République Algérienne Démocratique et Populaire Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 20 août 1955 - Skikda

Faculté des Sciences

Département d'informatique

Mémoire de Fin d'Études

Pour l'obtention du diplôme de Master en Informatique

Option : Système informatique.

Thème

Algorithme génétique pour le placement des machines virtuelles dans les centres de données Cloud

Réalisé par :

- **Ali Guechi Nardjes.**
- **Kedissa Aya.**

Encadré par:

- **Laouar Walid.**
- **Bouaita Riad.**

Session juin 2022

Remerciement

Nous remercions Allah qui nous a aidé et donné le courage, la volonté pour la réalisation de ce PFE. Tout d'abord, nous tenons à exprimer nos gratitude à Mr LAOUAR Walid en tant que encadrant de ce PFE et Mr BOUAITA Riad en tant que co-encadrant. Ils nous ont permis de réaliser ce travail sous leurs directions et aussi pour leurs conseils, orientations durant toute la réalisation de ce travail. Nous remercions les membres du jury pour avoir accepté d'examiner et évaluer notre PFE. Nos remerciements également à tous nos enseignants durant les années de nos études. Enfin, nous tenons à remercier toute personne qui nous ont aidé et encouragé le long de ce travail.

Dédicace

Nous dédions ce mémoire :

A Nos très chers parents qui ont toujours nous ont soutenus et encouragés durant toutes nos années d'études,

A nos frères, nos sœurs, A nos familles et tous nos amis,

A tous qui ont aidé pour réaliser ce travail,

Nous vous disons merci.

Résumé

Avec la demande croissante des services Cloud Computing, les fournisseurs Cloud doivent proposer des solutions adaptant des méthodes et des mécanismes qui augmentent les performances et assurent la disponibilité et la fiabilité des centres de données et des systèmes Cloud. La virtualisation des serveurs est un élément clé pour y parvenir, qui permet de partager les ressources d'une machine physique entre plusieurs machines virtuelles de manière totalement isolée. L'optimisation de la virtualisation a un effet très significatif sur les performances globales des systèmes Cloud. Cela nécessite un placement efficace et efficient des machines virtuelles dans les machines physiques. Comme il s'agit d'un problème d'optimisation de complexité NP-Complet et impliquant de multiples contraintes, nous proposons une méthode basée sur les algorithmes génétiques pour placer des machines virtuelles dans le Cloud. En considérant l'utilisation des machines, notre méthode vise à réduire la consommation d'énergie dans les différentes hôtes des centres de données. Les résultats de simulation montrent l'efficacité de l'algorithme utilisé par rapport à d'autres méthodes (Aléatoire, Round Robin) en termes d'énergie consommée. Cette simulation a été mise en œuvre en utilisant le simulateur CloudSim Plus, un simulateur Cloud le plus populaire et le plus accessible au public.

Mots Clé : Cloud Computing, Datacenter, machine virtuelle, efficacité énergétique, algorithme génétique.

الملخص

مع الطلب المتزايد على خدمات الحوسبة السحابية ، يحتاج مقدمو الخدمات السحابية إلى تقديم حلول تكيف الأساليب والآليات التي تزيد من الأداء وتضمن توافر وموثوقية مراكز البيانات والأنظمة السحابية. تعد المحاكاة الافتراضية للخادم عنصرًا أساسيًا لتحقيق ذلك ، مما يسمح بمشاركة موارد الجهاز المادي بين العديد من الأجهزة الافتراضية بطريقة معزولة تمامًا. التحسين الافتراضي له تأثير كبير جدًا على الأداء العام للأنظمة السحابية. يتطلب هذا توزيعا فعالًا للأجهزة الافتراضية داخل الأجهزة المادية. نظرًا لأن هذه مشكلة تحسين ذات تعقيد NP-Hard وتتضمن قيودًا متعددة ، فإننا نقترح طريقة تعتمد على الخوارزميات الجينية لوضع الأجهزة الافتراضية في السحابة. من خلال التفكير في استغلال الآلات ، تهدف طريقتنا إلى تقليل استهلاك الطاقة في المضيفين المختلفين لمراكز البيانات. أظهرت نتائج المحاكاة كفاءة الخوارزمية المستخدمة مقارنة بالطرق الأخرى (العشوائية ، والتلدين المحاكي) من حيث الطاقة المستهلكة. تم تنفيذ هذه المحاكاة باستخدام محاكي CloudSim Plus، المحاكي السحابي الأكثر شيوعًا والمتاح للجمهور.

الكلمات المفتاحية: الحوسبة السحابية ، مركز البيانات ، الآلة الافتراضية ، كفاءة الطاقة ، الخوارزمية الجينية.

Abstract

With the growing demand for Cloud Computing services, Cloud providers need to offer solutions that adapt methods and mechanisms that increase performance and ensure the availability and reliability of datacenters and Cloud systems. Server virtualization is a key element to achieve this, which allows the resources of a physical machine to be shared between several virtual machines in a completely isolated manner. Virtualization optimization has a very significant effect on the overall performance of Cloud systems. This requires effective and efficient placement of virtual machines within physical machines. As this is an optimization problem of NP-Hard complexity and involving multiple constraints, we propose a method based on genetic algorithms to place virtual machines in the Cloud. By considering the utilization of the machines, our method aims to reduce the energy consumption in the different hosts of the data centers. The simulation results show the efficiency of the algorithm used compared to other methods (Random, Round Robin) in terms of energy consumed. This simulation was implemented using the CloudSim Plus simulator, the most popular and publicly available Cloud simulator.

Keywords: Cloud Computing, Datacenter, virtual machine, energy efficiency, genetic algorithm.

SOMMAIRE

Liste des figures

Liste des tableaux

Acronymes

ACO	**Ant Colony Optimization**	**OX**	**Crossover Order**
AE	Algorithme Évolutionnaires	**PAAS**	Plate-form as a Service
AG	Algorithme Génétique	**PDG**	President Director General
AWS	Amazon Web Services	PE	Programmation Évolutionnaire
BAAS	Backend as a service	PES	Processor Cores
CPU	Central Processing Unit	PG	Programmation Génétique
CX	Cycle Crossover	PL	Programmation Linéaire
DAAS	Data as a Service	PMX	Partially Mapped Crossover
DES	Discrete Event Simulation	POC	Problèmes D'optimisation Combinatoire
DRY	Do not Repeat Yourself	PSO	particle swarm optimization
EC2	Elastic Cloud Computing	RAM	Random Access Memory
ERP	Enterprise Resource Planning	**REST**	Restful Web Services
GA	Genetic Algorithm	RLP	restricted Linear Programming
IAAS	Infrastructure as a Service	**SAAS**	Software as a Service
IBM	International Business Machines Corporation	SDK	software development kits
IDC	International Data Corporation	**SECAAS**	Security as a Service
IHM	Interface Homme machine	**SLA**	Customer Service Level
IP	Internet Protocol	**SOAP**	Simple Object Access Protocol
KISS	keep it simple, stupid	SOC	Separation of Concerns
MBAAS	Mobile Backend as a Service	**STAAS**	Storage as a Service
MP	Machine Physique	TCP	Transmission Control Protocol
MV	Machine Virtuelle	**TEAAS**	Test Environment as a Service

NIST	National Institute of Standards and Technology	TS	Tabu Search
NP	Nondeterministic Polynomial	UDP	User Datagram Protocol
NS	Network Simulator	UML	Unified Modeling Language
OEP	L'optimisation par Essaim de Particules	**VPN**	Virtual Private network
OS	Operating System		

Introduction Générale

Le Cloud Computing (informatique dans les nuages) est une notion propagée ces dernières années dans le monde de l'informatique et s'est imposé comme un paradigme majeur d'utilisation des ressources informatiques. Ces ressources mises à disposition par le fournisseur Cloud sont accessibles via un réseau informatique sous forme de services.

En raison de la demande croissante de services Cloud, l'optimisation de la consommation des ressources devient encore plus essentielle pour augmenter les performances, la disponibilité et la fiabilité des systèmes Cloud. Les technologies de virtualisation se sont avérées très utiles pour répondre à ces exigences. La virtualisation permet aux utilisateurs et aux applications de partager les ressources physiques du Cloud de manière efficace, efficiente et isolée. Avec la virtualisation des serveurs, plusieurs machines virtuelles peuvent fonctionner sur une seule machine physique de manière totalement isolée. De cette manière, les fournisseurs Cloud peuvent servir un plus grand nombre de clients de manière flexible et efficace. Différentes machines virtuelles demandées par les utilisateurs peuvent avoir des exigences différentes en matière de traitement, de mémoire, d'E/S et de mise en réseau. Les serveurs physiques peuvent également avoir des capacités différentes. Cela conduit à un problème d'optimisation connu sous le nom de problème de placement de machines virtuelles. Une solution à ce problème peut viser à augmenter l'utilisation des machines physiques pour réduire la consommation d'énergie. A cet effet, l'optimisation de l'utilisation des ressources devient un problème essentiel pour plus économiser d'énergie consommée, et respecter les accords de niveau de service client (SLA).

Le problème de placement de machines virtuelles est un problème NP-complet multi-objectifs contraint. Les approches basées sur des algorithmes génétiques ont été l'une des méthodes les plus largement utilisées pour résoudre ce type de problèmes complexes. L'algorithme génétique imite le processus de sélection naturelle et tente ainsi de trouver une solution proche de l'optimum à un problème complexe donné.

Dans ce travail, nous proposons un modèle de placement de machines virtuelles en minimisant la consommation d'énergie à l'aide d'une métaheuristique : algorithmes génétiques. Ce mémoire est subdivisé en quatre principaux chapitres.

Dans le premier chapitre, nous présentons les concepts généraux liés au Cloud Computing, son évolution, ses caractéristiques, ses modèles de déploiements et de services.

Dans le deuxième chapitre, nous avons essayé de présenter les différentes méthodes d'optimisation combinatoires à savoir les méthodes exactes, les heuristiques et les métaheuristiques en présentant leurs principales caractéristiques. À la fin du chapitre nous présentons quelques métaheuristiques et nous annonçons les différences entre Les méthodes d'optimisation.

Au niveau du Troisième chapitre, les concepts de l'algorithme génétique sont présentés d'une manière détaillée. Nous détaillons ensuite l'algorithme proposé pour résoudre le problème de placement des machines virtuelles dans le Cloud. Nous décrivons également la formulation du problème étudié, les différents paramètres de l'algorithme (sélection, croisement, et mutation), ses contraintes ainsi que le modèle de la fonction objectif constituant la minimisation de la consommation énergétique dans les centres de données.

Le dernier chapitre se préoccupe à la concrétisation de la solution proposée via une implémentation de celle-ci. Nous décrivons en particulier l'environnement de développement, le simulateur CloudSim Plus et le langage de programmation utilisé. Nous présentons ensuite les résultats expérimentaux validant l'algorithme proposé à travers la métrique de l'énergie consommé.

Chapitre 1 : Généralités Sur Le Cloud Computing

1.1. Introduction

Le cloud computing est un concept relativement nouveau car il est en cours de transformation. La puissance informatique devient ainsi virtuelle et se consomme aux besoins des clients et devient extensible. C'est une technologie omniprésente qui offre de nombreux services et logiciels en plus du stockage des systèmes clous peuvent fournir presque tous les types de services ont besoin où est les clients peuvent demander et éviter de construire leur propre infrastructure physique.

Ce chapitre présente l'histoire du Cloud Computing, une analyse des définitions proposées par le monde académique. Les acteurs, les caractéristiques, les modèles de déploiement et de service du Cloud sont ainsi présentés. Nous décrivons également la technique de virtualisation et les hyperviseurs ainsi que les autres technologies liés au concept du Cloud Computing.

1.2. Le Cloud Computing

1.2.1. L'Histoire Du Cloud Computing

Le concept sous-jacent du Cloud Computing a été introduit retour dans les années :

- 1959 : Note de John McCarthy sur la nécessité de temps partagé des ordinateurs.
- 1961 : Conférence de John McCarthy suggérant que les ordinateurs devenir un utilitaire similaire à un téléphone service.
- 1966 : Douglas Parkhill publie un livre intitulé *Challenges of the Computer Utility* [1]. Les entreprises ont commencé à proposer un réseau privé virtuel (VPN) avec une qualité de service comparable à un prix bien inférieur de coût. Initialement avant le VPN, elles fournissaient un point à point dédié circuits de données, ce qui était un gaspillage de bande passante. Mais, en utilisant les services VPN, ils peuvent basculer le trafic vers l'équilibre d'utilisation du réseau global. Le Cloud Computing maintenant étend cela pour couvrir les serveurs et l'infrastructure réseau. De nombreux acteurs du secteur se sont lancés dans le Cloud informatique et l'ont mis en œuvre [2].
- 1995 : Amazon commence à vendre des livres sur le World Wide Web.
- 1999 : Salesforce.com propose un service logiciel sur le World Wide Web accessible moyennant paiement.
- 1999 : Ian Foster et Carl Kesselman publient un livre intitulé *The Grid: Blueprint for a new Computing infrastructure,* et développent le kit d'outils Global pour créer une grille informatique.
- 2004 : Google lance un service de messagerie gratuit.
- 2006 : Amazon a joué un rôle clé et a lancé l'informatique payante : Amazon Web Services (AWS) et l'Elastic Cloud Computing (EC2).

- 2006 : Google commence à proposer Google Apps avec 2 Go d'espace disque libre sur son infrastructure.
- 2010 : Microsoft commence à fournir un service Cloud appelé Azure.
- 2011 : Microsoft propose un Cloud intelligent [1].

1.2.2. L'origine du terme "Cloud Computing"

Une recherche Google de « Cloud Computing » a donné 72 500 000 visites « le 3 octobre 2013 ». C'est évidemment l'un des plus chauds mots à la mode en informatique. C'est une métaphore pour les ordinateurs connectés à Internet qui sont généralement enfermés dans une frontière semblable à un nuage. Les termes sont devenus populaires après avoir été utilisés par Éric Schmidt (PDG de Google) en 2006 lors d'une conférence de l'industrie. Une société appelée Net Centri (disparue aujourd'hui) a déposé en 1999 une marque de « Cloud Computing » pour des services éducatifs, Elle n'a pas été approuvée, Selon une première entrée de Wikipédia (supprimée depuis). il a été utilisé pour la première fois dans une publication académique par Ramnath Chellappa du Département de gestion de l'Université du Texas, Austin en 1997[1].

1.2.3. Définitions

Le terme Cloud Computing a été défini de plusieurs manières par les analystes, les universitaires, les praticiens de l'industrie et les sociétés informatiques. Le tableau (1.1) montre comment les analystes distingués définissent ou décrivent le Cloud Computing.

1.2.3.1. Définition de Gartner

un style d'informatique dans lequel des capacités liées à l'informatique massivement évolutives sont fournis « en tant que service » à l'aide des technologies Internet à de multiples clients »[3].

1.2.3.2. Définition de IDC

un modèle émergent de développement, de déploiement et de livraison informatique, permettant la livraison en temps réel de produits, services et solutions sur Internet (c.-à-d. services Cloud) »[4].

1.2.3.3. Définition de Groupe 451

Le Cloud est l'informatique en tant que service, et fourni par des ressources informatiques indépendantes de l'emplacement[5].

1.2.3.4. Définition de Merrill Lynch

l'idée de fournir des informations personnelles (par exemple, e-mail, traitement de texte, présentations.) et applications de productivité d'entreprise (par exemple, automatisation de la force de vente, service, comptabilité) à partir de serveurs centralisés » [4].

1.2.3.5. Définition de NIST

Le Cloud Computing est un modèle permettant un accès omniprésent et pratique à un pool partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications, accès réseau et services à la demande) qui peuvent être rapidement provisionnés et libérés avec un minimum d'effort administratif ou d'interaction avec les fournisseurs de services [6].

Toutes ces définitions ont une caractéristique commune : elles tentent de décrire et de définir le Cloud Computing du point de vue des utilisateurs finaux et leur accent est mis sur la façon dont cela pourrait être vécu par eux. Selon ces définitions, la caractéristique principale de Cloud Computing est la fourniture d'une infrastructure informatique et d'applications en tant que service de manière évolutive [4].

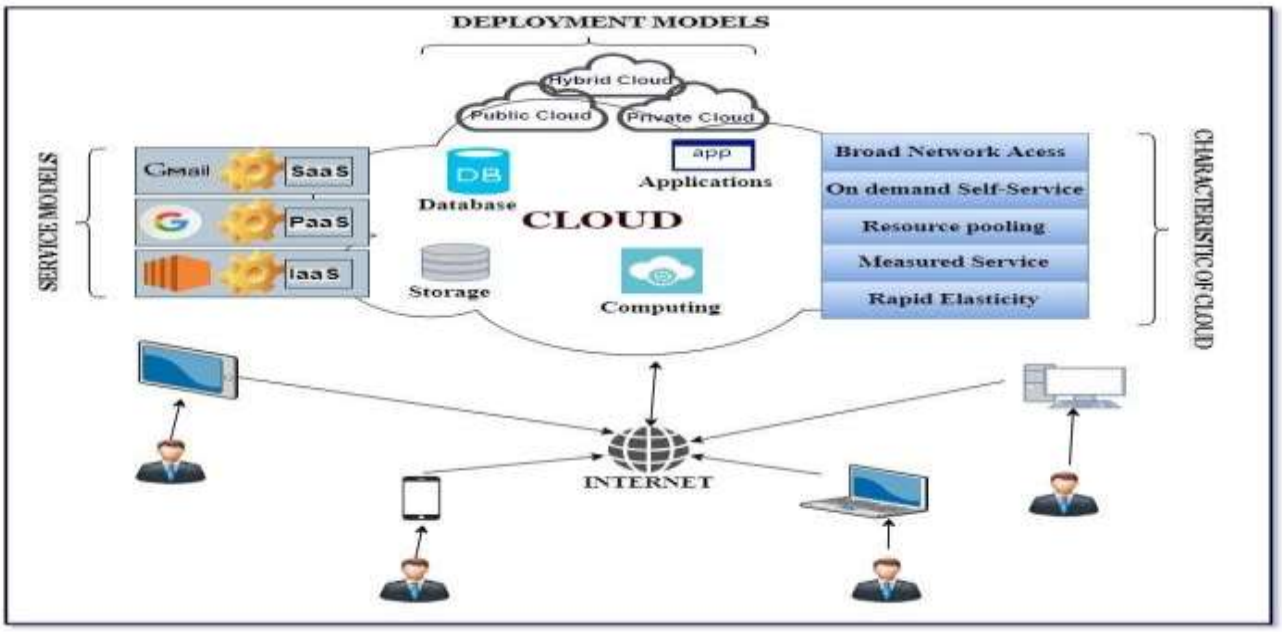


Figure 1.1 Vue générale de l'environnement Cloud Computing[4].

1.2.4. Les Caractéristiques principales du Cloud Computing

- Moins de compétences informatiques sont requises pour la mise en œuvre.
- Un service fiable peut être obtenu en utilisant plusieurs sites adaptés à la continuité des activités et reprise après sinistre.
- La maintenance est plus facile en cas de Cloud applications car elles n'ont pas besoin d'être installées sur chaque utilisateur ordinateur.
- La fonction de paiement à l'utilisation (*pay per use*) permet de mesurer l'utilisation d'application par client sur des bases régulières [2].

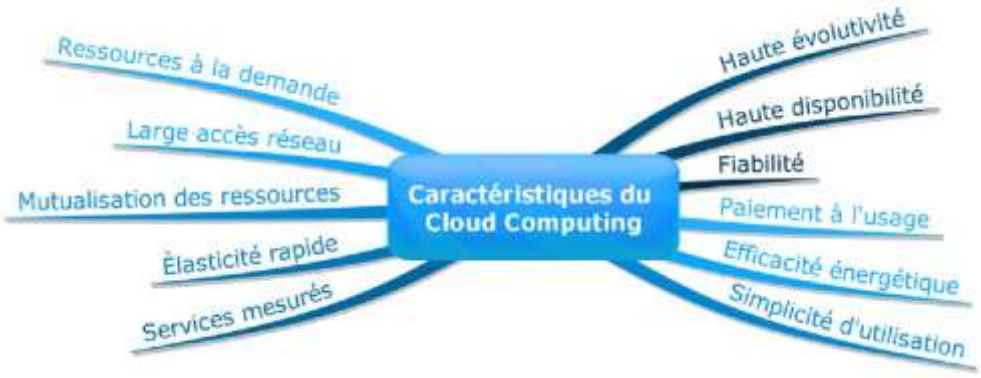


Figure 2.2 Caractéristiques du Cloud Computing [4]

1.2.5. Les Acteurs du Cloud Computing

On distingue les acteurs de Cloud suivant[7] :

1.2.5.1. Consommateur de Cloud (Cloud Consumer)

Est le principal acteur du service de Cloud. Un consommateur de Cloud représente une personne ou une organisation qui entretient une relation commerciale avec un fournisseur de Cloud et utilise le service d'un fournisseur. Un consommateur parcourt le catalogue de services d'un fournisseur, demande le, établit des contrats de ce service avec lui et le utilise. Le consommateur peut être facturé pour le service fourni et doit organiser les paiements en conséquence.

1.2.5.2. Fournisseur Cloud (Cloud provider)

Un fournisseur Cloud peut être une personne ou une organisation. C'est l'entité chargée de mettre un service à la disposition des parties intéressées. Un fournisseur acquiert et gère l'infrastructure informatique requise pour fournir les services, exécute le logiciel de Cloud qui fournit les services et prend des dispositions pour les fournir aux consommateurs de Cloud via un accès réseau.

1.2.5.3. Auditeur du Cloud (Cloud Auditor)

Un auditeur Cloud est une partie qui peut effectuer un examen indépendant des contrôles des services Cloud dans le but d'exprimer une opinion à ce sujet. Des audits sont effectués pour vérifier la conformité aux normes par l'examen des preuves objectives. Un auditeur de Cloud peut évaluer les services fournis par un fournisseur Cloud en termes de contrôles de sécurité, d'impact sur la confidentialité, de performances, etc.

