer la température

15 : **Jusqu'à ce que** la condition de refroidissement soit atteinte

16 : **Retourner** M

Description de l'algorithme :

1)Initialisation de l'archive

L'initialisation de l'archive M dans l'algorithme du recuit simulé Multi-objectif se fait généralement avant le début du processus d'optimisation. L'objectif principal de l'initialisation de l'archive est de créer un ensemble vide prêt à stocker les solutions non dominées découvertes au fur et à mesure de l'exécution de l'algorithme.

Méthode couramment utilisée pour initialiser l'archive M :

1. **Création de l'archive** : L'archive M est généralement représentée sous la forme d'une liste, d'un tableau ou d'une structure de données similaire, qui peut stocker les

Solutions non dominées. Au début de l'algorithme, M est créée en tant qu'ensemble vide.

2. **Génération d'une solution initiale** : Une solution initiale est générée aléatoirement ou à l'aide d'une méthode spécifique au problème pour commencer le processus d'optimisation.

3. **Évaluation de la solution initiale :** Les valeurs des objectifs associées à la solution initiale sont calculées à l'aide des fonctions objectives du problème. Ces valeurs sont utilisées pour comparer les solutions et déterminer si une solution est dominée par une autre.

4. **Ajout à l'archive :** La solution initiale est ajoutée à l'archive M car elle est la première solution et n'est pas dominée par d'autres solutions (puisque l'archive est vide). La solution est donc considérée comme non dominée par défaut.

Une fois que l'archive M est initialisée avec la solution initiale, le processus d'optimisation commence avec les itérations du recuit simulé Multi-objectif. Au fur et à mesure que de nouvelles solutions sont générées et évaluées, elles sont comparées aux solutions présentes dans l'archive pour déterminer si elles sont non dominées. Les nouvelles solutions non dominées sont ajoutées à l'archive, tandis que les solutions dominées sont ignorées ou éliminées.

2) Mise à jour de l'archive

La mise à jour de l'archive M dans l'algorithme du recuit simulé Multi-objectif se fait tout au long du processus d'optimisation, à mesure que de nouvelles solutions sont générées et évaluées. L'objectif de la mise à jour de l'archive est de conserver un ensemble diversifié de solutions non dominées, représentant le front de Pareto.

Les étapes de la mise à jour de l'archive M dans l'algorithme du recuit simulé multi objectif sont :

1. **Génération d'une nouvelle solution** : À chaque itération de l'algorithme, une nouvelle solution est générée en modifiant légèrement la solution courante. Cette nouvelle solution est évaluée pour obtenir les valeurs des objectifs correspondantes.

2. **Comparaison avec les solutions de l'archive** : La nouvelle solution est comparée aux solutions déjà présentes dans l'archive M pour déterminer si elle est dominée par l'une d'entre elles ou si elle est non dominée et doit être ajoutée à l'archive.

3. **Ajout de la nouvelle solution à l'archive :** Si la nouvelle solution est non dominée par toutes les solutions de l'archive, c'est-à-dire qu'elle n'est pas pire que les solutions existantes dans tous les objectifs et qu'elle est meilleure dans au moins un objectif, elle est ajoutée à l'archive M. Cela garantit que l'archive contient toujours des solutions non dominées et diversifiées.

4. **Élimination des solutions dominées :** Si la nouvelle solution est dominée par une ou plusieurs solutions de l'archive, elle est généralement rejetée et n'est pas ajoutée à l'archive. Cela permet de maintenir l'ensemble des solutions non dominées et de ne conserver que les solutions optimales.

5. **Gestion de la taille de l'archive :** Si la taille maximale de l'archive est atteinte après l'ajout de la nouvelle solution, une stratégie de gestion de l'archive peut être appliquée pour maintenir sa taille. Cela peut impliquer l'élimination de certaines solutions de l'archive, par exemple en utilisant des critères tels que la similarité entre les solutions ou leur contribution à la diversité de l'ensemble.

- **Probabilité d'acceptation :**

Le recuit simulé Multi-objectif trouve son origine dans les travaux de Serafini [51] où de nombreuses règles d'acceptation probabilistes ont été conçues et discutées dans le but d'augmenter la probabilité d'accepter des solutions non dominées, c'est-à-dire des solutions non dominées par aucune solution générée jusqu'à présent. Pour cela, on a maintenu la première approche qui consiste à n'accepter avec certitude que les solutions améliorantes, acceptant ainsi avec une probabilité inférieure à 1 toute autre solution. Cette règle d'acceptation forte permet une exploration approfondie du domaine réalisable et a été exprimée comme suit [51] :

$$\mathbf{P\ (Accept\ S_C) = \prod_{i=1}^{m} min\ (1\ ;\ exp\ (\frac{W_i \Delta E_i}{T}))}$$

Où :

P : est la probabilité d'acceptation,

ΔE_i : Est la différence entre les valeurs des objectifs de la nouvelle solution et de la solution courante,

W_i : c'est la poids (pondération) de la fonction objective i.