1.2.5.4. Courtier du Cloud (Cloud broker)

À mesure que le Cloud évolue, l'intégration des services Cloud peut être trop complexe à gérer pour les consommateurs. Au lieu de contacter directement le fournisseur de Cloud, les consommateurs peuvent demander des services Cloud à des courtiers Cloud. Un courtier est une entité qui gère l'utilisation,

les performances et la fourniture de services Cloud et les relations de courtage entre les fournisseurs et les consommateurs.

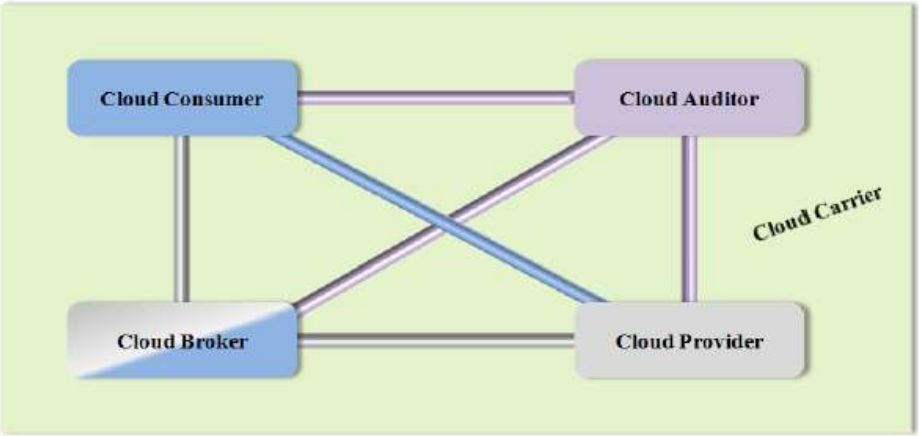


Figure 3.3 Interactions entre les acteurs du Cloud Computing [7].

: Le chemin de communication entre un fournisseur et un consommateur de Cloud.

: Les chemins de communication permettant à un auditeur Cloud de collecter des informations d'audit.

: Les chemins de communication permettant à un courtier Cloud de fournir un service à un consommateur Cloud.

1.3. Les concepts connexes au Cloud Computing

1.3.1. Grid Computing

L'informatique en grille peut signifier différentes choses pour différentes personnes. Les utilisateurs omniprésents et individuels (clientes d'applications) accèdent aux ressources informatiques (processeurs, stockage, applications de données, etc..) selon les besoins avec peu ou pas de connaissance de l'emplacement de ces ressources ou des technologies, du matériel et du système d'exploitation sous-jacents. L'informatique en grille peut être considérée comme un voyage sur la voie de l'intégration de diverses technologies et solutions qui nous rapprochent de l'objectif final. Ses principales valeurs résident dans les technologies d'infrastructure informatique distribuée sous-jacentes qui évoluent pour prendre en charge le partage d'applications et de ressources inter-organisationnelles [8].

Le Grid Computing permet de partager et d'unifier des réseaux hétérogènes de ressources informatiques. C'est le point de départ et la base du Cloud Computing.

Le Cloud Computing représente essentiellement la tendance croissante vers l'externe du déploiement de ressources informatiques, telles que la puissance de calcul, le stockage ou les applications métiers, et de les obtenir en tant que services.

1.4. La virtualisation

1.4.1. La Définition de la virtualisation

La virtualisation consiste essentiellement à créer une image virtuelle ou une "version" de quelque chose comme un serveur, un système d'exploitation, des périphériques de stockage ou des ressources réseau afin qu'ils puissent être utilisés sur plusieurs machines en même temps. L'objectif principal de la virtualisation est de gérer la charge de travail en transformant l'informatique traditionnelle pour la rendre plus évolutif, efficace et économique [9].

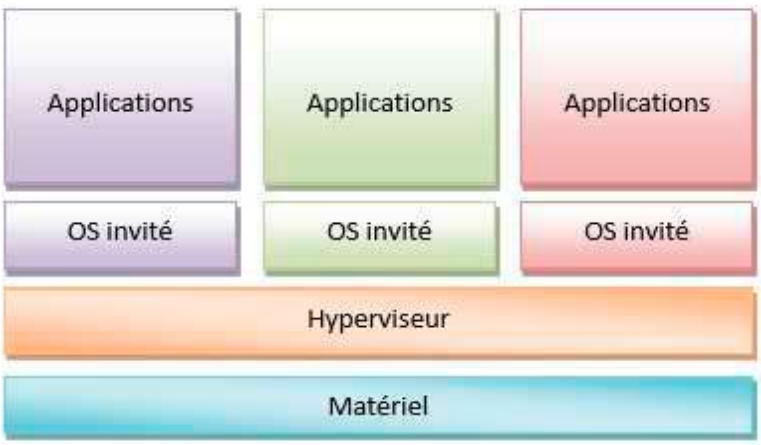


Figure 4.4. Les différentes couches d'un serveur virtualisé[9].

1.4.1.1. Intérêt de la Virtualisation

À l'aide de la virtualisation, nous pouvons augmenter l'utilisation des ressources à notre disposition pour obtenir plus d'avantages. Nous devrions virtualisé pour les raisons suivantes [10]:

a. Isolement entre les utilisateurs : un utilisateur doit être isolé des autres utilisateurs afin qu'il ne puisse pas avoir de ces informations sur les données et l'utilisation des autres utilisateurs et ne peuvent même pas accéder aux données des autres.

b. Partage de ressources : une ressource peut être fragmentée en plusieurs ressources virtuelles afin de pouvoir être utilisé par plusieurs utilisateurs utilisant la technique de virtualisation.

c. Ressources dynamiques : la réallocation des ressources telles que les ressources de stockage et de calcul est très difficile, mais avec la virtualisation, elles peuvent être facilement réaffectées.

d. Agrégation des ressources : les petites ressources disponibles peuvent être augmentées dans une large mesure à l'aide de la virtualisation.

1.4.1.2. Les Types de virtualisation

a. virtualisation du stockage : Le stockage disponible est virtualisé pour obtenir un accès important au stockage virtuel et il est utilisé en outre pour allouer la mémoire aux clients Cloud.

b. Virtualisation logicielle : Les logiciels construits par la société peuvent être utilisés par un grand nombre de systèmes au même temps à l'aide de la virtualisation. Une couche virtuelle est créée sur laquelle le logiciel est installé et utilisé [9].

1.4.2. Les hyperviseurs

Un hyperviseur est une plate-forme de virtualisation qui permet aux plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps [11].

a. hyperviseur de type 1 : ou natif, est un logiciel qui s'exécute directement sur une plateforme matérielle. Cette plateforme est alors considérée comme un outil de contrôle de système d'exploitation. Un système d'exploitation secondaire peut, de ce fait, être exécuté au-dessus du matériel. C'est un noyau hôte allégé et optimisé pour ne faire tourner initialement que des noyaux de systèmes d'exploitation invités adaptés et optimisés à cette architecture spécifique. Ces systèmes invités ayant "conscience" d'être virtualisé [11].

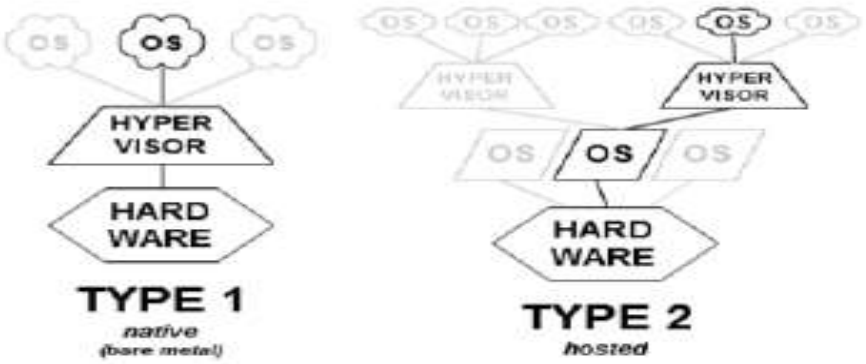


Figure 5.5 Les type des hyperviseurs[11].

b. hyperviseur de type 2 : C'est un logiciel qui s'exécute à l'intérieur d'un autre système d'exploitation. Un système d'exploitation invité s'exécutera donc en troisième niveau au-dessus du matériel. Les systèmes d'exploitation invités n'ayant pas conscience d'être virtualisés et n'ont pas besoin d'être adaptés[11].

1.5. L'Architecture d'un système Cloud

De nombreuses organisations et chercheurs ont défini l'architecture du Cloud Computing. Fondamentalement, l'ensemble du système peut être divisé en : *pile centrale* et *la gestion* (*management*). Dans la pile principale, il y a trois couches : (1) Ressource (2) Plate-forme et (3) Application. La couche ressources est la couche d'infrastructure qui est composée de ressources informatiques, de stockage et de mise en réseaux physiques et virtualisées. La couche plate-forme est la partie la plus complexe qui pourrait être divisée en plusieurs sous-couches. Par exemple, Une infrastructure informatique gère la répartition des transactions et/ou la planification des tâches. Une sous-couche de stockage offre une capacité de stockage et de mise en cache illimitée. Le serveur d'applications et les autres composants prennent en charge la même logique d'application générale qu'auparavant avec une capacité à la demande ou une gestion flexible, de sorte qu'aucun composant ne sera le goulot d'étranglement de l'ensemble du système[12].

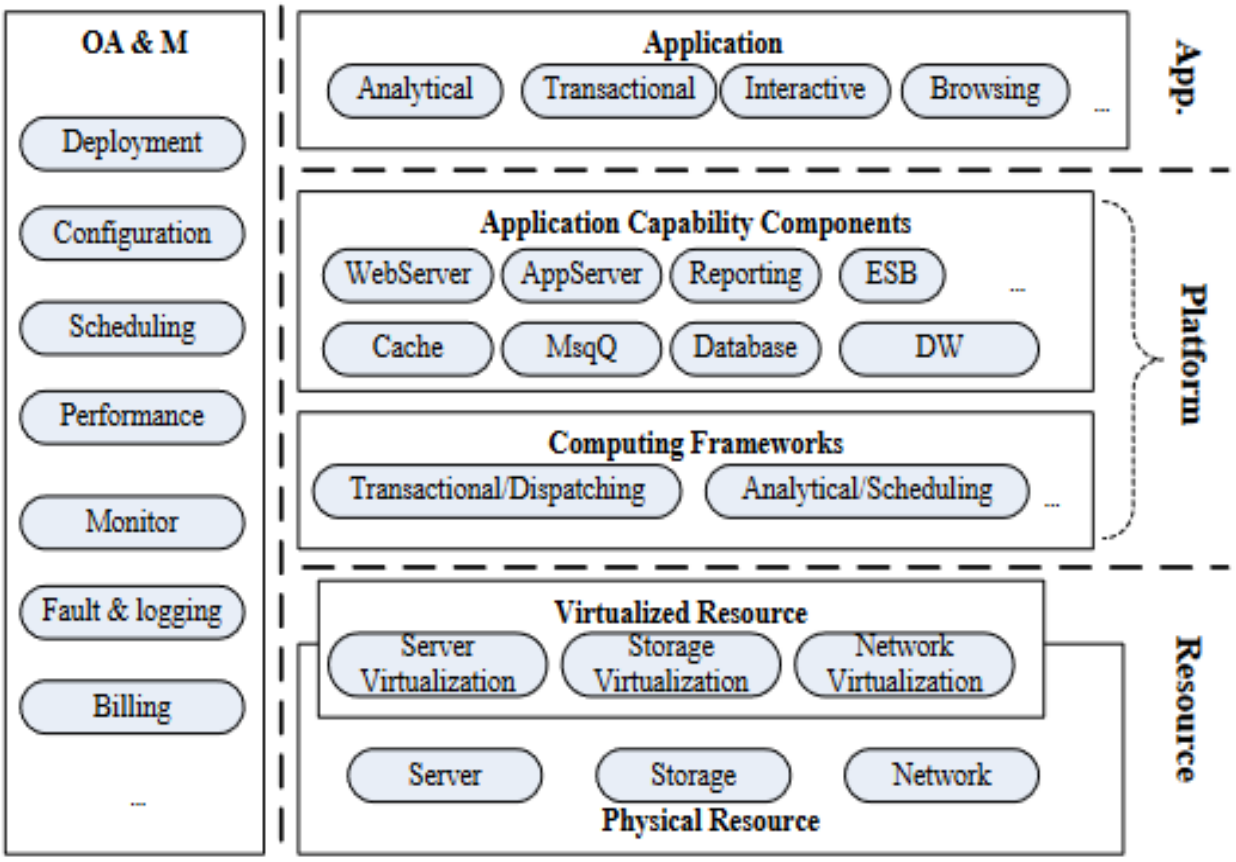


Figure 6.6 L'architecture de référence[12].

Sur la base de la ressource et des composants sous-jacents, l'application pourrait prendre en charge des transactions importantes et distribuées et la gestion d'un énorme volume de données. Toutes les couches fournissent un service externe via un service Web ou d'autres interfaces ouvertes

1.5.1. Les modèles de services du Cloud Computing

Les définitions du Cloud Computing comprennent différentes capacités informatiques, à savoir l'infrastructure, les plates-formes et les logiciels. Cela peut également être appelé différentes « formes », « segments », « styles », « types », « modèle»,« niveaux » ou « couches » du Cloud Computing. Quel que soit le terme utilisé, cette triple classification du Cloud s'est banalisée[4].

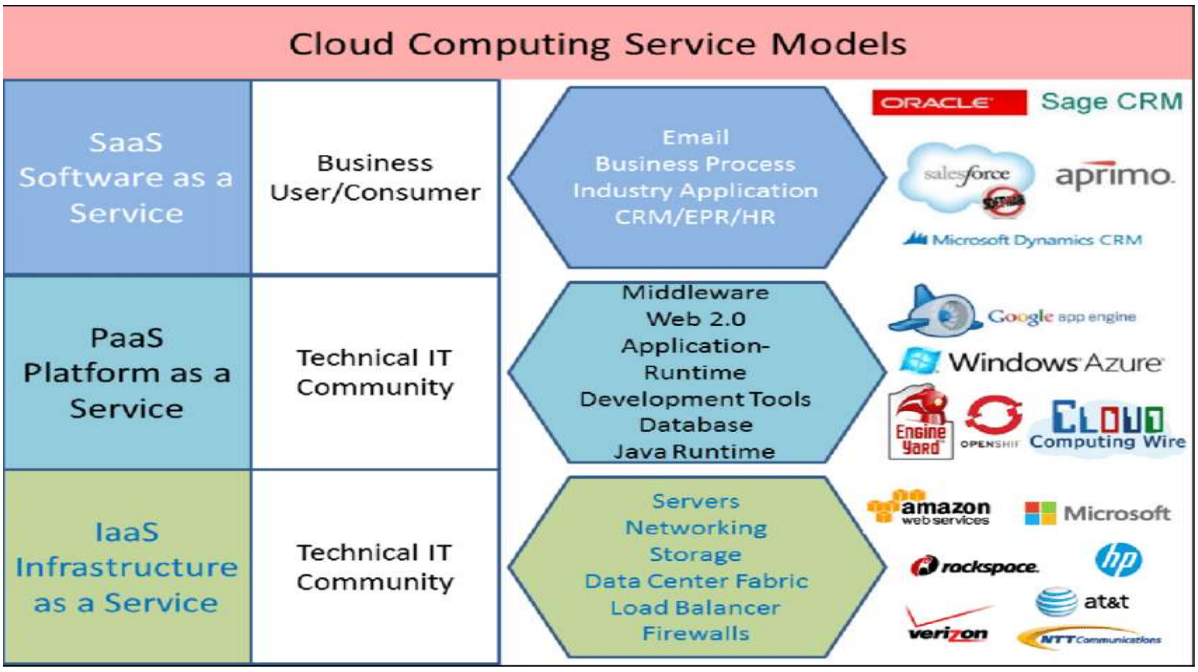


Figure 7.7 Modèle de service d'informatique en Cloud [4].

- ***Logiciel en tant que Service (SAAS)***

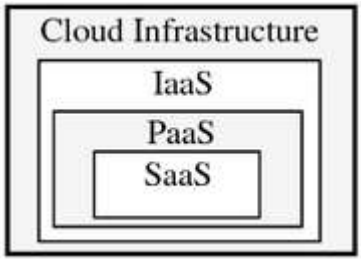


Figure 8.8 Architecture Logiciel en tant que Services (SaaS).

Dans ce cas, un fournisseur de logiciels crée un logiciel d'application qui s'exécute sur l'infrastructure installée par un fournisseur IaaS ou sur serveurs gérés par le fournisseur de l'application qui sont connectés à l'Internet [4].en d'autres termes, les consommateurs peuvent utiliser des applications en cours d'exécution sur un IaaS ou un PaaS via une interface unique [1].Google Apps est le SaaS le plus utilisé Ce logiciel d'application peut être utilisé simultanément par de nombreux clients (on dit qu'il s'agit d'un multi locataire maquette). L'application est accessible par un client à partir d'Internet à l'aide d'un dispositif d'accès approprié. Éliminant ainsi la nécessité d'installer et exécuter l'application sur le système de l'utilisateur [4].

- ***Plate-forme en tant que Service (PAAS)***

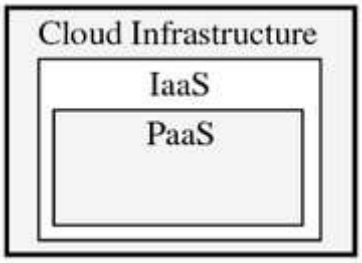


Figure 9.9 Plateforme en tant que service(PaaS).

Fournissent une plate-forme informatique utilisant l'infrastructure Cloud. Ce service est construit sur IaaS. Dans ce type de service le fournisseur maintient outre l'infrastructure matérielle et l'infrastructure logicielle comme un système d'exploitation, des langages de programmation, et des outils de développement et de déploiement de programmes d'application[1]. Les consommateurs peuvent développer. Dans d'autres termes, l'application généralement requise par le client déployé sur celle-ci. Ainsi le client n'a pas besoin de passer par les tracas de l'achat et l'installation du logiciel et du matériel nécessaires à cet effet. [13].Par les développeurs de ce service peuvent mettre la main sur tous les systèmes et environnements nécessaires au cycle de vie des logiciels, qu'il s'agisse développement, test, déploiement et hébergement d'applications Web [4].

- ***Infrastructure en tant que Service (IAAS)***

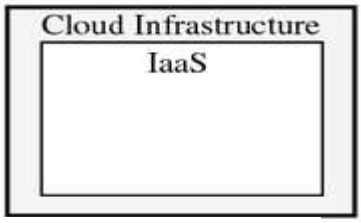


Figure 10.10 l'infrastructure en tant que service (Saas).

Fournit l'infrastructure requise en tant que service. Le client pas besoin d'acheter les serveurs, le centre de données ou les ressources du réseau. Aussi l'avantage clé ici est que les clients doivent payer uniquement pour la durée d'utilisation du service. Les consommateurs peuvent accéder aux ressources du Cloud à la demande. En conséquence, les clients peuvent obtenir un résultat beaucoup plus rapide prestation de services à moindre coût[4].

❖ Le serveur se compose du matériel informatique caractéristique et/ou logiciels requis pour la livraison de ce qui précède prestations mentionnées. Le tableau suivant montre les différents services de Cloud avec leurs exemples[2].

SAAS	**PAAS**	**IAAS**
Logiciel en tant que service	Platform as a Service	Infrastructure as a Service
• Applications gouvernementales (Gov-apps) • Communication (courriel) • Collaboration • Outils de productivité (bureautique) • ERP	• Développement d'applications • Services de sécurité • Gestion de base de données	• Les serveurs • Réseau • Stockage • La gestion • Rapportage

Exemples : • Salesforce.com • Suite Internet • Oracles • IBM • Google Apps	Exemples : • Gae • L'azur de Microsoft • Amazon EC2	Exemples : • goGrid • flexiscale • joyent

Tableau 1.2 différents services de Cloud Computing avec leurs exemples.

❖ Tous les services mentionnés ci-dessus sont payants à l'utilisation, ce qui fait du Cloud une option attrayante pour ceux organisations qui n'ont pas les moyens d'acheter, d'installer et de maintenir les services requis[2].

➢ **Stockage en tant que Service**

Le stockage en tant que service (STAAS) est un modèle commercial dans lequel un grand fournisseur de services loue de l'espace dans son infrastructure de stockage sur la base d'un abonnement. L'économie d'échelle dans l'infrastructure du fournisseur de services lui permet de fournir un stockage beaucoup plus rentable que la plupart des particuliers ou des entreprises peuvent fournir leur propre stockage, lorsque le coût total de possession est pris en considération. Le stockage en tant que service est souvent utilisé pour résoudre les problèmes de sauvegarde hors site. Les critiques du stockage en tant que service grande quantité de bande passante réseau nécessaire pour effectuer leur stockage en utilisant un service basé sur Internet[14].

➢ **La Sécurité en tant que Service**

La sécurité en tant que service (SECAAS) est un modèle commercial dans lequel un grand fournisseur de services intègre ses services de sécurité dans un infrastructure d'entreprise sur la base d'un abonnement plus rentable que la plupart des particuliers ou des entreprises peuvent fournir sur leur propre services de sécurité, lorsque le coût total de possession est pris en compte. Ces services de sécurité incluent généralement l'authentification, l'antivirus, l'antimalware/spyware, la détection des intrusions et la gestion des événements de sécurité. [14].

➢ **Données en tant que Service**

Les données en tant que service, ou DAAS, sont un cousin du logiciel en tant que service. Comme tous les membres de la famille « as a service » (AAS), le DAAS repose sur Produit, en l'occurrence les

données, le concept pouvant être fourni à la demande Utilisateurs, quelle que soit leur situation géographique ou leur séparation organisationnelle Fournisseurs et Consommateurs. De plus, l'émergence de l'architecture orientée services (SOA) a rendu la plate-forme réelle sur laquelle les données résident également hors de propos. Cette évolution a permis l'émergence récente du concept relativement nouveau de DAAS. Les données fournies en tant que service ont d'abord été principalement utilisées dans les mashups Web, mais elles sont désormais de plus en plus employés à la fois dans le commerce et, moins fréquemment, au sein d'organisations telles que l'ONU [14].

- **Environnement de Test en tant que Service**

L'environnement de test en tant que service (TEAAS), parfois appelé « environnement de test à la demande », est un environnement de test modèle de livraison dans lequel le logiciel et ses données associées sont hébergés de manière centralisée (généralement dans le Cloud) et sont généralement accessible par les utilisateurs à l'aide d'un client léger, normalement à l'aide d'un navigateur Web sur Internet[14].

- **Backend en tant que Service**

Backend as a service (BAAS), également connu sous le nom de "mobile Backend as a service" (MBAAS), est un modèle de fourniture de services Web et les développeurs d'applications mobiles avec un moyen de lier leur application au stockage Cloud Backend Fournit également la gestion des utilisateurs, les notifications push et Intégration avec les services de réseaux sociaux. Ces services sont fournis via l'utilisation de kits de développement de logiciels personnalisés (SDK) et d'interfaces utilisateur. Programmation d'applications (API). BAAS est un développement relativement nouveau Cloud, la plupart des startups BAAS datent de 2011 ou plus tard [14].

1.5.2. Modèle de déploiement du Cloud Computing

Pour déployer une solution de Cloud Computing, la tâche principale est décidée le type de Cloud à mettre en place Les modèles de déploiement. Il existe quatre modèles de déploiement des solutions du Cloud Computing. La figure ci-dessous montre aperçu du déploiement de ces quatre Cloud [2].

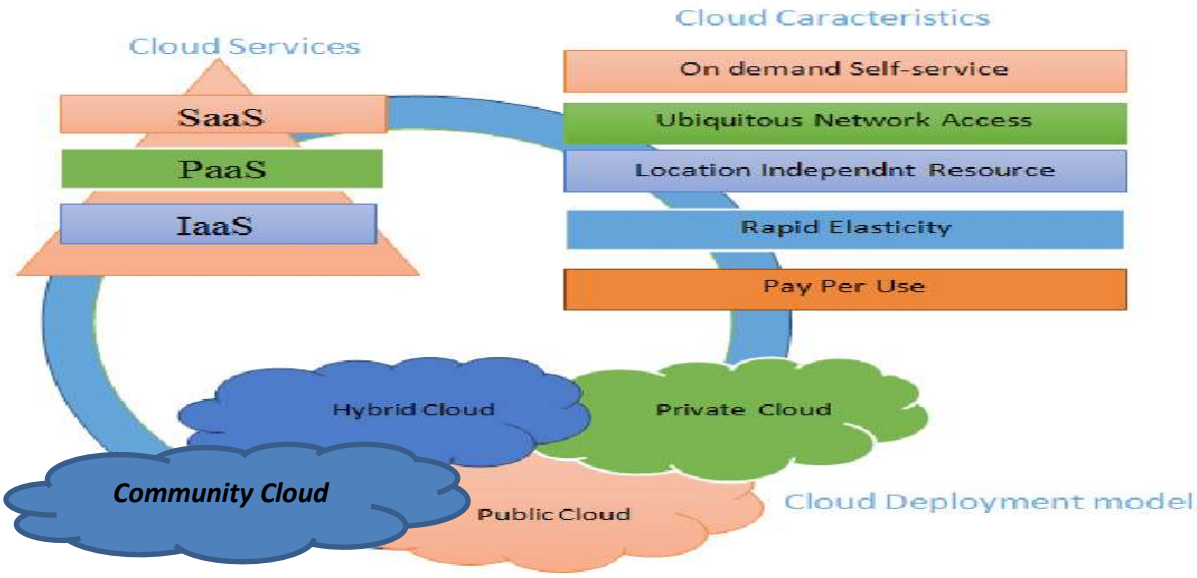


Figure 11.11 Modèles de déploiement Cloud, caractéristiques et infrastructures[2].

- ***Cloud Public***

Le Cloud public permet aux utilisateurs d'accéder au Cloud via interfaces à l'aide de navigateur Web. Les utilisateurs doivent payer uniquement pour la durée d'utilisation du service, c'est-à-dire le paiement à l'utilisation. Cela peut être comparé au système électrique que nous recevoir chez nous. Nous ne payons que le montant de ce nous utilisons. Le même concept s'applique ici. Cela aide à réduire les coûts d'exploitation sur les dépenses informatiques. Toutefois les Cloud publics sont moins sécurisés que les autres Cloud modèles comme toutes les applications et données sur le Cloud public sont plus sujets aux attaques malveillantes. La solution à ce peut être que les contrôles de sécurité soient mis en œuvre via une validation des deux côtés, par le fournisseur de Cloud ainsi que par le client. De plus, les deux parties doivent identifier leurs responsabilités dans leurs limites de fonctionnement[2].

- ***Cloud Privé***

Une opération de Cloud privé se situe au sein d'une organisation centre de données interne de l'entreprise. Le principal avantage ici est qu'il est plus facile de gérer la sécurité, la maintenance et mises à niveau et fournit également plus de contrôle sur le déploiement et utilisation. Le Cloud privé peut être comparé à intranet. Par rapport au Cloud public où toutes les ressources et les applications étaient gérées par le fournisseur de services, en Cloud privé ces services sont regroupés et mis en place disponible pour les utilisateurs au niveau organisationnel. Les ressources et les applications sont gérées par l'organisation

elle-même. La sécurité est renforcée ici car seul les utilisateurs des organisations ont accès au Cloud privé [2].

➢ ***Cloud Hybride***

C'est une combinaison de Cloud public et de Cloud privé. Dans ce modèle, un Cloud privé est lié à un ou plusieurs services Cloud externes. C'est un moyen plus sûr de contrôler les données et les applications et permet d'accéder informations sur Internet. Il permet à l'organisation de servir ses besoins dans le Cloud privé et si certains occasionnels en cas de besoin, il demande au Cloud public des ressources informatiques[2].

➢ ***Cloud Communautaire***

Lorsque de nombreuses organisations construisent et partagent conjointement une infrastructure Cloud, leurs exigences et leurs politiques, puis un tel modèle de Cloud est appelé Cloud communautaire. L'infrastructure Cloud pourrait être hébergée par un tiers fournisseur ou au sein de l'une des organisations de la communauté[2].

1.6. Avantages et Limite Du Cloud Computing

1.6.1. Avantages Du Cloud Computing

Le Cloud Computing offre de nombreux avantages aux utilisateurs et rend le travail beaucoup plus facile et certaines des clés les avantages sont mentionnés ci-dessous :

- Infrastructure informatique élastique et évolutive disponible « à la demande ». Une organisation peut demander plus de puissance de calcul au fur et à mesure des besoins avec une illusion de disponibilité infinie de la puissance du processeur et du stockage. Cette installation est un grand avantage pour les "start-up" qui n'ont pas besoin inutilement investir dans d'énormes infrastructures. Au fur et à mesure de leurs affaires se développe, ils peuvent demander plus de puissance de calcul à un fournisseur et ne payent que ce qu'ils utilisent[1].
- ressources informatiques coûteuses avec les compétences rares peuvent être exploitées dans de nombreux les clients [15]. « Utilisation accrue Ne payez que ce qui est utilisé » [9].
- Faible maintenance Activités de maintenance traditionnelles à forte intensité de ressources sont transférées au fournisseur. Supprime l'informatique en tant que goulot d'étranglement [15]. « Une fois les

contrats établis, les utilisateurs peuvent acheter directement des fournisseurs sans intervention informatique »

- Meilleurs niveaux de service. Les ressources sont partagées[9]. L'Infrastructure plus fiable, la disponibilité de ressources qualifiées 24 heures sur 24, 7 jours sur 7 peut permettre aux fournisseurs de Cloud de fournir de meilleurs services qu'un magasin informatique interne aux ressources limitées[15].
- déploiement rapide est important de noter que l'informatique en Cloud donne à l'avantage d'un déploiement rapide. Une fois opté pour Ce mode de fonctionnement, tout le système peut être entièrement fonctionnel en quelques secondes minutes. Bien sûr, le temps passé ici dépendra du type exact de technologie nécessaire à l'entreprise[16].

1.6.2. Limite Du Cloud Computing

Certains des risques perçus sont énumérés ci-dessous [1]:

- L'utilisation d'un Cloud dépend essentiellement d'une communication ininterrompue avec l'infrastructure du fournisseur de Cloud. Échec de la communication coupera un service Cloud. Elle peut être atténuée par fournir un deuxième moyen de communication indépendant (redondant). Par exemple, si une connexion au Cloud utilise la fibre communication optique, une autre pourrait être par système sans fil tel comme WiMax.

- Chaque fois que des données ou un programme sont envoyés sur un système de communication et les données sont stockées sur un disque partagé système, il existe un risque que des personnes indiscrètes écoutent la communication ligne et voler ou corrompre des données ou les voler stockage sur disque. Pour atténuer cela, il est essentiel d'utiliser des cryptages de toutes les données envoyées au Cloud sur une communication système et également crypter les données avant de les stocker.

- Un autre risque est la détérioration de la qualité de service d'un fournisseur de Cloud ou un fournisseur qui cesse ses activités en raison d'une faillite. Dans un tel cas, les organisations doivent avoir un plan de secours pour déplacer leurs applications vers un autre fournisseur. La principale difficulté ce faisant, c'est le manque de standardisation des services Cloud fournies par différents fournisseurs.

- Des problèmes juridiques complexes peuvent survenir si les serveurs des fournisseurs sont pays étranger et les programmes et les données d'une organisation sont corrompu ou volé. Une organisation doit clarifier quelles lois appliquer lors de la signature du Niveau de service Agreement avec un Cloud fournisseur de services.

- Un problème récent est la surveillance clandestine du trafic de données sur Internet par les agences de renseignement des États-Unis. Et Royaume-Uni. Comme l'infrastructure des fournisseurs de Cloud est répartie sur l'ensemble du monde, il n'est peut-être pas judicieux d'utiliser ces services, en particulier si le données à traiter ou le programme est sensibles.

1.7. Les défis de l'adoption du Cloud

Comme le Cloud Computing en est encore à ses balbutiements, l'adoption actuelle est associée à de nombreux défis[17].

Sécurité : Il est clair que la question de la sécurité a joué le plus rôle important dans l'entrave à l'informatique en Cloud. Sans aucun doute, mettre vos données, exécuter votre logiciel sur le disque dur de quelqu'un d'autre disque utilisant le processeur de quelqu'un d'autre semble intimidant pour beaucoup, Problèmes de sécurité bien connus tels que la perte de données, le phishing, le botnet (fonctionnant à distance sur un ensemble de machines) pose de sérieux problèmes menaces pour les données et les logiciels de l'organisation. De plus, le modèle multi-tenant et les ressources informatiques mutualisées dans le Cloud Computing ont introduit de nouveaux défis de sécurité qui nécessitent de nouvelles techniques pour s'y attaquer. Par exemple, les pirates prévoient d'utiliser le Cloud pour organiser le botnet en tant que Cloud fournit souvent des services d'infrastructure plus fiables à un prix relativement moins cher pour eux de lancer une attaque.

Modèle d'établissement des coûts : Les consommateurs de Cloud doivent considérer les compromis entre calcul, communication et intégration. Lors de la migration au Cloud peut réduire considérablement le coût de l'infrastructure, il augmente le coût de la communication des données, c'est-à-dire le coût de transférer les données d'une organisation vers et depuis le public et Community Cloud et le coût unitaire (par exemple une VM) de la ressource informatique utilisée est susceptible d'être plus élevée. Par exemple, pour s'attaquer au problème de sécurité, consommateurs de Cloud peuvent être amenés à scinder des données confidentielles en morceaux et répartissez-les sur différents Cloud afin que la compromission de la sécurité dans un Cloud ne conduise pas au désastre dans son ensemble. Cependant, le fractionnement et le mélange des données non n'ajoute qu'un coût financier supplémentaire substantiel, mais peut également affecter les performances du système (c'est-à-dire le coût en temps)

Modèle de charge :Du point de vue d'un fournisseur de Cloud, la ressource élastique pool (via la virtualisation ou la multi location) a fait l'analyse des coûts beaucoup plus compliquée que les données régulières centres, qui calcule souvent leur coût en fonction consommations du calcul statique. De plus, une

instanciation la machine virtuelle est devenue l'unité d'analyse des coûts plutôt que le serveur physique sous-jacent. Pour les fournisseurs de Cloud SaaS, le coût de développement de la multi location au sein de leur offre peut être très important. Celles-ci comprennent : la refonte et le redéveloppement du logiciel qui a été utilisé à l'origine pour une location unique, coût de la fourniture de nouveaux fonctionnalités qui permettent une personnalisation intensive, des performances et l'amélioration de la sécurité pour l'accès simultané des utilisateurs Par conséquent, les fournisseurs de SaaS doivent peser le compromis entre la fourniture de multi-location et les économies de coûts générés par la multi-location tels que la réduction des frais généraux grâce à amortissement, réduction du nombre de licences de logiciels sur site, etc. Par conséquent, un modèle de facturation stratégique et viable pour le SaaS fournisseur est crucial pour la rentabilité et la pérennité de Fournisseurs de Cloud SaaS.

Accord de niveau de service (SLA) : Bien que les consommateurs de Cloud n'aient pas le contrôle sur les ressources informatiques sous-jacentes, ils doivent assurer la qualité, la disponibilité, la fiabilité et les performances de ces ressources lorsque les consommateurs ont migré leur cœur de métier fonctions sur le Cloud qui leur a été confié. En règle générale, ceux-ci sont fournis via le niveau de service Des accords (SLA) négociés entre les fournisseurs et consommateurs. Le tout premier problème est la définition du SLA spécifications d'une manière qui a un niveau approprié de granularité, à savoir les compromis entre expressivité et complexité, différentes offres Cloud (IAAS, PAAS, SAAS et DAAS) devront définir différentes méta spécifications SLA. Cela pose également un certain nombre de problèmes de mise en œuvre pour les fournisseurs de Cloud. De plus, des mécanismes SLA avancés besoin d'intégrer constamment les commentaires et la personnalisation des utilisateurs caractéristiques dans le cadre d'évaluation SLA.

Quoi migrer : Basé sur une enquête (taille de l'échantillon = 244) menée par IDC en 2008, les sept systèmes/applications informatiques en cours de migration vers le Cloud sont : les applications de gestion informatique (26,2 %), Applications collaboratives (25,4 %), applications personnelles (25 %), applications métier (23,4 %), applications Développement et déploiement (16,8 %), capacité du serveur (15,6 %) et capacité de stockage (15,5 %). Ce résultat révèle que les organisations ont toujours des problèmes de sécurité / confidentialité lors du déplacement leurs données sur le Cloud. Actuellement, les fonctions périphériques telles car la gestion informatique et les applications personnelles sont les plus faciles Systèmes informatiques à déplacer. Les organisations sont conservatrices dans employant IaaS par rapport au SaaS. C'est en partie parce que les fonctions marginales sont souvent externalisées vers le Cloud, et le cœur les activités sont maintenues en interne.

1.8. Conclusion

Dans ce chapitre, nous avons discuté d'une nouvelle vague dans le domaine de technologies de l'information : Cloud Computing "Défis De L'adoption ". Et Nous avons aussi décrit son architecture, ses avantages et certaines limites. Il y a nul doute que le Cloud Computing est la tendance de développement pour l'avenir. Nous pouvons avoir un approximativement infini des capacités, évolutivité. Nous verrons dans le prochain chapitre de ce travail fonctionnement des différentes méthodes exactes puis d'heuristique, de métaheuristique avec différence entre les méthodes d'optimisation.

CHAPITRE 2: Les Approches D’optimisation

2.1. Introduction

En ingénierie, de nouveaux problèmes combinatoires se posent, ne peut pas être résolu avec les méthodes existantes, ce qui inspire les scientifiques et des chercheurs pour suggérer de nouvelles méthodes et s'en inspirer, telles que, Phénomènes naturels et comportement collectif de quelques insectes développer des algorithmes d'optimisation, ces méthodes sont appelées métaheuristique[18].

Dans ce chapitre, nous commençons par donner les différentes méthodes exactes puis la définition d'heuristique, de métaheuristique et de leurs caractéristiques. Nous présenterons par la suite quelques métaheuristique en finit ce chapitre par la différence entre Les méthodes d'optimisation.

2.2. Méthodes d'optimisation

L'optimisation combinatoire (OC) occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications pouvant être modélisées sous la forme d'un problème d'optimisation combinatoire (POC) telles que le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de la coloration de graphes, etc.

2.2.1. Méthodes exactes

Nous montrons d'abord quelques méthodes de la classe des algorithmes complets ou exacts, ces méthodes donnent une garantie de découvrir la solution optimale pour une instance de taille finie dans un temps limite et de prouver son optimalité[19].

2.2.1.1. La Méthode Séparation et Évaluation (Branch And Bound)

L'algorithme de séparation et évaluation, plus connu sous son désignation anglaise Branch and Bound (B&B), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des nœuds, et des feuilles[20].

Le Branch-and-Bound est basé sur trois axes principaux :

- **L'évaluation**

L'évaluation agrée de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale. Le but est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence. Le Branch-and-Bound utilise une élimination de branches dans l'arborescence de recherche [20].

- **La séparation**

La séparation consiste à deviser le problème en sous-problèmes. Ainsi, en trouvant tous les sous-problèmes et en conservant la meilleure solution considérée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions. L'ensemble de nœuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des nœuds actifs[20].

- **La Stratégie de Parcours**

La largeur d'abord : Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées.

La Profondeur d'abord : Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

Le Meilleur d'abord : Cette stratégie consiste à explorer des sous problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale[20].

2.2.1.2. La Méthode de Coupes Planes (Cutting-Plane)

La méthode de coupes planes a été développée par Schrijver, elle est destinée à résoudre des problèmes d'optimisation combinatoire (POC) qui se formulent sous la forme d'un programme linéaire (PL) :

$$Min\ c^T x : Ax \geq b, x \in R^n \qquad (1.1)$$

Cette méthode consiste à résoudre un problème relaxé, et à ajouter itérativement des contraintes du problème initial. On définit une contrainte pour le problème de minimisation (1.1) par le couple (s,s_0) où $s \in R^T$ et $s_0 \in R$, cette contrainte est dite violée par la solution courante x si pour tout $y \in \{x : Ax \geq b\}$ on a $s^T x < s_0$ et $s^T y > s_0$, on appelle alors ces contraintes des coupes planes. On arrête l'algorithme lorsqu'il n'y a plus de contraintes violées par la solution courante, on obtient ainsi une solution optimale pour le problème initial. La méthode des coupes planes est peu performante mais sa performance est améliorée lorsqu'elle est combinée avec la méthode "Branch and Bound"[21].

2.2.1.3. La Méthode (Branch And Cut)

La méthode Branch and Cut n'est pas toujours efficace face aux problèmes difficiles. De même, bien que l'algorithme du "Branch and Bound" puisse être très performant pour une certaine classe de

problèmes, pour cela on utilise la méthode "Branch and Cut" qui combine entre l'algorithme du "Branch and Bound" et de la méthode des coupes planes. Pour une résolution d'un programme linéaire en nombres entiers, la méthode "Branch and Cut" commence d'abord par relaxer le problème puis appliquer la méthode des coupes planes sur la solution trouvée. Si on n'obtient pas une solution entière alors le problème sera divisé en plusieurs sous-problèmes qui seront résolus de la même manière. On veut résoudre le problème d'optimisation (min $c^t x$: Ax ≥ b; x ∈ $\boldsymbol{R^n}$) avec A ∈ R^{mxn} et b ∈ R^m [22].

Algorithm 1 Branch and Cut

Liste des problèmes $= \emptyset$;
Initialiser le programme linéaire par le sous problème de contraintes (A_1, b_1) avec $A_1 \in \mathbb{R}^{m_1 xn}$et $b_1 \in \mathbb{R}^{m_1})$ avec $m_1 << m$;
Etapes d'évaluation d'un sous problème
Calculer la solution optimale $\bar{x}$ du programme linéaire $c^t\bar{x} = min(c^t x : A_1 x \geq b_1, x \in \mathbb{R}^n)$;
Solution courante= Appliquer la méthode des coupes polyédrales();
Fin étapes d'évaluation
Si Solution courante est réalisable **alors**
$x^* = \bar{x}$ est la solution optimale de $min(c^t x : Ax \geq b, x \in \mathbb{R}^n)$;
Sinon
Ajouter le problème dans Liste des sous problèmes;
Fin Si
Tant que Liste des sous problèmes $\neq \emptyset$ **faire**
Sélectionner un sous problème;
Brancher le problème ;Appliquer les étapes d'évaluation;
Fin Tant que

Figure12.1 Algorithme d'optimisation Brand and Cut.

2.2.1.4. La Méthode de la Génération de Colonnes

Le principe de la génération de colonnes repose sur le fait que toutes les variables d'un programme linéaire ne seront pas forcément utilisées pour atteindre la solution optimale. L'objectif de cette méthode est de résoudre un problème réduit avec un ensemble limité de variables.

Le problème consistant à chercher la meilleure variable à rajouter dans le problème restreint est appelé sous-problème associé au problème maître (ou oracle). Il a comme objectif de trouver la variable (ou colonne) de coût réduit minimum (c.-à-d. la plus prometteuse pour améliorer la solution) [22].