T : est la température actuelle du système.

3.5) Etapes principales de l'algorithme

L'algorithme du recuit simulé Multi-objectif suit généralement les étapes suivantes :

1. **Initialisation :** Un ensemble de solutions initiales est générée aléatoirement pour commencer le processus d'optimisation.
2. **Boucle principale :** L'algorithme effectue une série d'itérations pour explorer l'espace des solutions. À chaque itération, une solution voisine est générée en modifiant légèrement la solution courante. La probabilité de transition vers cette nouvelle solution est calculée en utilisant une fonction d'acceptation qui dépend de la différence entre les valeurs des objectifs.
3. **Mise à jour de la solution courante** : La solution courante est mise à jour si la nouvelle solution est acceptée en fonction de la probabilité de transition.
4. **Gestion de l'archive** : L'archive est mise à jour pour conserver les solutions non dominées rencontrées au cours du processus. Les nouvelles solutions non dominées sont ajoutées à l'archive, tandis que les solutions dominées sont éliminées.
5. **Critère d'arrêt :** L'algorithme s'arrête lorsque certaines conditions prédéfinies sont satisfaites, telles qu'un nombre maximal d'itérations ou une stagnation de l'amélioration des solutions.

L'algorithme du recuit simulé Multi-objectif offre une approche efficace pour explorer et rechercher des solutions optimales dans les problèmes Multi-objectifs. En utilisant l'archive pour stocker les solutions non dominées, il fournit une représentation complète des compromis possibles entre les objectifs, aidant ainsi le décideur à prendre des décisions éclairées en fonction de ses préférences.

3.6 Conclusion

Cette section a été consacrée à l'expression de à notre approche « Recuit simulé Multi-objectif » pour résoudre le problème de placement de machines virtuelles dans les centres de données Cloud. Cette technique d'optimisation a été utilisée pour résoudre des problèmes d'optimisation multi objectif. Contrairement aux problèmes d'optimisation mono-objectif où un seul objectif est à maximiser ou à minimiser, les problèmes multi objectifs ont plusieurs objectifs concurrents.

Dans la section qui suit, nous présenterons notre implémentation de cette approche via une simulation en présentant ainsi, les résultats obtenus.

Chapitre IV :

Implémentation Et Résultats Expérimentaux

4.1Introduction

Pour finir notre travail, on va utiliser un simulateur CloudSim Plus de nôtre projet. Les résultats de la simulation seront détaillés à la fin de ce chapitre.

Nous commençons ce chapitre par flatter les outils de simulations utilisées, à savoir, le Language java, l'IDE NetBeans et le simulateur Cloud Sim Plus. Nous présentons comme l'IHM que certains avons développée pour 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 qui :

- Peut-être décrit en quelques mots : Orienté Objet, Simple, Robuste, Dynamique et Sécurisé, Indépendant de la Plateforme (VM), Semi Compilé/Semi Interprété, Bibliothèque Importante (JDK API).
- Il assure la portabilité entre différents environnements (machine/OS).
- Toutes les sources sont disponibles : http://www.openjdk.org
- Est Open Source depuis novembre 2006 (2008 complètement).
- Se trouve sous 3 environnements d'exécutions différents

- Java ME (Micro Edition) pour PDA, téléphone
- Java SE (Standard Edition) pour desktop
- Java EE (Entreprise Edition) pour serveur Servlet/JSP/JSTL, Port let/JSF, JTA/JTS, JDO/EJB Java Mail, etc.

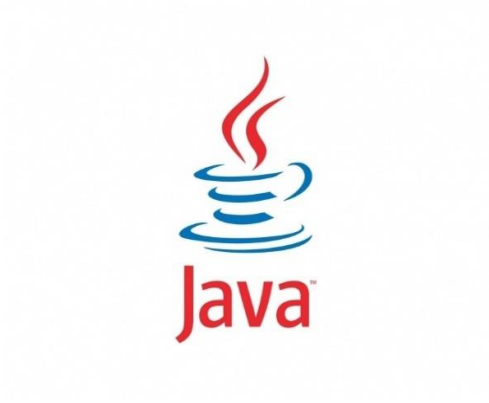

Figure 4.1 : Logo de langage Java.

4.2.2 Environnement de développement NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, éditeur graphique d'interfaces et de pages Web).

NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java [52].

4.2.3 Simulateur CloudSim Plus

4.2.3.1 Définition

CloudSim est une infrastructure logicielle java prenant en charge plusieurs fonctionnalités essentielles du cloud, telles que la file d'attente des tâches, le traitement des événements, la création d'entités du cloud, la communication entre entités, la mise en œuvre de règles de courtier, etc. Le cadre permet aux développeurs de spécifier les caractéristiques des différentes entités des fournisseurs de Cloud, notamment [53] :

1) Testez les services d'application dans un environnement reproductible et contrôlable.