Le coût réduit des variables est calculé à l'aide des variables duales obtenues après la résolution du problème restreint. Le point du dual ainsi utilisé dans le sous problème est appelé point de séparation. Souvent, il s'agit d'une solution optimale du dual du problème restreint. On considère le programme linéaire continu (LP) suivant :

$$\begin{cases} Min \sum_{i \epsilon T} c_i \, x_i \\ \sum_{i \epsilon T} a_{ij} \, x_i \geq b_j \, , j = 1 \ldots , n \\ x_i \geq 0 \; \forall i \in T \end{cases}$$

Nous supposons que le nombre de variables de T est trop grand pour que le problème (LP) puisse être résolu en temps raisonnable, et que nous voulions le résoudre par génération de colonnes. Nous cherchons donc à résoudre le problème restreint associé au problème maître avec un ensemble restreint de variables notéR_l. Il faut que le problème restreint soit réalisable. Il est possible d'utiliser des colonnes simples par exemple des colonnes aléatoires, ou encore celles issues d'une solution faisable obtenue à partir d'une heuristique.

Le problème restreint (RLP) est donné sous la forme suivante :

$$\begin{cases} Min \sum_{i \in R_l} c_i x_i \\ \sum_{i \in R_i} a_{ij} x_j \geq b_j \; j = 1 \ldots n \\ x_i \geq 0 \; \forall i \; \in R_l \end{cases}$$

Le problème (RLP) est maintenant de taille réduite et sera plus facile à résoudre par un solveur. Cette résolution nous fournira les valeurs optimales des variables duales $v_j$associées aux contraintes. Ces valeurs sont passées au sous problème qui nous permet d'obtenir la ou les colonnes à rajouter dans l'ensembleR_l.

Le calcul du coût réduit nous permet de savoir si une colonne a fait diminuer la valeur de l'objectif (et donc de l'améliorer). Prenons par exemple la colonne xi du problème maître (LP), son coût réduit vaut :

$$\bar{c_i} = c_i - \sum_{j=1}^{n} a_{ij} v_j$$

Puisque (LP) est un problème de minimisation, le sous problème cherche aussi à minimiser ce coût réduit. Si le coût réduit minimum est positif, alors aucune colonne ne peut être ajoutée au problème restreint (RLP)

pour améliorer l'objectif. La solution optimale du problème restreint est donc une solution optimale du problème maître (LP). Sinon, on rajoute une ou des colonnes parmi celles ayant un coût réduit négatif en faisant une mise à jour de l'ensemble R_l et on résout après le nouveau problème restreint (RLP) [22].

2.2.2. Heuristiques

2.2.2.1. Définition 1

Une heuristique est un algorithme approché qui fournit rapidement une solution réalisable, sans garantie d'optimalité, pour un problème d'optimisation spécifique. Il s'agit d'une règle empirique, d'une stratégie, d'une méthode ou d'une astuce utilisée pour améliorer les performances d'un système qui tente de trouver des solutions à des problèmes complexes[23].

2.2.2.2. Définition 2

Dans les problèmes à complexité exponentielle, il est souvent impossible de réaliser une exploration complète de l'ensemble des solutions réalisables en un temps raisonnable. Une stratégie consiste alors à partir d'une solution initiale réalisable, on propose une amélioration basée sur une modification locale pour qu'elle puisse quitter un optimum local et donc converger le plus vite possible vers l'optimum global. C'est le principe fondamental d'une heuristique[24].

2.2.2.3. Solutions Heuristiques

Il existe différentes solutions heuristiques. On en cite quelques-unes sans trop rentrer dans les détails [24] :

1. Best-Fit Decreasing : Best-fit decreasing trie les nombres dans l'ordre décroissant, puis affecte chaque nombre tour à tour à la classe la plus complète dans laquelle il rentre. Best-fit decreasing emballe l'ensemble de nombres ci-dessus dans deux bacs. L'algorithme fonctionne dans le temps O (nlogn) [25].

2. First-Fit Decreasing : First-fit decreasing trie les nombres dans l'ordre décroissant, ordonne les bacs et attribue à tour de rôle chaque numéro au premier bac dans lequel il rentre. Mais First-fit decreasing nécessite trois bacs pour emballer l'ensemble de nombres ci-dessus, Cette algorithme aussi fonctionne dans le temps O (nlogn)[25].

3. Méthodes par Séparation-Evaluation Avortées : exécuter partiellement une méthode par Séparation-Evaluation, et l'arrêter lorsqu'un critère d'arrêt est vérifié **[18]**. Limite de temps qui risque de ne pas avoir de solution réalisable (borne primale), écart prédéfini entre borne primale ou borne duale et critère

mixte qui est la limite de temps s'il y a une borne primale OU écart prédéfini entre borne primale ou duale[25].

2.2.3. Métaheuristique

Une métaheuristique est un algorithme d'optimisation qui s'applique pour la résolution des problèmes d'optimisation difficiles et complexes souvent dans les domaines de la recherche opérationnelle ou de l'intelligence artificielle. Les métaheuristiques sont généralement des algorithmes stochastiques, qui cherchent à faire progresser vers un optimum global, autrement dit l'extremum global d'une fonction par échantillonnage d'une fonction objective[26].

2.2.3.1. Caractéristiques

Les caractéristiques essentielles des métaheuristiques sont les suivantes :

- Les métaheuristiques sont basées sur des paramètres qui guident l'exploration et affectent la qualité de la solution.
- Les métaheuristiques nécessitent la spécification d'un point de départ pour l'exploration, qui est choisie aléatoirement.
- Les métaheuristiques nécessitent la spécification d'une condition d'arrêt.
- Les métaheuristiques parcourent un espace de recherche en se basant sur la combinaison de deux actions intensification et la diversification[26].

2.2.3.2. Classification des Métaheuristiques

Les métaheuristiques sont généralement classées selon plusieurs paramètres, nous citons les suivantes:

a. Fonction Objectif

C'est-à-dire la manière d'utiliser la fonction objective par une métaheuristique. Cette catégorie comprend deux types de métaheuristiques statique et dynamique. Supposons qu'on a un problème d'optimisation afin de maximiser ou bien de minimiser une fonction f sur un espace de recherche de solutions nommé S, alors nous avons :

- Les métaheuristiques qui travaillent directement sur f dites statiques.
- Les métaheuristiques dynamiques qui travaillent sur une fonction g obtenue à partir de la fonction f en ajoutant quelques composants à fin de changer la topologie l'espace de recherche S[27].

b. Nombre de Solutions

Nous distinguons deux familles de métaheuristiques : à base de solution unique (S-métaheuristique) et les métaheuristiques à base de population de solutions (P-métaheuristique) [27].

- **Les Métaheuristiques À Solution Unique**

Cette famille de métaheuristiques commence la recherche avec une solution initiale (appelée configuration initiale), et au cours du processus de la recherche, elles essayent d'améliorer sa qualité progressivement tout en choisissant une nouvelle solution dans son voisinage, ils sont appelés aussi les méthodes de recherche locale ou méthodes de trajectoire car ils construisent une trajectoire dans l'espace des solutions tout en se redirigeons vers des solutions optimales, parmi les métaheuristiques les plus célèbres de cette famille, nous citons : la recherche taboue, le recuit simulé[26].

- **Les Métaheuristiques à Base de Population**

Dans cette famille de métaheuristiques, sont utilisés pour optimiser les problèmes complexes de grandes tailles. Contrairement aux méthodes exactes, les métaheuristiques à base de population de solutions améliorent une population de solutions, au fur et à mesure des itérations[28].

Ils sont parfois nommés des méthodes évolutives parce qu'elles font évoluer une population d'individus selon des règles bien précises, l'intérêt de cette famille de métaheuristiques est d'utiliser la population comme facteur de diversité pour augmenter la possibilité d'apparition de bonnes solutions en matière de qualité, parmi les métaheuristiques célèbres de cette famille, nous citons : les algorithmes de colonies de fourmis, la recherche coucou (cuckoo search)[29].

La figure ci-dessous, présente la classification des métaheuristiques selon le paramètre nombre de solutions.

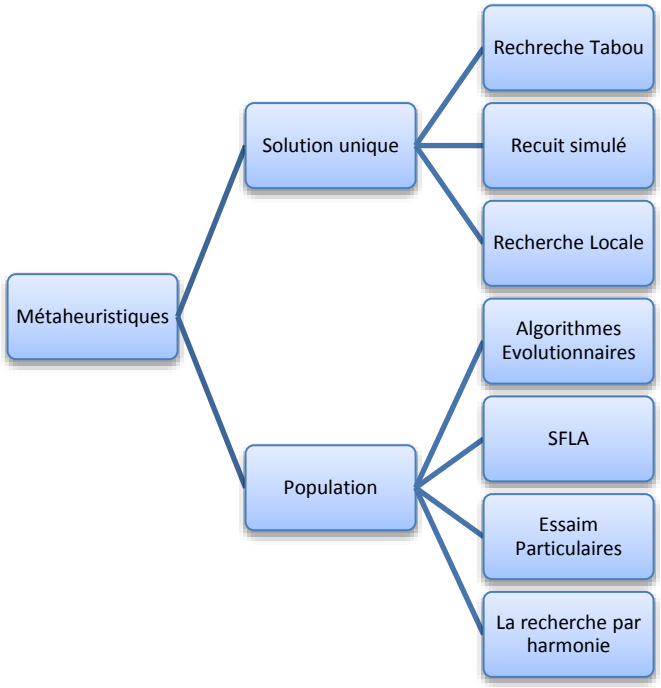


Figure13.2 Classification des métaheuristiques.

Les métaheuristiques à base de population sont classées en deux grandes familles : les algorithmes basés sur l'intelligence en essaim.

Les algorithmes d'intelligence en essaim inspirés par les insectes sociaux sont de plus en plus étudiés en informatique pour résoudre des problèmes complexes. On cité l'algorithme de colonies de fourmis (ACO), l'algorithme des chauves-souris (bat algorithm) et l'algorithme de la recherche coucou (Cuckoo search).

Et les algorithmes évolutionnaires (comme les algorithmes génétiques). En ce qui suit, nous nous présentons quelques méthodes.

2.2.3.3. Quelques Méthodes Métaheuristiques

1. Le recuit simulé : Le recuit simulé (SA) a été introduit par (Kirkpatrik et al. 1983) et (Cerný 1985) comme une méthode de recherche locale normale, utilisant une stratégie pour éviter les minima locaux. Cette métaheuristique est basée sur une technique utilisée depuis longtemps par les métallurgistes qui, pour obtenir un alliage sans défaut, faisant alterner les cycles de réchauffage (Ou de recuit) et de refroidissement lent des métaux. Le recuit simulé s'appuie sur des travaux faire par (Metropolis et al. 1953), qui ont pu décrire l'évolution d'un système en thermodynamique[22].

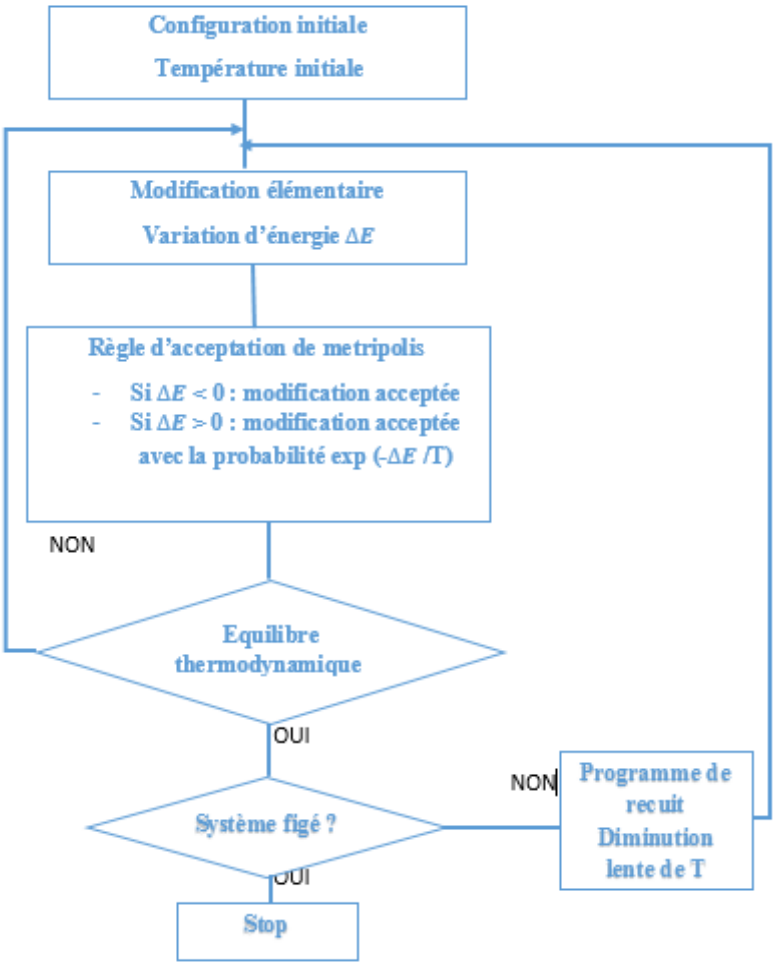


Figure14.3 Fonctionnement de l'algorithme de recuit simulé[22].

2.La recherche Tabou : La recherche tabou (TS) est une méthode de recherche locale combinée avec un ensemble de techniques permettant d'éviter d'être piégé dans un minimum local ou la répétition d'un cycle. La recherche taboue est introduite principalement par Glover (Glover 1986), Hansen (Hansen 1986). Cette méthode a montré une grande efficacité pour la résolution des problèmes d'optimisation difficiles. En effet, à partir d'une solution initiale s dans un ensemble de solutions local S, des sous-ensembles de solution N(s) appartenant au voisinage S sont générés. Par l'intermédiaire de la fonction d'évaluation nous retenons la solution qui améliore la valeur de f, choisie parmi l'ensemble de solutions voisines N(s) [22].

Algorithm La recherche tabou

1: Initialisation :
   s_0 une solution initiale
   $s \leftarrow s_0$, $s^* \leftarrow s_0$, $c^* \leftarrow f(s_0)$
   $T = \emptyset$
2: Générer un sous-ensemble de solution au voisinage de s
   $s' \in N(s)$ tel que $\forall x \in N(s)$, $f(x) \geq f(s')$ et $s' \notin T$
   Si $f(s') < c^*$ alors $s^* \leftarrow s'$ et $c^* \leftarrow f(s')$
   Mise-à-jour de T
3: Si la condition d'arrêt n'est pas satisfaite retour à l'étape 2

Figure15.4 Algorithme de la recherche tabou[22].

3. L'optimisation par Essaim de Particules : L'Optimisation par Essaim Particulaire (OEP, ou PSO en anglais) a été proposée par (Kennedy et Eberhart 1995). Cette méthode est inspirée du comportement social des animaux évoluant en essaim. Au départ, ils cherchaient à simuler la capacité des oiseaux à voler de façon synchrone et leur aptitude à changer brusquement de direction tout en restant en une formation optimale**.** Les particules sont les individus et elles se déplacent dans l'hyperespace de recherche en se basant sur des informations limitées.

Chaque individu utilise, non seulement, sa propre mémoire, mais aussi l'information locale sur ses plus proches voisins pour décider de son propre déplacement. Des règles simples, telles que " aller à la même vitesse que les autres ", " se déplacer dans la même direction " ou encore " rester proche de ses voisins " sont des exemples de comportements qui suffisent à maintenir la cohésion de l'essaim [22].

Algorithm Optimisation par Essaim Particulaire (OEP)

1: Initialiser aléatoirement P_s particules : position et vitesse ;
2: Evaluer les positions des particules ;
3: **Tant que** le critère d'arrêt n'est pas atteint **faire**
4: **Pour** $i = 1, ..., P_s$ **faire**
5: Déplacer les particules selon la vitesse et la position
6: **Si** $f(x_i) < f(p_i)$
7: $p_i = x_i$;
8: **Si** $f(x_i) < f(p_g)$
9: $p_g = x_i$;
10: **Fin Si**
11: **Fin Si**
12: **Fin Pour**
13: **Fin Tant que**

Figure16.5 Algorithme d'optimisation par essaim particulaire[22].

1. Algorithmes Évolutionnaires

Les algorithmes évolutionnaires constituent une discipline impliquant la simulation du processus de l'évolution naturelle sur un ordinateur. Ceci peut être vu comme un processus d'optimisation ou des individus évoluent dans le temps, afin de devenir de plus en plus adéquats à un environnement donné, le problème à résoudre[24].

En général, les AE sont divisés en quatre saveurs principales, présentées dans cette section : les algorithmes génétiques, la programmation génétique, les stratégies d'évolution et la programmation évolutionnaire.

- **Algorithme Génétique**

Les algorithmes génétiques (AG) impliquent l'évolution d'une population de vecteurs de dimension fixe composée de caractères, généralement des bits. L'idée des AG est vaguement inspirée des chaînes d'ADN, qui composent tout organisme vivant.

Les AG sont utilisées dans le but de découvrir une solution, généralement numérique, résolvant un problème donné, et ce sans avoir de connaissance a priori sur l'espace de recherche. Seul un critère de qualité est nécessaire pour discriminer différentes solutions. Dans la plupart des cas, ce critère, l'adéquation, est une mesure objective qui permet de quantifier la capacité de l'individu à résoudre le problème donné.

Le processus de recherche s'effectue en appliquant itérativement à une population de solutions potentielles des opérations de variation génétique (généralement le croisement et la mutation), et des opérations de sélection naturelle biaisées vers les individus les plus forts. En utilisant ce processus Darwinien, la population de solutions potentielles évolue dans le temps jusqu'à ce qu'un certain critère d'arrêt soit atteint. La (figure 2.6) présente plus en détails le processus évolutionnaire des AG. Dans un contexte d'optimisation de fonctions à paramètres réels, une variante importante des AG consiste à représenter les individus directement par des vecteurs de nombres réels[30].

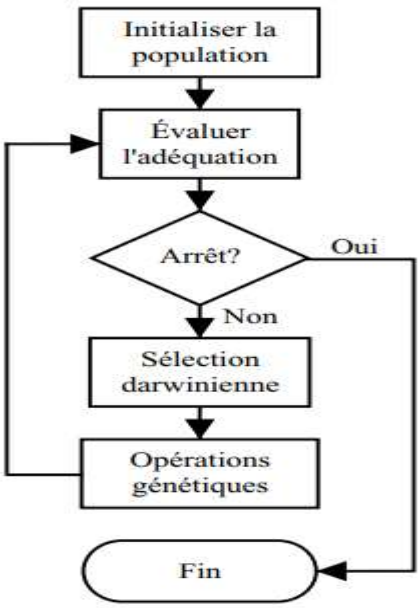


Figure17.6 Processus évolutionnaire classique des algorithmes génétiques[30].

- **Programmation Génétique**

La programmation génétique (PG) est un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques basées sur les mêmes principes d'évolution que les AG : des opérations de variation génétique, par des croisements et des mutations, et des opérations de sélection naturelle. La différence entre la PG et les AG réside essentiellement dans la représentation des individus. En effet, la PG consiste à faire évoluer des individus dont la structure est similaire à des programmes informatiques. La PG a été exprimée formellement par Koza au début des années 1990[31].

La PG utilisée par Koza représente les individus sous forme d'arbres, c'est-à-dire des graphes orientés et sans cycle, dans lesquels chaque nœud est associé à une opération élémentaire relative au domaine du problème. Plusieurs autres représentations comme des programmes linéaires et des graphes cycliques, ont été utilisés depuis. La PG est particulièrement adaptée à l'évolution de structures complexes de dimensions variables[32].

PG ressemble à l'évolution d'une population de programmes informatiques. Les quatre étapes de la programmation génétique consistent à :

1) Générer une population initiale de programmes informatiques comprenant les fonctions et les terminaux.
2) Exécutez chaque programme dans la population et attribuez-lui une valeur de fitness en fonction de sa capacité à résoudre le problème.
3) Créer une nouvelle population de programmes informatiques.

i) Copier les meilleurs programmes existants

ii) Créer de nouveaux programmes informatiques par mutation.

iii) Créer de nouveaux programmes informatiques par croisement (reproduction sexuée) [33].

- **Stratégies D'évolution**

Les stratégies d'évolution (SE) constituent une technique d'AE développée par I. Rechenberg et H-P. Schwefel à Berlin dans les années 1960[24].

Les stratégies d'évolution (SE) représentent les individus comme un ensemble de caractéristiques de la solution potentielle. En général, cet ensemble prend la forme d'un vecteur de nombres réels de dimension fixe. Les SE s'appliquent à une population de parents (de dimension μ) à partir de laquelle des individus sont sélectionnés aléatoirement afin de générer une population de descendants (de dimension $\Delta \geq \mu$). Ceux-ci sont ensuite modifiés par des mutations qui consistent à ajouter une valeur générée aléatoirement selon une fonction de densité de probabilité paramétrée. Les paramètres de la fonction de densité de probabilité, nommés les paramètres de la stratégie, évoluent eux aussi dans le temps selon les mêmes principes que les paramètres caractérisant les individus.

Pour former la nouvelle population, μ individus sont choisis (approche (μ, Δ)) parmi les μ meilleurs individus des Δ descendants, ou (approche $(\mu + \Delta)$) parmi les μ meilleurs individus des μ parents et Δ descendants. Les SE modernes peuvent aussi intégrer une approche multi-échelle ou des stratégies d'évolution sont imbriquées[24].

- **Programmation Évolutionnaire**

La programmation évolutionnaire (PE) a été développée par L.J. Fogel dans les années 1960 et reprise par D.B. Fogel et d'autres chercheurs dans les années 1990. La PE a été initialement conçu dans le but de faire évoluer des machines à états finis et a été par la suite étendue aux problèmes d'optimisation de paramètres. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que de simuler des opérateurs génétiques d'inspiration naturelle. Contrairement aux trois autres AE classiques, la PE n'utilise pas une représentation spécifique mais plutôt un modèle évolutionnaire de haut niveau, jumelé à une représentation et à un opérateur de mutation directement appropriés au problème à résoudre.