2) Vérifier la performance avant d'utiliser les vrais nuages.

3) Réglez les goulots d'étranglement du système avant de déployer des applications dans un nuage réel.

4) Testez différents scénarios de répartition de la charge de travail et de performances des ressources sur une infrastructure simulée afin de développer et de tester des techniques de provisioning d'applications adaptatives.

4.2.3.2 Architecture générale de CloudSim Plus

La structure logicielle de CloudSim et ses composants est représentée par une architecture en couche comme montré par la figure suivante.

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

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

Dans la couche plus haute de la pile de simulation, on trouve le code de l'utilisateur qui expose la configuration des fonctionnalités liées aux hôtes (nombre de machines, leurs spécifications), les politiques d'ordonnancement de Broker, applications (nombre de tâches et leurs besoins), VM, nombre d'utilisateurs. Elle contient les informations de base nécessaires au bon 62 fonctionnements des hôtes et des applications comme le nombre de machines, leurs spécifications, le choix des stratégies d'ordonnancement du broker, etc.

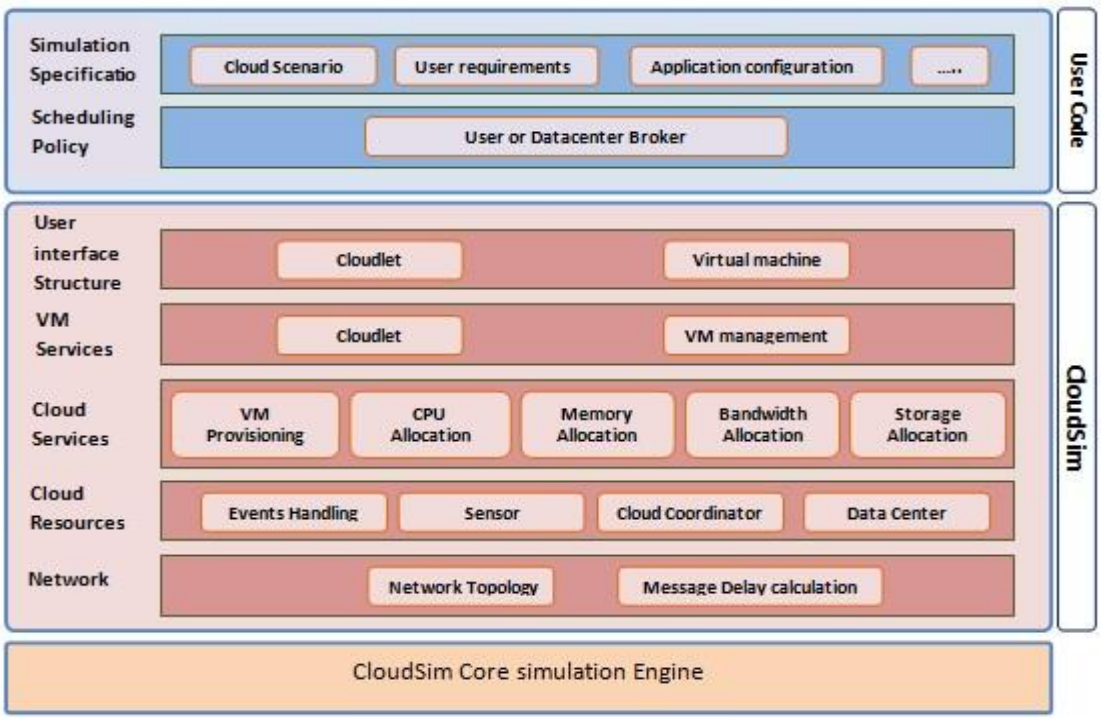

Figure 4.2: Architecture du CloudSim [61].

4.2.3.3 Les classes principaux du CloudSim

CloudSim est composé de plusieurs classes dont nous pouvons classifier en deux catégories : les classes qui modélisent les entités comme le Datacenter, le Broker, etc. et les classes modélisant les politiques d'allocation. Dans la figure 19 nous présentons globalement le diagramme de classe du simulateur CloudSim, pour laquelle nous avons exposé les classes principales.