Pour effectuer de la PE, une population de μ solutions potentielles au problème est d'abord générée aléatoirement. Chaque individu i de la population produit $\boldsymbol{\lambda}$ descendants résultant de mutations. Une opération de sélection naturelle est alors appliquée afin de former une nouvelle population de μ individus.

Le processus de mutation - sélection est répété itérativement jusqu'à ce qu'une solution acceptable soit trouvée[24].

La différence entre les deux familles est que dans les métaheuristiques à population, l'espace de recherche devient le produit cartésien $S^N = S \times S \times S \times S...$, où N est le nombre d'individus de la population et S est l'espace de recherche d'un individu[34].

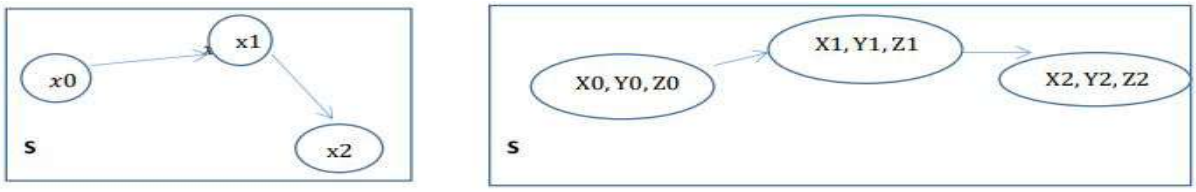


Figure18.7 Espace de recherche dans les deux familles[34].

2.2.4. Différence entre les Approches Exactes Heuristiques Et Métaheuristiques

1. **Les Méthodes Exactes :** Ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité[24].

2. **Les Méthodes Heuristiques :** Elles identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée[24].

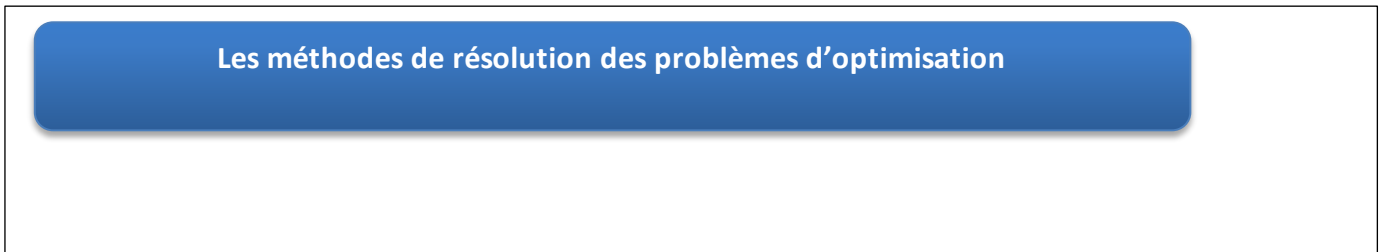
3. **Les Méthodes Métaheuristiques :** Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les métaheuristiques ont fait leur apparition. Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage[24].

Ce que l'on retient comme avantages et inconvénients des méthodes à utiliser pour résoudre efficacement un problème d'optimisation combinatoire est synthétisé dans (le tableau 2.4).

Approches	Approches Exactes	Approches Heuristiques	Approches Métaheuristiques
Avantages	Solution optimale (instances moyennes) Détection des cas faciles.	Solutions parfois optimales Mise en œuvre facile	Moins puissantes
Inconvénients	Temps de réponse long (cas des grandes instances) Caractérisation difficile	Difficiles à paramétrer	Non plus de l'optimum global en un temps fini

Tableau 2.3 Comparaison entre les approches exactes et les approches heuristiques.

La figure 2.8. Représente une synthèse de classification des méthodes de résolution de problèmes d'optimisation.



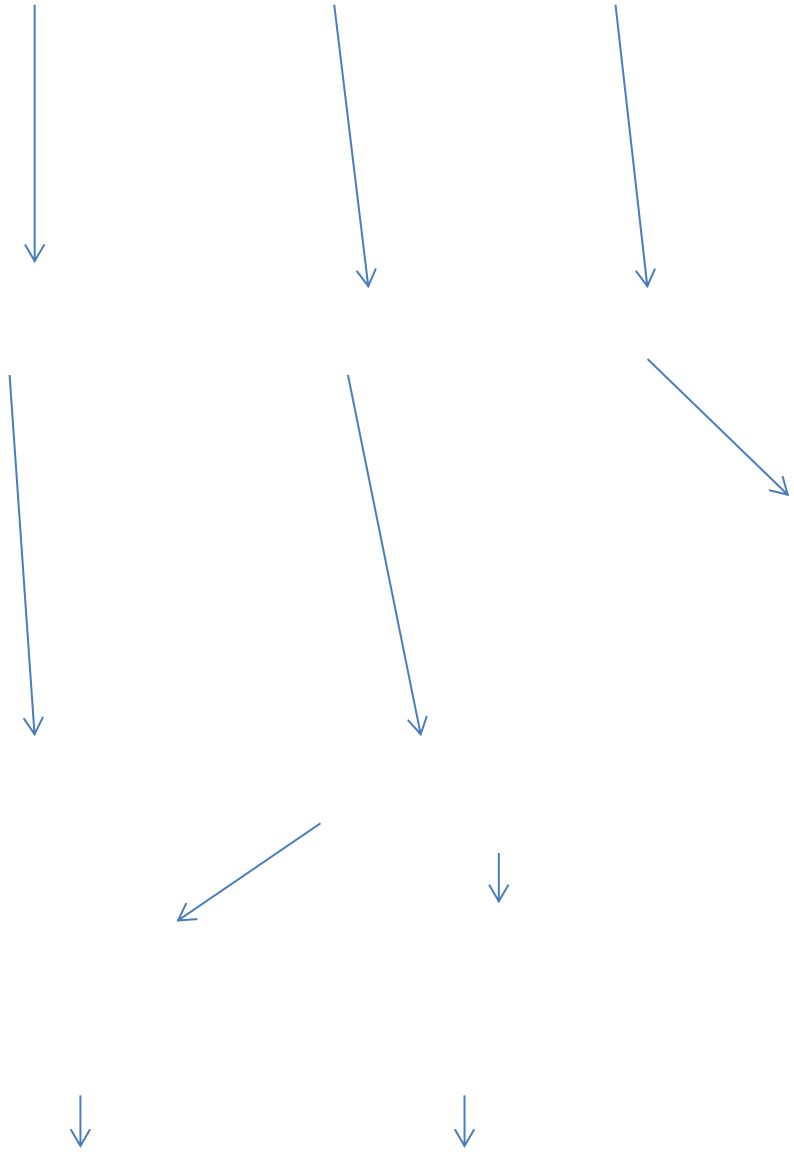


Figure19.8 Synthèse de Classification des méthodes de résolution de problème d'optimisation.

2.3. Conclusion

Dans ce chapitre, nous avons présenté les notions des méthodes exactes, heuristiques et de métaheuristiques, ensuite nous avons proposé une classification des métaheuristiques en se basant soit sur la fonction objective ou bien sur le nombre des solutions. Et en finit par la différence entre ces méthodes d'optimisation.

Cela nous a conduits à déterminer notre besoin pour mettre en place une optimisation de placement de la virtuelle machine dans l'environnement Cloud.

CHAPITRE 3: Problème de Placement par Algorithme Génétique

CHAPITRE 3 Problème de Placement par Algorithme Génétique

3.1. Introduction

Afin de résoudre le problème de placement des machines virtuelles dans les centres de données Cloud, plusieurs étapes doivent être réalisées. Pour chercher à minimiser l'énergie. Ce chapitre présent brièvement le placement des machines virtuelles et l'algorithme génétique et l'applique à plusieurs études de cas. Pour génétique ce sera le modèle à utiliser pour la résolution de ce problème étant donné qu'il regroupe tous les paramètres à prendre en considération.

3.2. Algorithme Génétique

3.2.1. Inspiration

L'Algorithme Génétique (AG) a été inspiré de la théorie de l'évolution de Darwin, qui simule la survie d'organismes plus sains et de leurs gènes. C'est un algorithme basé sur des populations où chaque individu correspond à un chromosome, et chaque paramètre représente un gène. L'AG utilise une fonction de fitness (objectif) pour évaluer la fitness de chaque individu dans la population. Ce qui rend cet algorithme fiable et capable d'estimer l'optimum global pour un problème donné est le processus consistant à maintenir les meilleures solutions à chaque génération et à les utiliser pour résoudre le problème. Par conséquent, toute la population est transmise de génération en génération. L'intersection entre individus conduit à exploiter la "région" entre deux solutions mères données. L'algorithme bénéficie également de la mutation. L'opérateur modifie au hasard les gènes du chromosome, maintient la diversité des individus dans la population et augmente le comportement d'exploration de l'algorithme génétique. Semblables à la nature, les opérateurs de mutation peuvent produire des solutions nettement meilleures et pousser d'autres solutions vers l'optimum global [35].

3.2.2. Définition

L'Algorithme génétique est un algorithme d'optimisation stochastique basé sur l'évolution avec un potentiel de recherche global proposé par Holland en 1975. Les AG font partie de la classe d'algorithmes qui s'inspirent des idées évolutives de la sélection naturelle. Ils suivent les principes de la théorie de Charles Darwin sur la survie du plus apte. Cependant, en raison de ses performances exceptionnelles en optimisation, l'AG a été considéré comme un optimiseur de fonction. L'algorithme commence par initialiser une population de solutions (chromosome). Il comporte une représentation du problème, généralement sous la forme d'un vecteur de bits. Ensuite, pour chaque chromosome, évaluer l'aptitude à l'aide d'une fonction d'aptitude adaptée au problème. Sur cette base, les meilleurs chromosomes sont sélectionnés dans le pool d'accouplement, où ils subissent un croisement et une mutation, donnant ainsi un nouvel ensemble de solutions [33].

3.2.3. Description :

Les algorithmes génétiques sont basés sur deux hypothèses:

1) la condition fitness d'un individu est une mesure de sa capacité relative à résoudre le problème.
2) la combinaison d'individus permet la formation de meilleurs progéniture.

Si la première hypothèse n'est pas correcte, alors il sera difficile pour l'AG de faire la distinction entre les bonnes solutions et les solutions médiocres. En conséquence, lors de la reproduction, les deux types de solutions généreront un nombre égal de copies, ralentissant tout mouvement vers des solutions améliorées. De plus, si les bonnes solutions (comme indiqué par le classement de fitness) ne contiennent pas de morceaux de la meilleure solution, il sera difficile pour l'AG de générer la meilleure réponse [36].

3.2.4. Concepts de base

3.2.4.1. Population initiale

La population initiale de solutions candidates est généralement généré aléatoirement dans l'espace de recherche. Cependant, des connaissances spécifiques à un domaine ou d'autres informations peuvent être facilement intégrées [37]. Un autre terme, L'algorithme génétique commence par une population aléatoire peut générer cette population augmenter la diversité à partir d'une distribution aléatoire gaussienne. Dans cette population plusieurs solutions sont incluses, qui représentent des chromosomes individuels. Pour Chaque chromosome. Il existe un ensemble de variables, qui sont utilisées pour modéliser les gènes. Objectif principal dans L'étape d'initialisation consiste à répartir la solution uniformément dans l'espace de recherche Peut accroître la diversité de la population et avoir de meilleures chances de trouver domaine prometteur. La section suivante traite des étapes pour améliorer les chromosomes dans la première population[38].

3.2.4.2. Évaluation

Une fois que la population est initialisée ou qu'une population descendante est créer, les valeurs de fitness des solutions candidates sont évaluées [37].

3.2.4.3. Sélection

Choisir la sélection naturelle est la principale inspiration de cette composante de l'algorithme génétique [38]. La sélection à louer plus de copies de ces solutions avec des valeurs de fitness et impose ainsi le mécanisme de survie du plus apte aux solutions candidates. L'idée principale de la sélection est de préférer les meilleures solutions aux pires, et de nombreuses procédures de sélection ont été proposées pour accomplir cette idée, y compris la sélection à la roulette, la sélection universelle stochastique, la sélection de classement et la sélection de tournoi. La phase de sélection détermine quels individus sont

choisis pour l'accouplement (reproduction) et combien de descendants chacun produits individuels sélectionnés. Le grand principe de la stratégie est le meilleur individu, plus grande est sa chance d'être parent. C'est le processus qui détermine quelles solutions doivent être conservées et autorisées à se reproduire et lesquels méritent de disparaître. L'objectif premier de l'opérateur de sélection est de mettre l'accent sur les bonnes solutions et éliminer les mauvaises solutions dans une population, tout en gardant taille de la population constante. La sélection introduit l'influence de la fonction de fitness sur le processus d'optimisation des algorithmes génétiques. La sélection doit utiliser la forme physique d'un individu donné, puisque la forme physique est la mesure de la « bonté » d'un individu. Cependant, la sélection ne peut pas être basée uniquement sur le choix du meilleur individu, car le meilleur individu peut ne pas être très proche de la solution optimale. Au lieu de cela, une chance que relativement inapte les individus sélectionnés doivent être préservés, afin de s'assurer que les gènes portés par ces personnes inaptes ne sont pas « perdus » prématurément de la population. En général, la sélection implique un mécanisme reliant l'aptitude d'un individu à la moyenne forme physique de la population. Plusieurs stratégies de sélection ont été développées et utilisé pour l'optimisation des algorithmes génétiques[39].

a) Sélection par roulette

Le rôle de la roulette est la zone couverte par l'ensemble du chromosome dans une population selon la valeur de fitness. Chaque individu donne la tranche de roulette et les tailles de tranche sont directement proportionnelles à la condition physique de chaque individu. Maintenant, ce processus est répété jusqu'à ce que le nombre souhaité d'individus soit sélectionné. Si la fitness est plus élevée, la taille du segment est plus grande [40].

Voici les étapes pour la sélection de la roulette :

- Calculer la somme des valeurs de fitness de chaque individu dans la population.
- Calculer la valeur fitness de chaque individu et leur probabilité de sélection en divisant, l'individu fitness du chromosome par la somme des valeurs de fitness de toute la population.
- Diviser la roulette en secteurs selon les probabilités calculées à la deuxième étape.
- Faites tourner la roue "n" nombre de fois. Quand la roulette s'arrête, le secteur sur lequel le pointeur pointe correspond à la personne sélectionnée.

La probabilité de sélection d'un individu a_j :

$$p_s(a_i) = \frac{f(a_i)}{\sum_{i=1}^{n} f(a_i)} \quad ; j = 1,2 \dots \dots n$$

Où 'n'est la taille de la population, f (a_i) est la valeur de fitness de l'individu a_i[39].

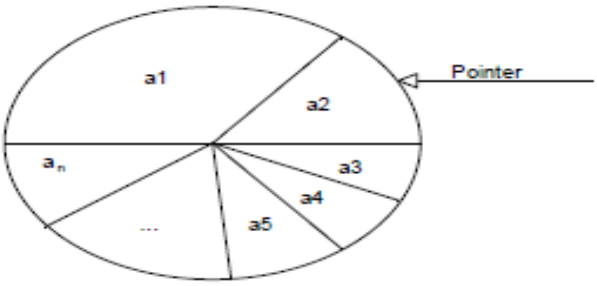


Figure 20.1 Sélection par roulette.

b) Sélection par rang

Dans la sélection par rang, classer tout d'abord la population et après cela, à partir du classement, chaque gène reçoit la forme physique. Dans sélection de classement, nous pouvons trier du meilleur au pire. Le but de la sélection des rangs est d'éviter une convergence trop rapide.

Avantages : Éviter une convergence trop rapide.

Inconvénients : les populations doivent être triées à chaque cycle[40].

c) Sélection par classement linéaire

La sélection de classement dans l'algorithme génétique a été introduite par Baker pour éliminer les inconvénients de la sélection proportionnelle. Dans la méthode de sélection du classement linéaire, les individus sont d'abord triés en fonction de leur valeur de fitness, puis les rangs qui leur sont attribués. Le meilleur individu obtient le rang ‘N’ et le pire on obtient le rang '1'. La probabilité de sélection est alors attribuée linéairement aux individus selon leurs rangs.

$p_i = \frac{1}{N}(n^- + (n^+ - n^-)\frac{i-1}{N-1})$; i ∈{1, N}

Où Pi est la probabilité de sélection de l’ième individu.

$\frac{n^-}{N}$ Est la probabilité de sélection du pire individu.

Chaque individu obtient un rang différent même si leur probabilités sont les mêmes. Le processus de classement comporte deux étapes. Dans la première étape, on trie les populations et en second, on attribue les rangs dans l'ordre correspondant à la Sélection proportionnée [39].

d) Sélection par classement exponentiel

Cette technique est différente de la sélection par classement linéaire technique de manière à ce que les probabilités ici soient exponentiellement pondérées. La base de l'exposant est c, où 0 < c < 1.

$$p_i = \frac{c^{N-i}}{\sum_{j=i}^{N} c^{N-j}} \quad i \in \{1,\ldots. N\}$$

La somme $\sum_{j=i}^{N} c^{N-j}$ normalise les probabilités pour assurer que$\sum_{i=1}^{N} p_i = 1$ L'algorithme du classement linéaire et exponentiel Le classement est le même. La seule différence est dans le calcul des probabilités de sélection. En cela aussi le meilleur individu est attribué le rang « N », et le pire est attribué « 1 » [39].

e) Sélection par Tournoi

Dans la sélection du tournoi, diverses tournées de quelques individus sont sélectionnées au hasard parmi la population et sélectionnés les meilleurs individus en tant que parent. Nous sélectionnons les meilleures valeurs de fitness avec vainqueur de chaque tournoi. Lorsque la taille du tournoi est changée, nous pouvons facilement ajuster la pression de sélection.

Avantage : a) Lors de la mise en œuvre en parallèle, il dispose d'un temps efficace et de complexité.

b) Mise à l'échelle ou tri de la condition physique non requis.

Inconvénients : Ce type de sélection ne donne pas garantie de reproduction de la meilleure solution[40].

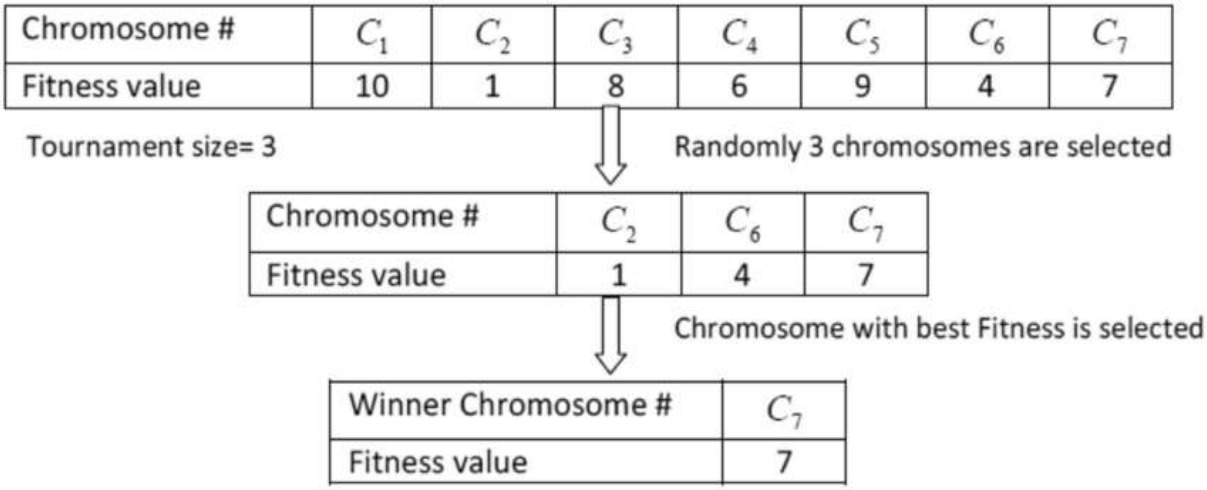


Figure 21.2 exemple de sélection par tournoi[39].

f) Sélection d'élitisme

Dans la sélection par élitisme, nous pouvons copier les premiers meilleurs chromosomes ou les quelques meilleurs chromosomes dans la nouvelle population. Tout d'abord, nous pouvons ranger le chromosome dans l'ordre décroissant. Après cette sélection est appliquée avec tous les deux chromosomes

dans l'ensemble d'arrangement. Aucune modification n'est nécessaire, nous pouvons transmettre le meilleur individu à la prochaine génération

Avantages : Cela améliore considérablement les AG performance parce que l'élitisme génère une population très en forme [40].