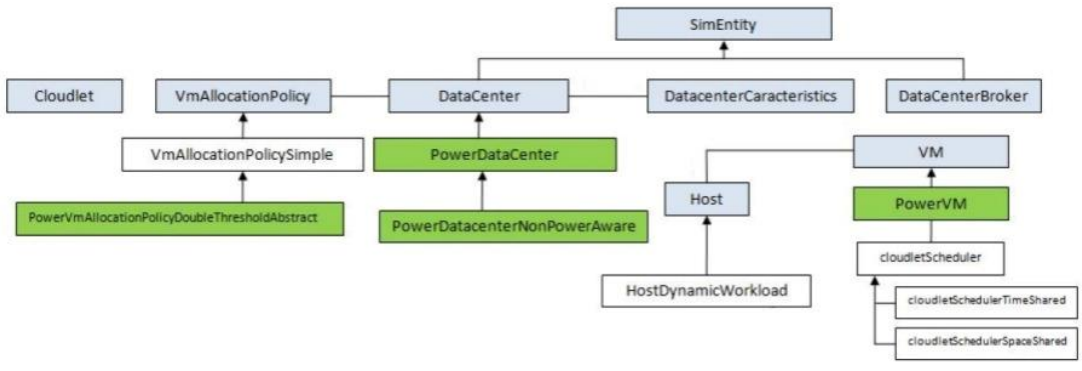

Figure 4.3: Diagramme de classe du simulateur CloudSim [61].

- **SimEntity**

Il s'agit d'une classe abstraite, elle représente l'entité de simulation qui est capable d'envoyer des messages à d'autres entités et de gérer les messages reçus ainsi les évènements. Toutes les entités doivent étendre cette classe et redéfinir ses trois méthodes : **StartEntity (), processeEvent ()** et **shutdownEntity ().** Ces méthodes définissent les actions pour l'initialisation, le traitement des événements et la destruction de l'entité.

- **Cloudlet**

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

- **Datacenter**

Cette classe modélise l'infrastructure du noyau (matériel, logiciel) offert par des fournisseurs de service dans un environnement de Cloud Computing. Il encapsule un ensemble de machines de calcul qui peuvent être homogènes ou hétérogènes en ce qui concerne leur configuration de ressources (mémoire, noyau, capacité et stockage). En outre, chaque composant du Datacenter instancie un composant généralisé d'approvisionnement en ressources qui implémente un ensemble de politiques d'allocation de bande passante, de mémoire et des dispositifs de stockage. La classe Power Datacenter utilisée dans le package « power » hérite de la classe Datacenter et elle permet la simulation des centres de données en calculant leurs énergies consommées.

- **DatacenterBroker**

Cette classe modéliser le courtier, qui est responsable de la médiation entre les utilisateurs et les prestataires de service selon les conditions de QoS des utilisateurs et il déploie les tâches de service à travers les Clouds. Le Broker agissant au nom des utilisateurs identifie les prestataires de service appropriés du cloud par le service d'information du cloud CIS (cloud information services) en négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs.

- **DatacenterCaracteristic**

C'est la classe qui contient les informations sur la configuration des ressources des Datacenter. Parmi eux le système d'exploitation, le VMM, la liste des hôtes, le coût parseconde d'utilisation des ressources, etc.

Host Cette classe représente un serveur informatique physique dans un Cloud. L'Host exécute des actions liées à la gestion des machines virtuelles et a une politique définie

pour l'approvisionnement mémoire et bande passante, ainsi que d'une politique de répartition des PE (Processeur Elément) à des machines virtuelles. Un hôte est associé à un Datacenter. Il peut héberger un ou plusieurs VMs.

- **Host**

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

- **VM**

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

- **VMAllocationPolicy**

C'est une classe abstraire qui représente la politique de provisionnement des hôtes aux machines virtuelles dans un Datacenter. Pour notre politique nous avons utilisé la classe Power VmAllocationPolicy Double Threshold Abstract qui hérite la classe VMAllocationPolicy.

4.3 Interface de l'application :

Interface Principale :