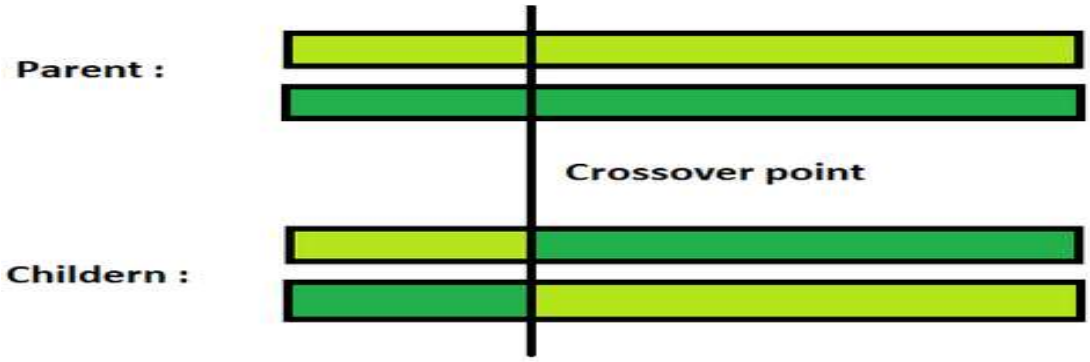
3.2.4.4. Croisement (recombinaison)

La recombinaison combine des parties de deux ou plusieurs parents solutions pour créer de nouvelles solutions, éventuellement meilleures (c'est-à-dire la progéniture). Il existe de nombreuses façons d'accomplir cela et des performances compétentes basées sur un mécanisme de recombinaison correctement conçu. La progéniture sous recombinaison ne sera pas identique à un parent particulier et combinera plutôt les traits parentaux d'une manière nouvelle[37]. Et Une fois les individus sélectionnés à l'aide d'opérateurs de sélection, ils doivent être utilisés pour créer une nouvelle génération. Dans la nature, les chromosomes des gènes mâles et femelles se combinent pour créer de nouveaux chromosomes. Ceci est simulé en combinant La sélection de deux solutions (solutions parents) dans l'algorithme AG produit deux nouvelles solutions (solutions enfants). Il existe différentes techniques pour les opérateurs de croisement dans la littérature[38]

1) Croisement à point unique

Le croisement à point unique est le croisement le plus approuvé qui est largement utilisé. Un site de croisement est sélectionné au hasard sur la longueur des chaînes accouplées et les bits qui sont très proches des sites croisés sont échangés. Lorsqu'un site approprié est choisi, une meilleure descendance peut être obtenue en combinant les bonnes qualités des parents. Si le site approprié est choisi, une meilleure progéniture peut être obtenue en combinant des parents de bonne qualité, sinon cela entrave fortement la qualité des cordes. Dans le croisement à point unique, la tête et la queue d'un chromosome se séparent et si la tête et la queue ont un bon matériel génétique, aucun des descendants n'obtiendra

directement les deux bonnes caractéristiques[41].



Chromosome1	11011|00100110110
Chromosome2	11011|11000011110
Offspring1	11011|11000011110
Offspring2	11011|00100110110

Figure 22.3 croisement de points unique[41].

2) Croisement de points N

Le croisement de points N a été implémenté pour la première fois par De Jong en 1975. Il a de nombreux sites de croisements mais la règle utilisée est la même que celle que nous avons utilisée dans le croisement à point unique. Dans la signification de croisement à 2 points du croisement, les sites sont 2. L'ajout de plus en plus de sites de croisements affecte les perturbations des blocs de construction qui réduisent parfois les performances de l'algorithme génétique. Mais cela permet à la tête et à la queue d'un chromosome d'être acceptées ensemble dans la progéniture[41].

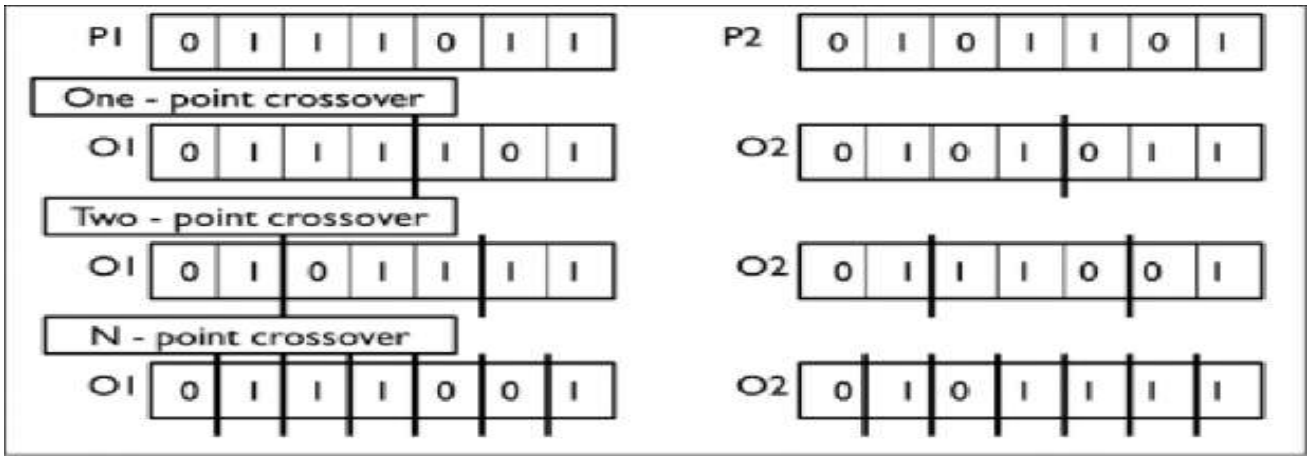


Figure 23.4 croisement de N point [41].

3) Croisement uniforme

Le croisement uniforme ne fragmente pas les chromosomes pour la recombinaison. Chaque gène de la progéniture est créé en le copiant à partir du parent choisi en fonction du bit correspondant dans le masque de croisement binaire de même longueur que la longueur des chromosomes parents. Si le bit dans le masque croisé est 1, alors le gène résultant est copié à partir du premier parent et si le bit dans le masque croisé est 0, alors le gène résultant est copié à partir du second parent. Un nouveau masque de croisement est généré arbitrairement pour chaque paire de chromosomes parents. La quantité de point de croisement n'est pas fixée initialement. Ainsi, la progéniture a un mélange de gènes des deux parents[41].

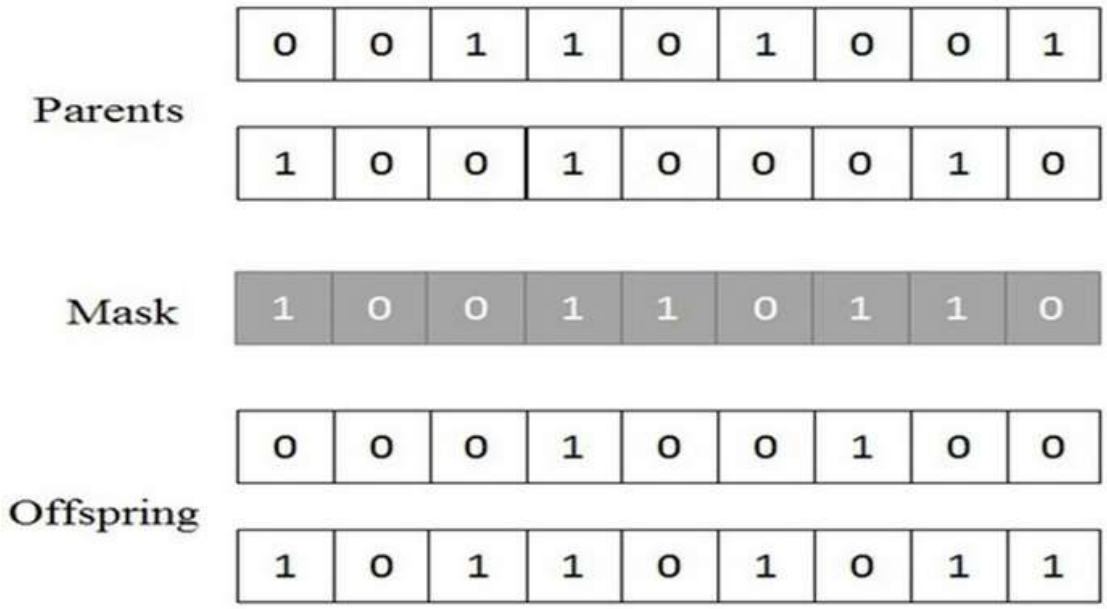


Figure 24.5 Croisement uniforme[41].

3.2.4.5. Mutation

Alors que la recombinaison opère sur deux ou plusieurs chromosomes parentaux, la mutation modifie localement mais de manière aléatoire une solution. il existe de nombreuses variantes de mutation, mais cela implique généralement un ou plusieurs changements apportés aux traits d'un individu [37]. C'est Le dernier opérateur évolutif, où un ou plusieurs gènes sont modifiés après avoir la création des de solutions. Le taux de mutation dans l'algorithme génétique est défini très faible car un taux de mutation élevé convertit l'algorithme génétique en une recherche aléatoire brute. L'opérateur de mutation maintient la diversité de la population en introduisant un autre niveau d'aléatoire. exister En fait, cet opérateur empêche les solutions de devenir similaires et augmente la probabilité d'éviter les solutions locales dans l'algorithme génétique [38].

3.2.4.6 Remplacement

La population de descendants créer par sélection, croisement et mutation remplace la population parentale d'origine. De nombreuses techniques de remplacement telles que le remplacement élitiste, le remplacement par génération et les méthodes de remplacement en régime permanent sont utilisées dans les AG[37].

3.3. Problème De Placement Des Machines Virtuelles

Le problème de placement est un problème ancien et classique. Il comporte à mettre des objets définis et caractérisés par un volume, dans des cases aux volumes limités. L'objectif est de placer tous les objets en utilisant un nombre minimum de cases.

Dans ce qui va suivre, on donnera une similitude entre la définition classique du problème de placement et sa définition dans un contexte de Cloud Computing. En effet, on fera similitude en considérant les machines virtuelles comme des objets, et les hôtes comme les cases à remplir. Le nouveau problème de placement consiste alors à placer les machines virtuelles dans un minimum d'hôtes[42].

3.3.1. Placement de MVs dans le Cloud

Dans les centres des données traditionnels, les applications sont taillées à des serveurs physiques spécifiques qui sont le plus fréquemment soit surchargés ou peu utilisés, et cela dans le but de rencontrer aux demandes complexes et aléatoires. Comme conséquence, cette utilisation non raisonnable des ressources produit forcément des coûts opérationnels élevés et des nouveaux investissements (nombre de serveurs) non pas pour pallier aux pertes de ressources, mais surtout pour pouvoir répondre aux différentes nouvelles demandes et charges. Cette situation dans laquelle on trouve des serveurs non utilisés ou peu

utilisés, coûtera alors en termes de mètres carrés occupés, en consommation énergétique, et en toutes les opérations de supervisions et de maintenance associées [42].

Pour certains, le problème de placer des MVs pour réduire la consommation d'énergie se résume à la surutilisation des PMs dans le centre de données. Mettre ces serveurs physiques en veille est l'objectif principal. Selon [43] il est très important de choisir un serveur adapté pour la veille. Pour cela, l'approche suit une topologie similaire à la topologie de réseau en arbre. L'algorithme parcourt ainsi l'arbre et essaie de mettre le sous arbre de chaque nœud parcouru une MV. Le nombre de MVs pouvant être acceptées par un même serveur sera alors estimé pour chaque nœud. Si des ressources et une bande passante suffisantes sont disponibles, le nœud sera validé pour attribuer cette VM au serveur actuel. Pour cette approche, plus la capacité des serveurs est grande et plus la méthode est efficace.

Du fait que les MVs ont un impact direct sur la consommation énergétique des PMs sur lesquelles elles sont hébergées, une étude a montré l'importance de réduire les besoins énergétiques d'un centre de données grâce à la gestion énergétique des ressources allouées aux MVs [44].

Dans ce même modèle, la planification des machines virtuelles tient compte de la puissance des équipements informatiques. En effet, chaque équipement participe de près à la définition de l'empreinte écologique d'un centre de données, allant des machines virtuelles jusqu'aux commutateurs qui relient les différents châssis en passant par le système de ventilation. Les MVs est la seule variable parmi tous les équipements, ce sont ces dernières qui définissent la puissance nécessaire à leurs alimentations. (La figure 3.11) illustre l'architecture d'un centre de données utilisé pour la modélisation du problème. Elle démarre à partir d'un commutateur principal. Les commutateurs agrégés s'occupent de sécuriser le trafic réseau entre les différents composants. Les châssis sont ceux qui regroupent les serveurs [45].

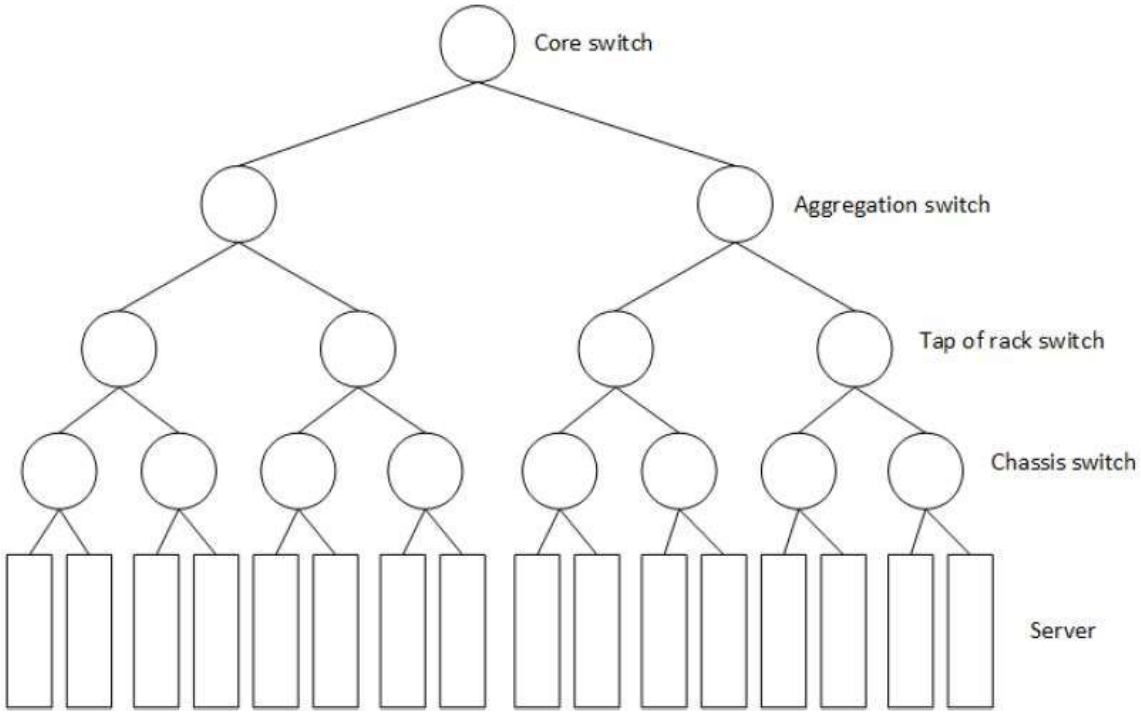


Figure 25.10 Topologie d'un centre de données[45].

Dans cet environnement Cloud Computing, un système de gestion des machines virtuelles sera chargé d'affecter chaque machine virtuelle au centre de données adéquat en fonction du client qui a soumis sa demande. (La figure 3.12) montre une vue de cet environnement, où le VMMS s'occupera de la gestion des machines virtuelles.

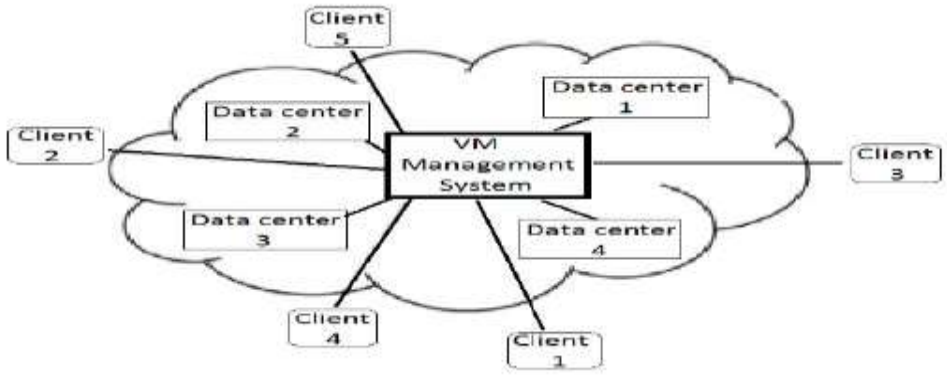


Figure 26.11 Environnement de Cloud Computing [45].

3.3.2. Intérêts Au Problème De Placement De Machines Virtuelles

Pendant la période 2007 jusqu'à 2030, la consommation énergétique au moderne international va agrandir de 76% [46], avec une grande participation en terme de consommation des centres des données.

Ceci produit clairement à réfléchir sur les méthodes à suivre pour diminuer cette consommation dans le Cloud. De plus, et selon le rapport Gartner, un centre des données de taille moyenne consomme en énergie électrique semblable de 25000 foyers. Et selon un rapport McKinsey [47]. En 2010, les coûts de consommation énergétique des centres des données sont évalués à 11.5 bn de dollars, sachant aussi que ces coûts augmentent chaque 5ans.

À ce processus, s'ajoute une information importante sur la consommation d'un serveur en mode veille, et qui est estimée à 70% de sa consommation maximale. Cette perte d'énergie à cause des serveurs en mode veille est considérée comme une cause majeure de la consommation totale d'un centre des données. Ceci mets simplement le besoin de diminuer et de baiser cette consommation par des mécanismes d'optimisation comme celui du placement intelligent des MVs dans le centre des données pour solliciter un plus petit nombre possible de serveurs, et d'éteindre le reste[48].

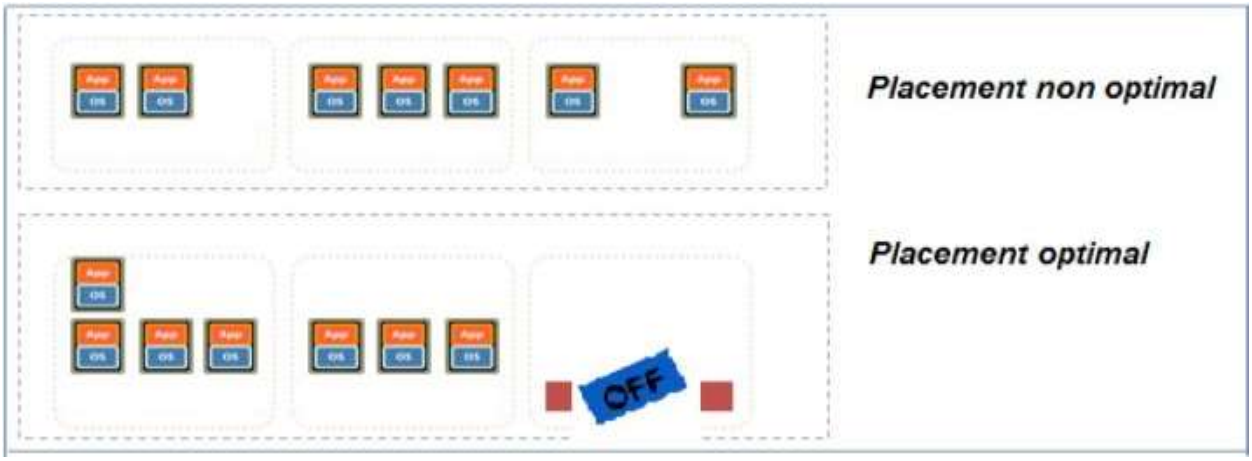


Figure 27.13 Exemple de placement de MVs [48].

3.3.3. Optimisation du Placement des Machines Virtuelles

Après avoir acquis des informations sur les machines virtuelles dans le Cloud, un ensemble de candidats appropriés est mis en évidence. Le mécanisme de sélection des ressources élit la solution candidate qui répond à toutes les exigences et optimise l'utilisation de l'infrastructure. La sélection des ressources peut être effectuée à l'aide d'un algorithme d'optimisation. De nombreuses stratégies d'optimisation peuvent être utilisées, à partir de techniques simples et bien connues telles que les algorithmes métaheuristiques simples «L'algorithme génétique » [49].

3.4. Formulation de problème

En raison de la demande croissante de services Cloud, l'optimisation de la consommation des ressources devient encore plus essentielle pour augmenter les performances, la disponibilité et la fiabilité

des systèmes Cloud. Différentes machines virtuelles demandées par les utilisateurs peuvent avoir des exigences différentes en matière de traitement, de mémoire, d'entrée/sortie et de mise en réseau. Les serveurs physiques peuvent également avoir des capacités différentes. Cela conduit à un problème d'optimisation connu sous le nom de problème de placement de machine virtuelle. Une solution à ce problème peut viser à augmenter l'utilisation des machines physiques pour réduire les coûts et la consommation d'énergie. Le problème de placement des machines virtuelles est un problème NP difficile. Dans notre travail, nous proposons une solution basée sur l'algorithme génétique au problème de placement de la machine virtuelle, qui utilise une fonction de fitness et une structure chromosomique et considère l'utilisation des ressources, l'utilisation de la bande passante du réseau et les coûts énergétiques en même temps. À la fin, nous arrivons à une solution qui est assez bonne et se rapproche de la solution optimale au problème. Avant de présenter notre algorithme, nous devons définir certains paramètres que nous avons utilisés, comme indiqué dans le tableau 3.1

$\boldsymbol{V}$	un ensemble de machines virtuelles
$\boldsymbol{P}$	un ensemble de machines physique
$\boldsymbol{v_i}$	machine virtuelle i
$\boldsymbol{v_i^{cpu}}$	CPU requis de v_{ie} en MIPS
$\boldsymbol{v_i^{mem}}$	mémoire requise de v_i en MO
$\boldsymbol{v_i^{bp}}$	bande passante requise de v_i en Mb/s
$\boldsymbol{p_j}$	Machine physique j
$\boldsymbol{p_j^{cpu}}$	Capacité CPU de p_j en MIPS
$\boldsymbol{p_j^{mem}}$	Capacité de mémoire de p_j en MO
$\boldsymbol{p_j^{bp}}$	Capacité de bande passante de p_j en Mb/s
$\boldsymbol{v_j}$	liste des machines virtuelles affectées à p_j
$\boldsymbol{p_j^{cpu_j}}$	Utilisation totale du Cpu de p_j en MIPS
$\boldsymbol{p_j^{mem_j}}$	Utilisation totale du mémoire de p_j en MO
$\boldsymbol{p_j^{bp_j}}$	Utilisation totale de la bande passante de p_j en Mb/s
$\boldsymbol{E_j}$	énergie totale consommée par p_j exprimé en watt
$\boldsymbol{E_j^{max}}$	la consommation d'énergie de p_j lorsque $U_j = 100\%$
$\boldsymbol{E_j^{idle}}$	la consommation d'énergie de p_j lorsque $U_j = 0\%$
$\boldsymbol{U_j}$	le pourcentage d'utilisation du processeur de p_j

Nous visons à minimiser l'énergie consommée par les machines physiques. Nous pouvons exprimer la fonction objective de notre problème par l'équation:

$$\sum_{p_j \in P} E_j \qquad (3.1)$$

Sous les contraintes suivantes :

$$V = \bigcup_{p_j \in P} v_j \qquad (3.2)$$

$$p_j^{bp} \geq \sum_{v_i \in v_j} v_i^{bp} \qquad (3.3)$$

$$p_j^{mem} \geq \sum_{v_i \in v_j} v_i^{mem} \qquad (3.4)$$

$$p_j^{cpu} \geq \sum_{v_i \in v_j} v_i^{cpu} \qquad (3.5)$$

$$v_i \bigcap v_j = \emptyset, si\ i \neq j \qquad (3.6)$$

$$Uj = w_j^{cpu} / P_j^{cpu} \qquad (3.7)$$

Éq. 3.2 indique que chaque machine virtuelle est affectée à une machine physique, tandis que l'Eq. 3.6 spécifie que chaque machine virtuelle est affectée à une seule machine physique. Les équations 3.3, 3.4 et 3.5 indiquent que la demande totale des machines virtuelles pour chaque ressource (CPU, mémoire et bande passante) ne dépasse pas la quantité de ressource correspondante de la machine physique à laquelle ces machines virtuelles sont affectées. Comme indiqué dans l'Eq. 3.7, nous définissons l'utilisation d'une machine physique en pourcentage de l'utilisation du processeur. La consommation d'énergie d'une machine est calculée en fonction de l'utilisation de la machine, comme indiqué dans l'équation. Nous donnons ensuite les composants de notre algorithme génétique pour machine virtuelle placement.