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

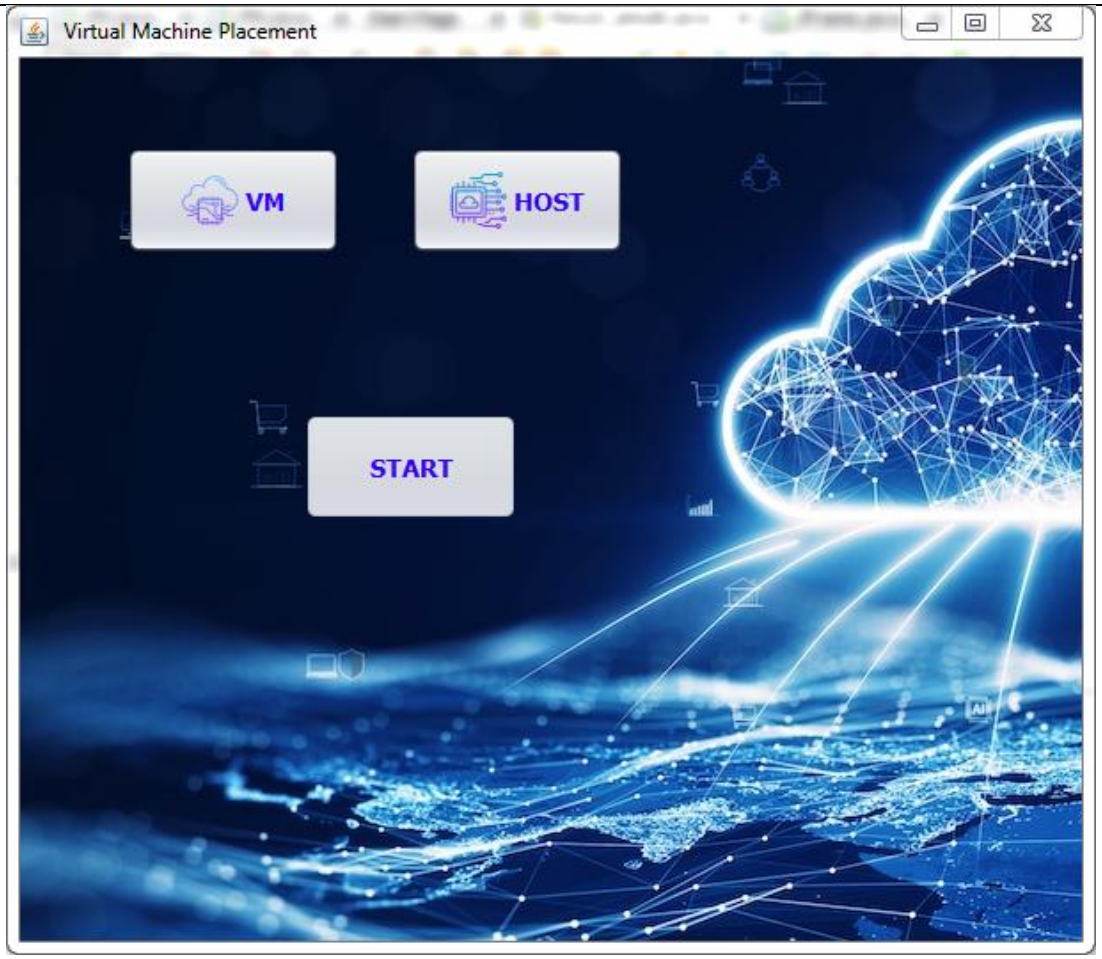

Figure 4.4 : Interface principale.

La première fenêtre sur la figure 4.5 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 :