3.4.1. Placement de MV à l'aide de Algorithme Génétique

En 2013 Ajith Singh. N et al. Examinaient le problème de placement de MV en se concentrant sur la maximisation de l'utilisation des ressources et de la réduction d'énergie. Le problème de placement de MV est résolu en utilisant l'algorithme génétique avec la classification hiérarchique pour réduire la consommation d'énergie dans les serveurs [50].

3.4.2. La description de l'algorithme génétique

Les algorithmes évolutionnaires sont inspirés du domaine de la biologie, les techniques utilisées reproduisent le schéma d'évolution des espèces et en adoptent le vocabulaire. Les solutions sont appelées individus et l'algorithme traite simultanément plusieurs individus d'individu utilisé dans la simulation de notre problématique est illustré ici-bas :

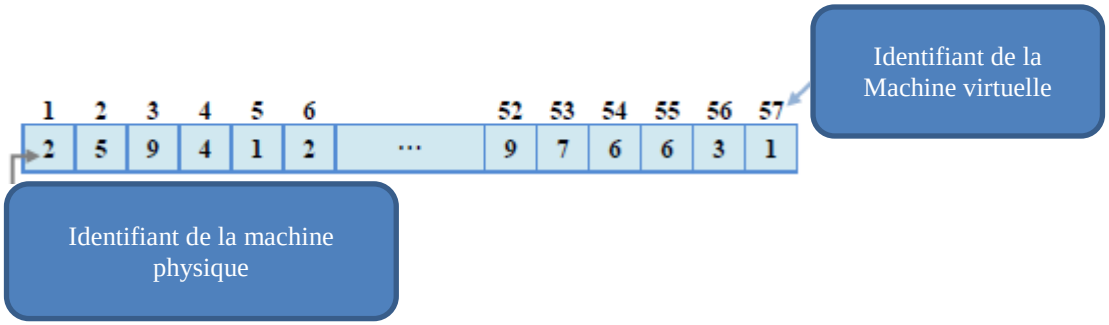


Figure 28.14 La simulation de problématique.

L'ensemble de ces individus est appelé population et évolue à chaque itération de l'algorithme. La population relative à une itération donnée s'appelle génération, on obtient donc une nouvelle génération à chaque itération. Les individus qui servent à produire la nouvelle génération sont appelés parents et les individus résultants sont appelés enfants ou fils.

- ✓ **L'opération de sélection**

Nous avons décidé d'utiliser l'opération de sélection uniforme par rapport au rang. Notre opération de sélection qui est décrite dans la fonction ci-dessous consiste à choisir de façon équiprobable les individus de rang inférieur ou égal à μ avec μ ≤ N. Les autres individus sont exclus de la population et ne peuvent pas participer au croisement.

La probabilité de sélection s'exprime par:

$$P_s(Ii) = \begin{cases} \frac{1}{\mu}, & si\ 1 \le i \le \mu \\ 0, & \mu < i \le N \end{cases} \quad (3.11)$$

Function Selection

Require: Population

Ensure: Best population

2: create best population

3: create new individual

4: While THE TERMINATION CONDITION is not true do

5: set the best of population in new individual

6: add the new individual to best population

7: return best population

8: end function

✓ **L'opération de croisement**

Détermine quels gènes sont hérités de quels parents, comme décrit dans la fonction de croisement suivante. Le taux split dans l'algorithme consiste à déterminer le site de croisement qu'est sélectionné au hasard sur la longueur des chaînes accouplées. Selon le hasard généré valeur, un parent est sélectionné et une machine virtuelle est affectée à la même machine physique dans le chromosome du ressort, celle qui est affectée dans le chromosome du parent sélectionné.

Function Crossover

Require: Two parent individual: ind1, ind2 and split point

Ensure: Two offspring individual: new sol1 and new Sol2 //sol represent solution

2: create 2 new individual newSol1 newSol2

3: for i = 1 to |V| do

4: if i ≤ split then

5: set gene i of new Sol1 as gene i of ind1 & new sol2 as gene i of ind2

6: else

7: set gene i of new Sol1 as gene i of ind2 & new sol2 as gene i of ind1

8: end if

9: end for

10: end function

✓ **Opération de mutation**

L'opération de mutation sélectionne de manière aléatoire une machine virtuelle et une machine physique, supprime cette machine virtuelle de sa machine physique actuelle et la place sur la machine physique nouvellement sélectionnée. Si cette migration entraîne une surcharge sur la machine physique nouvellement sélectionnée, l'opération de mutation est annulée. O n na décrit en détail le fonctionnement de la mutation. L'opération de mutation est essentielle pour les méthodes qui utilisent des algorithmes génétiques car elle augmente la variété parmi la population de printemps.

Function Mutation

Require: population before mutation

Ensure: population after mutation

1: for i = 1 to population size do

2: create new individual

3: for i to |V|

4: if rate > random

5: randomly select a virtual machine v

6: select a physical machine p

7: remove v from its current physical machine

8: add v to p

9: end for

10: return new population

11: end function

- **Modèle énergétique**

Généralement, CPU, mémoire, capacité du disque dur et bande passante du réseau consommation d'énergie décidé, il a une priorité absolue. Nous supposons que la machine physique du centre de données Cloud a trois modes : le mode inactif, le mode actif et le mode veille. De nombreux documents montrent que la consommation d'énergie moyenne d'une machine physique inactive est de 70 % de celle fonctionnant à pleine vitesse. Ainsi, afin de réduire la consommation énergétique globale, nous devons convertir la machine physique inactive machine en mode veille. L'utilisation du processeur a une relation linéaire classique avec la charge de travail de l'ensemble du système [51]. C'est-à-dire que l'utilisation du processeur est au nom de l'efficacité dans une certaine mesure. Le modèle de l'énergie utilisée a été exprimé par l'équation suivante [52], [53], [54].

$$\boldsymbol{Ej = E_j^{max} * \left(E_j^{idle} + (0.3 * Uj)\right)} \qquad (3.8)$$

- De plus, nous trouvons le meilleur individu parmi la nouvelle population et le comparons avec le meilleur actuel. Si le meilleur individu de la nouvelle population est meilleur que le meilleur individu actuel, nous définissons le nouveau comme le meilleur individu actuel. Pour la prochaine itération de la boucle de génération, nous continuons avec la nouvelle génération et éliminons l'ancienne puisque la

nouvelle génération se compose de meilleurs individus que l'ancienne population. Le pseudocode de notre algorithme principal est donné dans l'algorithme suivant**.**

Algorithme : Genetic Algorithm

Require: VM demands, PM resources

Ensure: Assignment of VMs to PMs

1: Generate a population of POPSIZE number of random individuals,

POP;

2: while THE TERMINATION CONDITION is not true do

3: for each individual i in POP do

4: calculate energy value f (i)

5: end for

6: for each Individual i in POP do

7: invoke the Selection Operation, that is using of uniform selection with respect to the rank Technique to select another individual to pair

8: end for

9: for each pair of parents do

10: use single point Crossover Operation to produce an offspring

11: end for

12: for each offspring do // offspring represent descendants

13: apply Mutation Operation according to mutation rate

14: end for

15: find the best individual among offspring, new Best

16: if new Best is better than current best individual then

17: replace current best individual with new Best

18: end if

19: end while

20: return best individual

21: end Function

3.5. Conclusion

Dans ce chapitre nous avons présenté le problème de placement des machines virtuelles dans les centres de données Cloud. Ce problème étant de complexité NP-complet, nous avons utilisé une métaheuristique (l'algorithme génétique) afin d'avoir un placement qui minimise l'énergie consommée par les différents hôtes des centres de données Cloud.

Dans le chapitre suivant, nous présenterons l'implémentation et les résultats expérimentaux de notre stratégie en décrivant également l'environnement, le langage de programmation ainsi que le simulateur Cloud utilisé.

CHAPITRE 4: Implémentation

4.1. Introduction

L'objectif de ce chapitre est de décrire notre Implémentation dans ce PFE. Cette implémentation consiste à utiliser la métaheuristique « algorithme génétique » pour résoudre le problème de placement des machines virtuelles dans les machines physique des centres de données Cloud. Nous présentons notamment, le langage java et l'environnement de développement IDE NetBeans, ainsi que le simulateur CloudSim Plus (son architecture et ses classes fondamentales).

Nous présentons également l'IHM que nous avons développée ainsi que les résultats de simulation obtenus.

4.2. Langage Et Environnement De Développement

Ce projet a été implémenté et simulé sur une machine avec un processeur intel(R) core i5, une vitesse de 2.30 **GHZ** avec une capacité de mémoire de 6 GB sous le système d'exploitation windows10 de 64bits, et avec les outils de développement suivants :

4.2.1. Langage de programmation Java

4.2.1.1.Langage de Programmation Java

Java est un langage de programmation créé par James Gosling, un développeur de Sun Microsystems en 1991. Suivant Java développé par Sun Microsystems et est largement utilisé pour créer du contenu exécutable qui peut être distribué sur le réseau. Java est un langage de programmation orienté objet avec des éléments tels que C++ et d'autres langages qui ont des bibliothèques adaptées à l'environnement Internet [55]. Java Progamming Language appartient à la catégorie programmation de haut niveau où le langage est facilement compris par les utilisateurs Par rapport au langage C++[56].Java est un langage de programmation qui peut être exécuté sur une variété d'ordinateurs, y compris les téléphones portables. Ce langage a été créé à l'origine par James Gosling alors qu'il était encore chez Sun Microsystems et fait actuellement partie d'Oracle et est sorti en 1995. [57].

4.2.1.2.Pourquoi Nous Avons Utilisé Java

Java est un langage de programmation orienté objet. La programmation orientée objet a été remplacée par la programmation structurée car il présente de nombreux avantages dans la gestion de projets extraordinaires généralement complexe. La programmation utilise un langage

orienté objet suivre les concepts orientés objet offre principalement flexibilité, convivialité et facilité de maintenance[58].

4.2.2. Environnements de développement

NetBeans 8.2 : est un environnement de développement intégré (IDE) basé sur Java[59].IDE est le produit utilisé pour bien programmer, écrire du code, compiler, rechercher des erreurs et distribuer des programmes. Alors que la plate-forme NetBeans est un module qui est le cadre initial/la fondation profonde construire des applications de bureau de grande taille[58].Ce logiciel est en cours de développement conjoint, gratuit NetBeans est un projet de code ouvert réussi avec un très large éventail d'utilisateurs, une communauté grandissante, et compte près de 100 partenaires. Sun Microsystems a fondé le projet de code ouvert NetBeans en juin 2000 et continue d'en être le sponsor principal [57].

4.3. Outils De Simulation De Cloud

Ces dernières années ont vu une tendance croissante au développement de plateformes de simulation d'événements discrets (DES) pour soutenir la prise de décision et la recherche liées au Cloud Computing. La complexité des environnements Cloud augmente avec l'échelle et l'hétérogénéité, ce qui pose un défi pour la gestion efficace des applications Cloud et des ressources du centre de données. Ces dernières années, il y a eu une augmentation significative du développement et de l'extension d'outils pour prendre en charge DES pour le Cloud Computing, ce qui a donné lieu à plusieurs d'outils qui varient en termes d'utilité et de fonctionnalités. Grâce à une littérature disponible, cette partie fournit une vue d'ensemble et une analyse des fonctionnalités à plusieurs niveaux de certains outils DES pour les environnements de Cloud Computing [60].

Le tableau 4.1 lister les outils actuellement identifiés qui prennent en charge (DES) pour le Cloud Computing, par ordre alphabétique.

Bazaar Extension	*CloudSimDisk*	*ICanCloud*
CACTOSim	*CloudSimSDN*	*iFogSim*
CDOSim	*CMCloudSimulator*	*MDCSim*
CEPSim	*DartCSim*	*MR-CloudSim*
Cloud2Sim	*DCSim*[(2)]	*NetworkCloudSim*

CloudAnalyst	*DCSim*[3]	*SimGrid*
CloudEXP	*DISSECT-CF*	*SimIC*
CloudNetSim++	*EMUSim*	*SPECI*
CloudReports	*GDCSim*	*TeachCloud*
CloudSched	*GreenCloud*	*UCloud*
CloudSim	*GroudSim*	*WorkflowSim*

Tableau 3.4 Outils des identifiés liés au Cloud Computing.

4.3.1. CloudSim

CloudSim est une plate-forme de simulation Java open source et extensible pour activer la modélisation, la simulation et l'expérimentation continues des services de Cloud Computing et d'application. CloudSim est la plate-forme de facto de choix pour le développement d'outils de simulation open source, 18 des outils analysés étaient des dérivés ou des extensions de CloudSim.

L'architecture CloudSim suit une approche en couches. À la couche fondamentale, la gestion des applications, les hôtes de machines virtuelles et les états du système dynamique sont fournies. En étendant la fonctionnalité de provisionnement de MV de base, l'efficacité de différentes stratégies à cette couche peut être étudiée. À la couche supérieure, le code utilisateur représente les entités de base pour les hôtes, et grâce à des entités extensibles de cette couche, on peut permettre à l'application de générer des demandes en utilisant une variété d'approches et de configurations, de scénarios de Cloud de modèle, d'implémenter des applications personnalisées, etc.

Dans l'implémentation CloudSim, il n'y a pas d'entités réelles disponibles pour simuler des entités de réseau, telles que des routeurs ou des commutateurs. Au lieu de cela, la latence du réseau entre deux composants est simulée sur la base des informations stockées dans une matrice de latence. Le moteur de gestion d'événements de CloudSim utilise les informations de latence du réseau interstitielles pour induire des retards dans la transmission de messages aux entités. Ce retard est exprimé dans les unités de temps de simulation telles que les millisecondes. Le cadre CloudSim fournit des modèles et des entités de base pour valider et évaluer l'approvisionnement soucieux de l'énergie des techniques ou des algorithmes[61].

4.3.2. CloudSim Plus

CloudSim Plus un nouveau cadre complet, repensé, hautement extensible et moderne Java 8 Framework pour la modélisation et la simulation de l'infrastructure, des services, des mécanismes et des algorithmes sous-jacents. CloudSim Plus permet aux chercheurs de modéliser et de simuler différents scénarios de Cloud, en les mettant en œuvre à l'aide de Java. Ces scénarios peuvent être utilisés pour expérimenter des solutions existantes et potentiellement nouvelles pour les problèmes.

- **Définition**

CloudSim Plus est basé sur CloudSim 3. Il a passé un processus de repens et de réingénierie étendu pour fournir un cadre de simulation de Cloud mis à jour, moderne, très extensible et plus facile à utiliser. Ces changements visent à permettre la maintenabilité du projet durable pour une évolution à long terme. Pour atteindre ces objectifs, CloudSim Plus est fondé sur plusieurs mesures de conception de logiciels et d'ingénierie, les principes et les pratiques tels que le couplage, la cohésion, les modèles de conception, les principes solides et d'autres comme ne vous répétez pas (DRY) et KISS[62].

- **Architecture**

CloudSim Plus est un composé de différents modules. L'API CloudSim Plus est le module principal qui représente l'API du Framework de simulation. Il s'agit du module unique requis pour permettre la mise en œuvre des expériences de simulation de Cloud. (La figure 4.1) montre une vue simplifiée de sa structure de package. Les packages avec une couleur plus forte contiennent des fonctionnalités exclusives de CloudSim Plus. Ces plus légers ont été introduits juste pour fournir une meilleure organisation et une meilleure séparation des préoccupations (SOC) [63]. Mais contenant des classes a été étendue refactorings et reconcevoir.

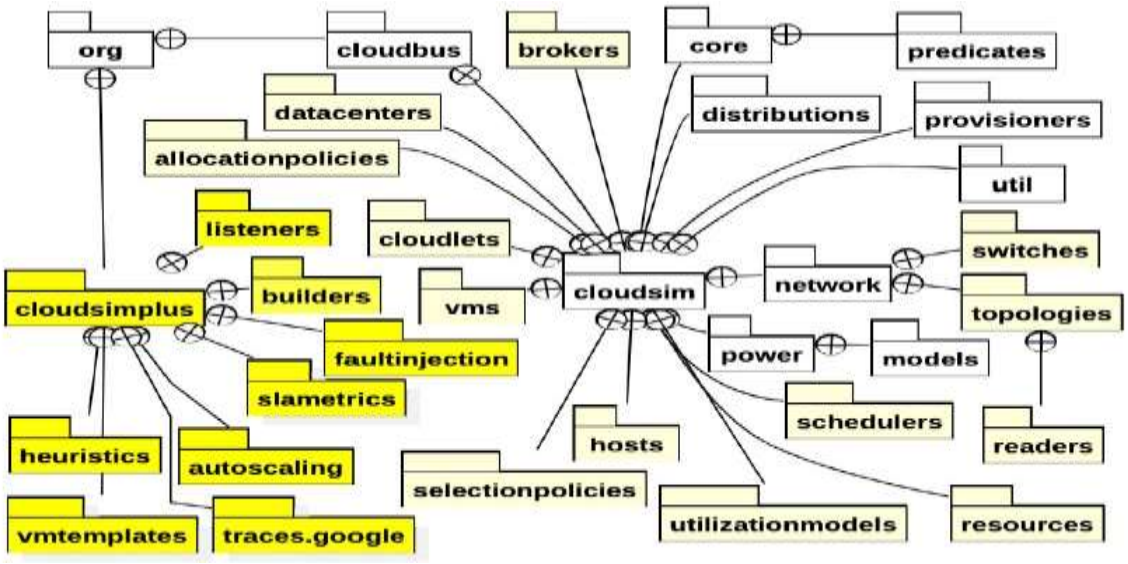


Figure 29.1 Structure du package API CloudSim Plus[63].

Les packages les plus pertinents de CloudSim Plus et les classes qu'ils contiennent peuvent être brièvement décrites ci-dessous:

Distributions: classes qui fournissent la génération de nombres pseudo aléatoires suite à plusieurs distributions statistiques utilisées par l'API de simulation. De plus, ils peuvent être utilisés par les développeurs mettant en œuvre leurs propres simulations[62].

Network: classes de création d'une infrastructure réseau Datacenter permettant des simulations de réseau.

Power: classes pour permettre des simulations de pouvoir, y compris Modèles de consommation d'énergie qui peuvent être étendus par les développeurs créant leurs simulations.

Le projet CloudSim d'origine est vient avec des classes qui fournissent des fonctionnalités de base pour implémenter rapidement les scénarios de simulation, tels que des modèles d'utilisation de ressources comme RAM, CPU et Bande passante. Il dispose d'un ensemble de classes de base et de quelques interfaces qui doivent être étendues par ses utilisateurs afin d'introduire des fonctionnalités spécifiques pour leurs simulations. Visant une meilleure organisation de classes et d'interfaces dans CloudSim Plus, les packages légèrement mis en surbrillance de (la figure 4.1) Ont été introduits, regroupant des classes et des interfaces avec des objectifs spécifiques dans des packages bien définis[62].

Ces nouveaux packages sont décrits ci-dessous:

(1) **hosts, datacenter, vms, Cloudlets**: classes de groupe qui fournissent différentes implémentations pour les centres de données, les hôtes, les machines virtuelles et les Cloudlets (applications), y compris les implémentations de puissance et composants éveillés du réseau[62].

(2) **Allocation policies:** des classes qui fournissent des mécanismes à un centre de données pour sélectionner un hôte pour placer ou migrer une machine virtuelle. Le Framework fournit une politique la moins ajustée appelée vm allocation Policy Simple qui sélectionne l'hôte avec des cœurs de processeurs moins disponibles pour placer une machine virtuelle donnée[62].

(3) **brokers**: Classes qui agissent au nom d'un client Cloud, assistant à ses demandes de création et de destruction de Cloudlets et de machines virtuelles, en attribuant de telles applications à des machines virtuelles spécifiques. Ces courtiers peuvent mettre en œuvre des algorithmes de prise de décision pour hiérarchiser la soumission des applications au Cloud, définir comment une machine virtuelle est sélectionnée pour exécuter une application donnée, etc. Il comprend un Datacenter Broker exclusif qui utilise une heuristique de recuit simulée pour mapper Cloudlets à des VM[62].