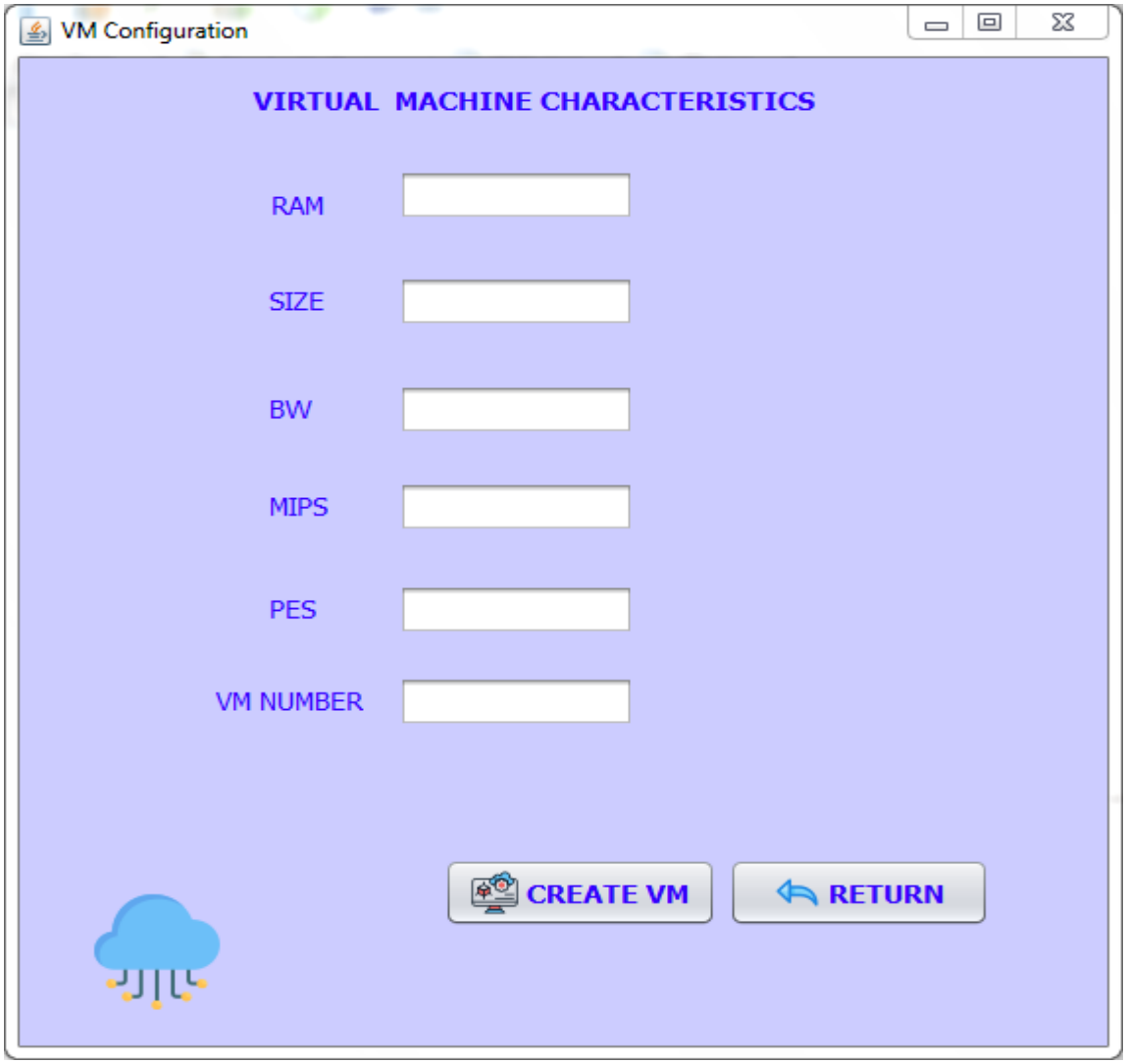

Figure 4.5 : Création des machines virtuelles.

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 passante « 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, un bouton retour (RETURN) se trouve en haut à gauche de la fenêtre.

Configuration des Hôtes :

Figure 4.6: Création des hôtes (Machine physique)

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 passante « 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, un bouton retour (RETURN) 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

Le tableau suivant (tableau 4.1) présente les résultats de notre simulation en utilisant diffèrent tailles de l'espace de recherche (nombre de VMs). On a effectué cette expérimentation à travers des nombres de VMs de 20, 50,100 et 150.

Nombre de VMs	**Solutions initiales**		**Solutions optimales (non dominantes)**	
	Energie(watts)	**SLATAH (%)**	**Energie(watts)**	**SLATAH (%)**
20	1475.0 1242.5 1418.0 1411.99 1409.04 1054.77 1230.96 1054.54 1230.01 1404.41 1228.52 1403.59 1403.28 1403.01 1402.79 1402.59 1227.42 1226.55 1402.19 1402.07	30.48 36.58 36.11 33.53 36.58 33.53 33.53 33.53 36.58 36.58 39.53 39.63 42.68 39.63 33.53 33.53 36.58 33.53 36.58 39.53	1225.01	33.53
50	3330.0 3338.125 3008.25 3175.95 3345.90 3344.14 3340.51 3514.45 3337.20 2808.45 3158.91 3509.51 3333.69 2981.54 2806.19 3155.75 3331.22	26.82 34.14 36.58 35.36 39.02 39.02 32.92 32.92 34.16 36.58 35.36 34.14 36.58 37.80 40.24 36.58 39.82	3150.03	32.92

	3155.58 3155.31 3330.70	37.80 34.14 34.1		
100	9475.02 9388.75 10060.62 9336.89 8618.80 79487.02 9661.09 8779.50 88775.50 9999.79 9645.74 9296.34 9994.62 9291.01 8415.25 8939.12 9460.11 9989.07 9111.90 8936.02	12.54 21.95 28.57 29.61 32.75 33.10 32.05 32.40 32.75 31.01 31.70 33.10 32.40 32.75 32.75 32.75 34.49 36.93 36.23 35.54	8925.09	30.66
150	11937.5 11922.5 11134.5 10929.44 11089.42 10900.37 10721.85 10364.57 10359.62 10882.33 10353.29 10352.63 11051.14 10698.50 10695.45 10787.15 1128.76 11042.60	25.08 31.35 33.79 35.54 33.10 32.40 34.49 34.49 34.14 34.49 35.19 36.23 35.19 35.19 33.79 34.48 35.88 34.14	10675.14 10675.14	32.05 32.05

	10515.67 10864.72	37.97 36.93		

Tableau 4.1. Résultats expérimentaux

Discussion :

D'après le tableau ci-dessus, on remarque une amélioration considérable soit dans l'énergie totale consommée, soit pour le SLATAH par rapport aux solutions initiales obtenues arbitrairement (20 solutions à chaque expérimentation). On note ici, que l'énergie a été amélioré presque vers le minimum, toutefois le SLATAH est amélioré par rapport à la plupart des solutions initiales. Ceci est dû de la nature des deux objectifs étudiés qui sont complètement conflictuels et ceci représente le challenge principal de l'optimisation multi-objectifs.

4.5 Conclusion

Le problème de placement des machines virtuelles est devenu un sujet très important et d'actualité dans le Cloud Computing.

Dans ce chapitre nous avons implémenté l'approche « le récuit simulé Multi-objectif », quelle est notre solution proposée pour résoudre le problème de placement des VM, nous avons utilisé le simulateur CloudSim qui nous a permis d'intégrer notre algorithme, dans l'environnement de développement IDE NetBeans, et bien sur le langage que nous avons utilisé c'est Java. Nous avons présenté toutes les étapes de notre implémentation, et les résultats obtenus avec les deux contraintes consommation des Energie et réduction du sla.

Les résultats de simulation montrent une amélioration considérable entre les deux contraintes.

Conclusion générale

Ce mémoire nous a permis d'apprendre d'une manière détaillée le fonctionnement du cloud computing, la notion de virtualisation et sa relation avec les critères d'énergie et du SLA au sein du cloud computing. Nous avons présenté des formules de consommation d'énergie pour calculer la consommation d'énergie totale dans les data centres Cloud.

Ce travail s'intéresse particulièrement à la problématique de placement des VMs dans le cloud computing. Ce problème est considéré comme étant un problème d'optimisation. Nous prévoyons d'améliorer les performances de notre méthode ont utilisé une méthode de type méta-heuristique qui considérées comme une solution pour de cette problématique. En effet, nous avons opté pour une approche Pareto via un algorithme Récuit simulé multi-objectif (MOSA).

Après l'analyse faite sur les résultats obtenus après l'exécution de notre programme qui simulé et implémenté sur le simulateur CloudSim Plus et le langage de programmation java pour résoudre le problème de placement des machines virtuelle, nous avons obtenu des résultats satisfaisants et logiques qui vérifient les contraintes fixées au préalable ainsi nos fonctions objectives.

Comme perspective, Nous voyons de nombreuses pistes de recherche future dans ce domaine. Nous espérons étendre notre travail et évoluer avec un nombre supérieur à deux fonctions objectives avec plus de contraintes, plus de machines virtuelles et serveurs et avec plusieurs centres de données tout en tenant compte de l'architecture de communication du réseau et de la sécurité dans les traitements. Notre objectif est de continuer à étudier dans ce domaine et de découvrir de nouvelles façons et d'acquérir une certaine expérience pour améliorer ce modèle et le rendre plus efficace afin qu'il devienne si utile en réalité

Références

[1] : MELL, Peter, GRANCE, Tim, et al. The NIST definition of cloud computing. 2011.

[2] : https://www.cisco.com, consulté le 22/02/2023, à 20 h.

[3] : https://www.cigref.fr, consulté le 22/02/2023, à 20 h.

[4] :https://www.tutorialspoint.com/cloud_computing/cloud_computing_overview.htm, consulté le 22/02/2023, à 21 h.

[5] : FIGER, Jean-Paul. Cloud Computing-Informatique en nuage. 2012.

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

[7] : DANNY, AFAHOUNKO. VIRTUALISATION. 2013.

[8] : HASSEN, O. M. R. I. Mise en place d'un Cloud Privé avec gestion centralisée d'hyperviseurs hétérogènes. 2017. PhD Thesis. Université Virtuelle de Tunis.

[9] : ABID, Adnan, et al. Challenges and Issues of Resource Allocation Techniques in Cloud Computing. KSII Transactions on Internet & Information Systems, 2020, 14.7.

[10] : S. François, "La Virtualisation", Projet de recherche et communication scientifique 2010., université libre de Bruxelles, Université d'Europe.

[11] : LYZA, Ammour ; CHANEZ, Chouggar. Allocation de ressource dans le cloud computing. 2017. PhD Thesis. Université Mouloud Mammeri.

[12] : LAPOINTE, Hugo B. Vers la sécurité des conteneurs : les comprendre et les sécuriser. 2022.

[13] : ASHA, N. ; RAO, G. Raghavendra. A review on various resource allocation strategies in cloud computing. International Journal of Emerging Technology and Advanced Engineering (IJETAE), 2013, 3.7.

[14]: KRISHNAVENI, N.; SIVAKUMAR, G. Survey on dynamic resource allocation strategy in cloud computing environment. International Journal of Computer Applications Technology and Research, 2013, 2.6: 731-737.

[15]: TCHANA, Alain-Bouzaïde. Système d'administration autonome adaptable : application au Cloud. 2011. PhD Thesis.

[16]: VIVEK, V.; SRINIVASAN, R.; RAJSINGH, E. B. Resource provisioning methodologies: an approach of producer and consumer favorable in cloud environment. Int J Emerg Technol Adv Eng, 2013, 3.4: 8-13.

[17] : M. A. ALOULOU, Introduction aux problèmes d'ordonnancement, LAMSADE université Paris Dauphine, 2005.

[18]: YASSA, S. *Optimal multi-constraints allocation of workflows in resources of cloud computing environment*. 2014. PhD Thesis. Doctorate thesis, University of Cergy-Pontoise Doctoral school of Sciences and engineering.

[19] : PATEL, Pradip D., et al. Live virtual machine migration techniques in cloud computing: A survey. *International Journal of Computer Applications*, 2014, 86.16.

[20]: KALE, Rinal M. Chawda1 Ompriya. Virtual Machine Migration Techniques in Cloud Environment: A Survey.

[21]: WADHWA, Bharti, et al. iez: Resource contention aware load balancing for large-scale parallel file systems. In: *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2019. p. 610-620.

[22] : DYHIA, Berkani ; KAHINA, Megharbi. *Ordonnancement de workflows dans le cloud*. 2017. PhD Thesis. Université Mouloud Mammeri.

[23] : BECHIR, AZIZA. *RESOLUTION DE PROBLEMES D'OPTIMISATION PAR LES SYSTEMES MULTI-AGENTS ET LES APPROCHES EVOLUTIONNAIRES*. 2016. PhD Thesis. Université Mohamed Khider-Biskra.

[24] : LAYEB, Abdesslem ; SAIDOUNI, D. E. Utilisation des approches d'optimisation combinatoire pour la vérification des applications temps réel. 2017.

[25] : Karloff, Howard. Linear Programming. Springer Science & Business Media, 2008.

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

[27] : DOUIRI, SIDI MOHAMED ; ELBERNOUSSI, SOUAD ; LAKHBAB, HALIMA. Université Mohammed v, rabat. «. *Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques*.

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

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

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

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

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

[33]: 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, October 2010.

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

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

[36]: 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).

[37]: RAMADAN, Hashem H.; KASHYAP, D. Quality of service (QoS) in cloud computing. International Journal of Computer Science and Information Technologies (IJCSIT), 2017, 8.3: 318-320.

[38] : BESSAI, Kahina. *Gestion optimale de l'allocation des ressources pour l'exécution des processus dans le cadre du Cloud*. 2014. PhD Thesis. Université Paris1 Panthéon-Sorbonne.

[39] : aut. Autin Baptiste / Conservatoire National Des Arts et Metiers. - PARIS : [s.n.], 2006.

[40]: Aarts, E., & Korst, J. (1989). Simulated annealing and Boltzmann machines. Wiley.

[41]: Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671-680.

[42]: KIRKPATRICK, S., GELATT, C. D., VECCHI, JR. (May 1983) Optimization by Simulated annealing, Science, vol. 220, no 4598, p. 671-680

[43] : HEMMAK Allaoua Support de cours d'optimisation combinatoire. Juin 2017

[44] : Amir Nakib Conception de méta heuristiques d'optimisation pour la segmentation d'images. Application aux images biomédicales

[45] : https://rfia2012.files.wordpress.com/2011/12/amine_le_recuit_simulc3a9.pdf, consulté le 07/05/2023, à 10h.

[46]: 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.

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

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

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

[50]: E. L. Ulungu, J. Teghem, P. H. Fortemps, and D. Tuyttens, "MOSA method: a tool for solving multiobjective combinatorial optimization problems," Journal of Multi-Criteria Decision Analysis, vol. 8, no. 4, pp. 221–236, 1999

[51]: P. Serafni, “Simulated annealing for multi objective optimization problems,” in Multiple Criteria Decision Making: Proceedings of the Tenth International Conference: Expand and Enrich the Domains of Tinking and Application, G. H. Tzeng, H. F. Wang, U. P. Wen, and P. L. Yu, Eds., pp. 283–292, SpringerVerlag, New York, NY, USA, 1

[52] : 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. MiraBejaïa 2013.

[53] : Silva Filho, Manoel C., et al. "CloudSim Plus: A Cloud Computing 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.

[54] : Livre blanc rédigé par le Groupe cloud computing de l’ADIRA, 38cours Eugénie 69003 Lyon « cloud computing ».

[55] : COURAUD, Nicolas 15 rue vignon 75008 PARIS « CLOUD COMPUTING Définition &Concepts, Enquête et analyse des Tendances ».

[56]: Khoa Dang Pham, Embedded Virtualization of a Hybrid ARM-FPGA Computing Plateforme ,2014

[57] : https://www.piloter.org/techno/support/virtualisation.htm, consulté le 29/05/2023, à 19h

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

[59] : Optimisation Multi-Objectifs des paramètres d'usinage par Algorithme Génétiques -scientific Figure on ResearchGate.Available from : https://www.researchgate.net/figure/Principe-de-fonction-dun-algorithme-genetique , consulté le 01/06/2023, à 16 h.

[60]: P. Kora et P. Yadlapalli, « Crossover operators in genetic algorithms: A review», Int.J. Comput.Appl., vol.162, no 10,2017

[61] : WWW.cloudbus.org , consulté le 01/06/2023, à 16 h.