(4) **schedulers**: classes pour planifier l'exécution de plusieurs Cloudlets dans une machine virtuelle et l'exécution de plusieurs machines virtuelles dans un hôte. Il comprend des planificateurs partagés en temps et partagés dans l'espace, en plus d'une implémentation exclusive du planificateur complètement équitable utilisé dans les versions récentes du noyau Linux[62].

(5) **resources:** des classes qui représentent des ressources Cloud physiques et logiques telles que les disques durs, les cœurs de processeur (traitements de traitement ou simplement PES), la RAM, la bande passante et les fichiers utilisateur[62].

(6) **Utilization models:** classes qui modélisent l'utilisation de ressources telles que le CPU, la RAM et la bande passante, définissant comment une ressource donnée est utilisée par une application le long du temps[62].

Les packages exclusifs CloudSim Plus sont présentés de couleur plus forte et sont brièvement décrits ci-dessous:

(1) **listeners:** classes utilisées pour obtenir des notifications pendant l'exécution de la simulation, puis surveiller le scénario de simulation. En utilisant de telles fonctionnalités de surveillance, il est possible, par exemple, d'allouer des machines virtuelles à la demande, lorsqu'une condition spécifique est remplie[62].

(2) **heuristics**: les classes qui fournissent la base pour mettre en œuvre des heuristiques qui pourraient être utilisées à différentes fins telles que la cartographie des Cloudlets aux VM et les décisions de migration des machines virtuelles[62].

(3) **builders**: classes qui implémentent le modèle de conception du constructeur [64] fonctionnant comme usines d'objets qui facilitent la création de plusieurs objets de simulation tels que les hôtes, les machines virtuelles et les Cloudlets.

- CloudSim Plus repose sur plusieurs classes pour fournir ses fonctionnalités. (La figure 4.2) présente un diagramme UML réduit qui indique les principaux qui doivent être utilisés pour modéliser et exécuter une simulation. Ces classes fournissent des fonctionnalités de base qui peuvent être étendues par un chercheur.

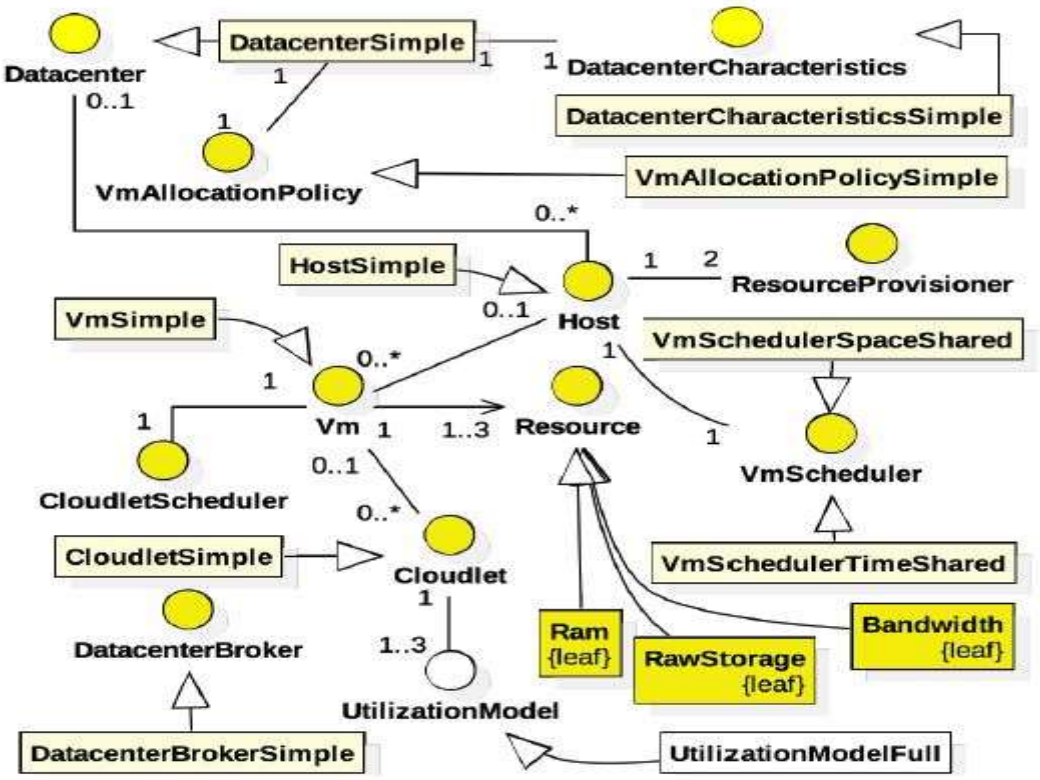


Figure 30.2 Principales classes impliquées dans la création de simulations à l'aide de CloudSim Plus[64].

Une description des classes principales utilisées pour créer une simulation est présentée ci-dessous.

(1) **Datacenter, Datacenter Characteristics et Vm Allocation Policy:** un centre de données contient un ensemble de machines physiques (hôtes, serveurs ou simplement PMs) qui fournissent ensemble l'infrastructure de Cloud de base. Chaque centre de données a des attributs qui définissent ses caractéristiques, telles que les coûts associés à différentes ressources physiques de ses hôtes. Ces attributs sont définis par un objet Datacenter Characteristics. Pour chaque centre de données créé, une instance Vm Allocation Policy doit être définie. Cet objet décide quel PM hébergera chaque machine virtuelle. Le cadre fournit la mise en œuvre de Vm Allocation Policy Simple, une politique de pire ajustement qui alloue les machines virtuelles à l'hôte avec la plupart des cœurs de processeurs disponibles (Pes) [62].

(2) **Host, Pe et Vm Scheduler:** une hôte représente une machine physique (PM) et pour chaque PM, une liste d'éléments de traitement (Pes) doit être définie (les cœurs de processeur de la machine). Comme le PM peut héberger des machines virtuelles, il est également nécessaire à un algorithme de planification qui sera utilisé pour gérer l'exécution simultanée de plusieurs machines virtuelles dans les Pes hôte. Il existe différents Vm Schedulers, tels que le temps et l'espace, qui peuvent être utilisés[62].

(3) **Datacenter Broker:** représente un logiciel qui agit au nom d'un client Cloud, recevant des demandes et effectuant des actions nécessitant des actions pour y assister. Ces actions incluent la soumission des machines virtuelles à allouer à l'intérieur d'une hôte d'un centre de données, soumettant des Cloudlets (applications) à exécuter dans certaines des machines virtuelles créées, en prenant des décisions sur la machine virtuelle à sélectionner pour placer un Cloudlet donné, etc [62].

(4) **Vm et Cloudlet Scheduler:** Un objet VM représente une machine virtuelle qui s'exécute à l'intérieur d'un hôte et exécutera des applications (Cloudlets). Un Cloudlet Scheduler définit la façon dont l'exécution concomitante de plusieurs applications est planifiée dans une machine virtuelle. Il suit le même raisonnement de Vm Scheduler et il existe les mêmes implémentations de base disponibles et un planificateur complètement juste, comme décrit précédemment [62].

(5) **Cloudlet et Utilization Model:** un Cloudlet représente une application qui s'exécutera dans une machine virtuelle, résolument définie en termes de caractéristiques, telles que le nombre de millions d'instructions à exécuter, le nombre de Pes et de modèles d'utilisation requis pour CPU, RAM et bande passante. Chaque objet Utilization Model définit comment une ressource donnée sera utilisée par le Cloudlet le long du temps. Certaines implémentations de base de Utilization Model sont fournies, telles que Utilization Model Full, ce qui indique qu'une ressource disponible donnée sera utilisée à 100% tout le temps[62].

- **CloudSim Plus Améliorations Par Rapport À CloudSim**

a. Améliorations d'extensibilité.

b. Duplication de code réduite.
c. Tests et couverture de code [62].

- **Les caractéristiques et les avantages de CloudSim Plus**

a. Arrivée dynamique des Cloudlets et des machines virtuelles et priorisation des Cloudlets.
b. Datacenter Brokers très extensibles et nouveaux.
c. Module réseau rénové et nouvel ensemble d'interfaces.
d. Listeners d'événements.
e. Constructeur de Classes.
f. Tests d'intégration.
g. Métriques de qualité de la conception du logiciel[62].

4.4. IHM développée

La version de CloudSim Plus n'a pas d'interface graphique, il utilise le mode console. Donc nous avons créé une IHM pour faciliter l'accès ainsi que la manipulation du simulateur.

Figure 31.3 La page d'accueil.

Après cela, nous suivons les étapes suivantes pour atteindre le résultat souhaité à la fin.

4.4.1. VM Frame

Illustrer à la figure (4.4). Permettent à l'utilisateur de gérer la liste des machines virtuelles, « **ADD** » pour ajouter une machine physique à la liste des MVs et au même temps, elle sera affectée à une machines physique d'une manière aléatoire, l'utilisateur doit remplir les informations nécessaires pour créer une machines Virtuelle, (Nombre de MV, choisir sa capacité, sa nombre des cœurs, la capacité du mémoire, et la capacité du Bande passante), « **Next** » pour passer à une autre interface qui il est Host Frame.

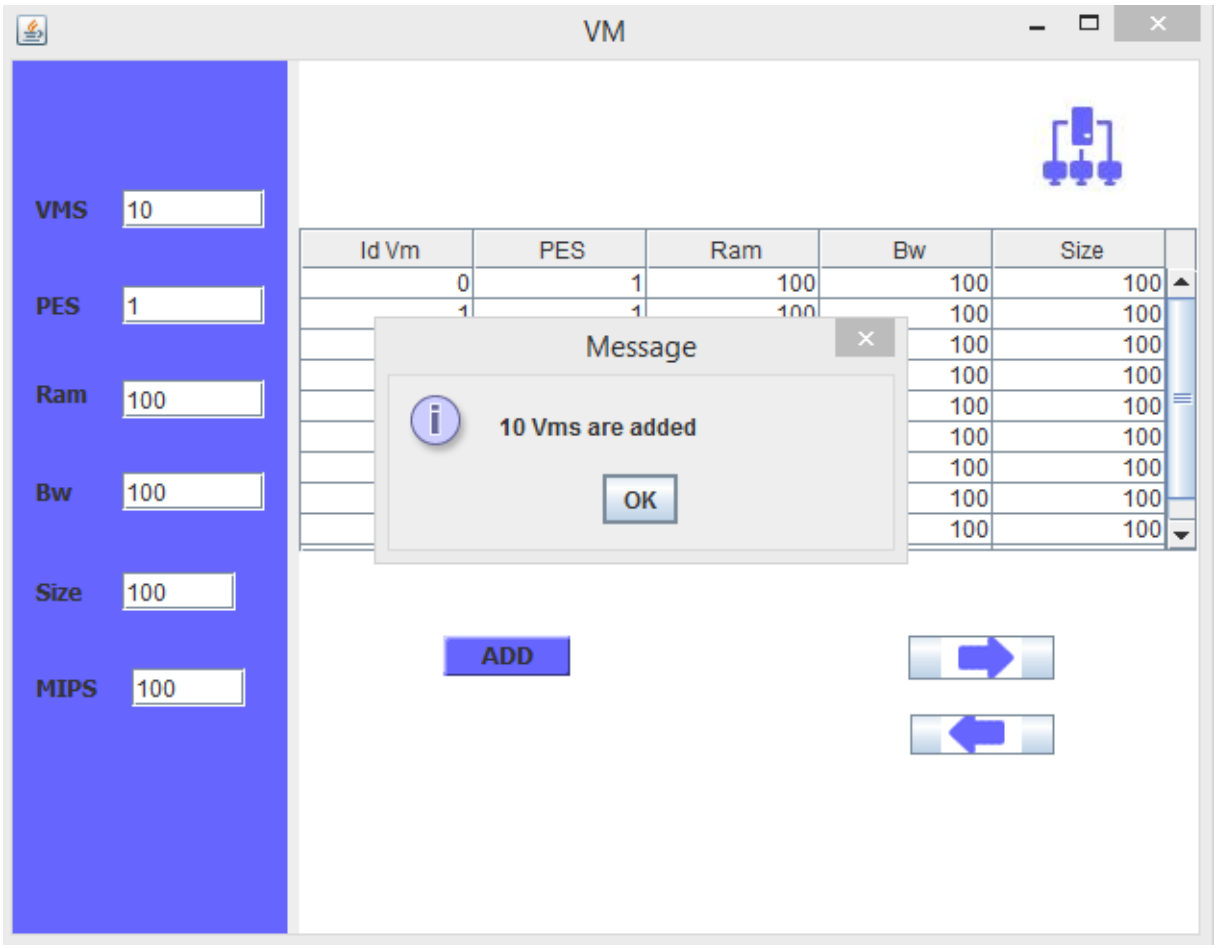


Figure 32.4 LA Gestion des Machine Virtuelles.

4.4.2. Host Frame

Illustré à la figure (4.5), les mêmes étapes à suivre pour la création et la suppression des machines Virtuelles, en cliquant sur le bouton « **ADD** », la machine sera ajoutée à la liste des PMs, « **Next** » pour passer à une autre interface qui il est Optimisation de placement et « **Previous** » pour retourner arrière a l'interface VM Frame.

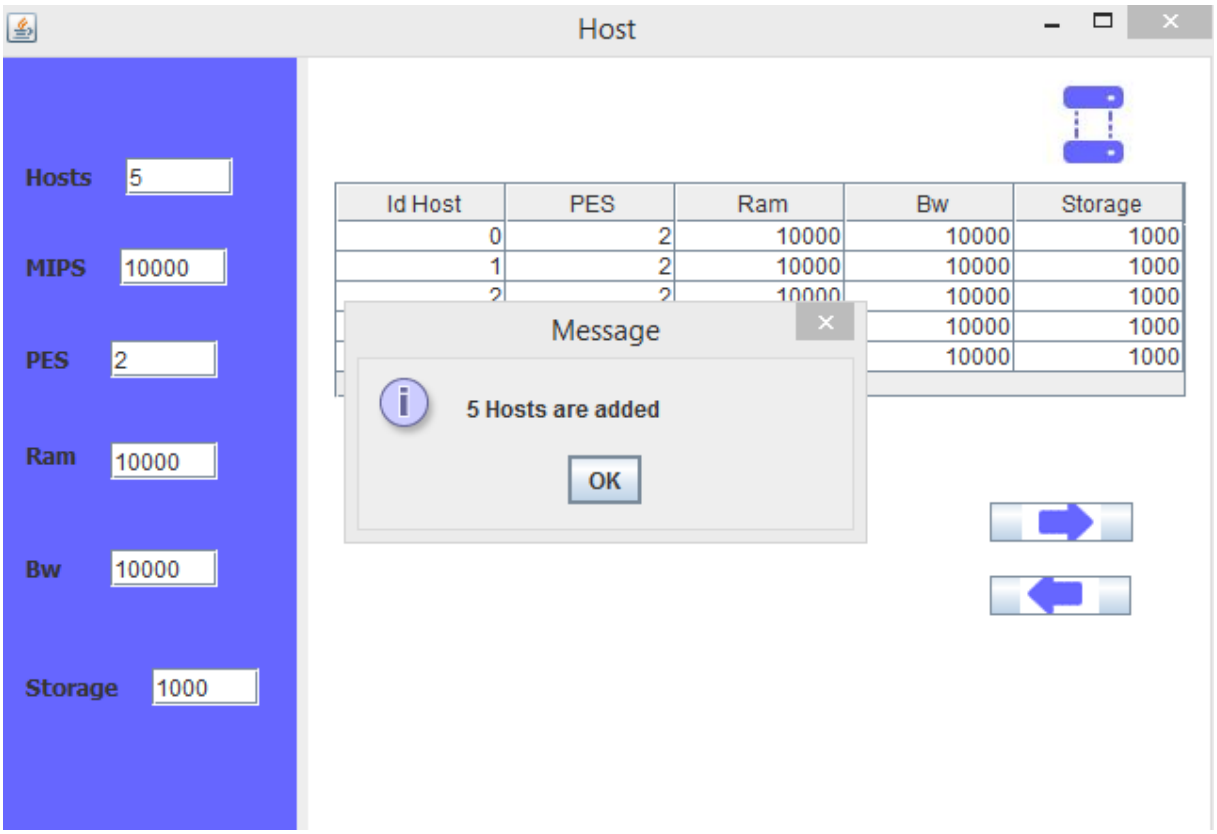


Figure 33.5 Gestion des Machines Physiques.

4.4.3. Optimisation de placement

Illustré à la figure 4.6, Où il y a un bouton  qui signifie « **optimiser le placement** », qui permet à l'utilisateur de lancer la simulation de notre approche génétique, et d'insérer la taille de population et le nombre de génération, pour obtenue à un placement optimale pour les machines virtuelles sur les machines physique, afin d'optimiser la consommation des ressources, et d'énergies. Elle affiche aussi chaque machine physique et les ensembles des machines virtuelles qui y sont affectés.

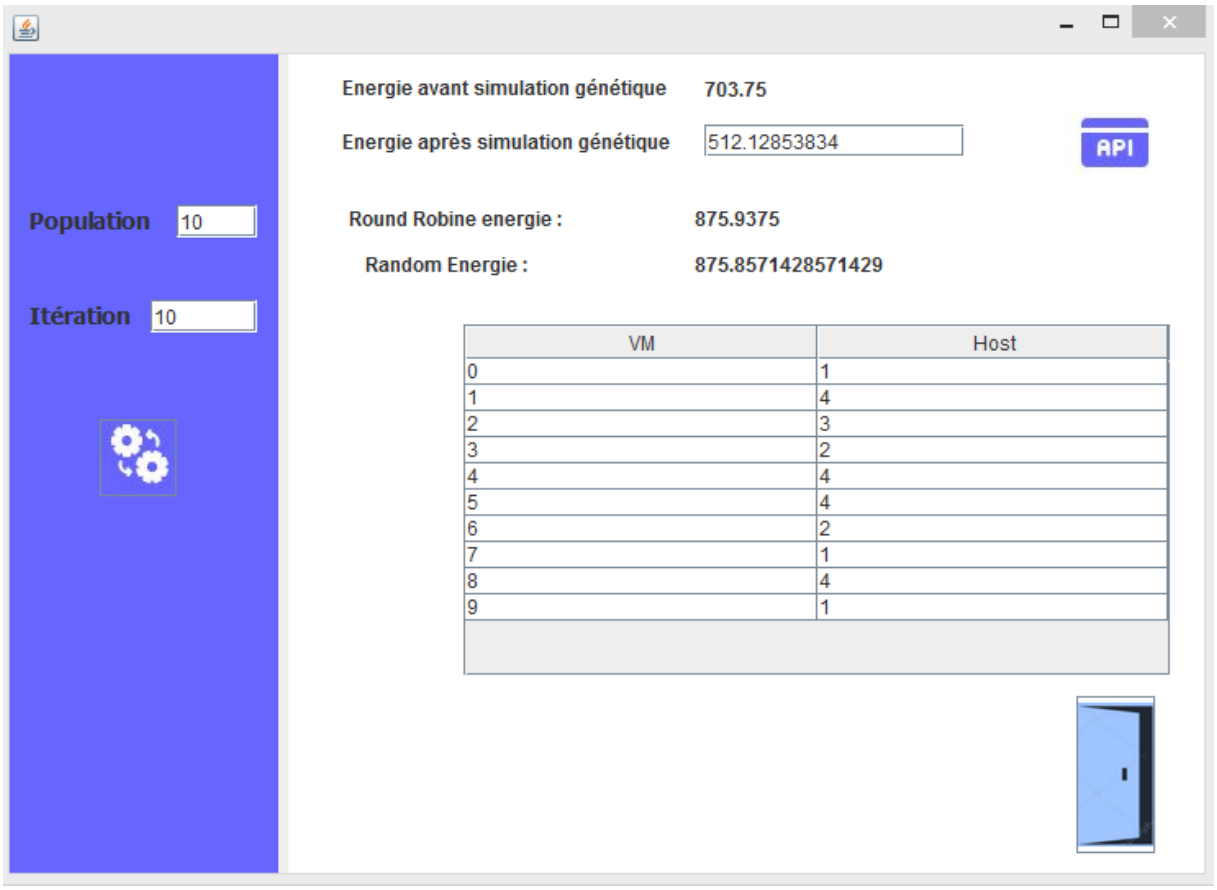


Figure 34.6 Optimisation de Placement.

4.5. Analyse des résultats

Voici quelques résultats de simulation de notre méthode :

Energie (watt)	Population = 10	Population = 20	Population = 50	Population = 100
VM= 10, PM=5	703.75	525.75	528.75	528.5
VM= 20, PM=10	1232.5	1232.4999	1057.5	1057.4999
VM= 50, PM=20	2643.75	2993.75	2993.7499	2818.75
VM= 100, PM=50	7212.5	7212.5	6862.5	6687.5

Tableau 4.5 Énergie consommé en fonction du nombre de MVs, nombre de PMs et la taille de population.

Les résultats de simulation montre que notre algorithme (AG) améliore la consommation d'énergie par rapport aux : placement aléatoire (Random) et Round Robin (RR). Tableau (4.6) :

	Énergie consommée (watt)		
Algorithme	Algorithme génétique	Round Robin	Random
VM= 10, PM=5	703.75	875.9375	875.8973
VM= 20, PM=10	1232.5	1775.8750	1775.7546
VM= 50, PM=20	2818.75	3504.6875	3504.5660
VM= 100, PM=50	7212.5	8759.375	8558.93

Tableau 5. 6 Énergie consommée pour les différentes méthodes d'optimisation.

Après l'analyse de résultats on a conclu l'efficacité de l'algorithme utilisé par rapport les autres méthodes (Random, Round Robin). Et le graphe suivant montre la différence entre eux :

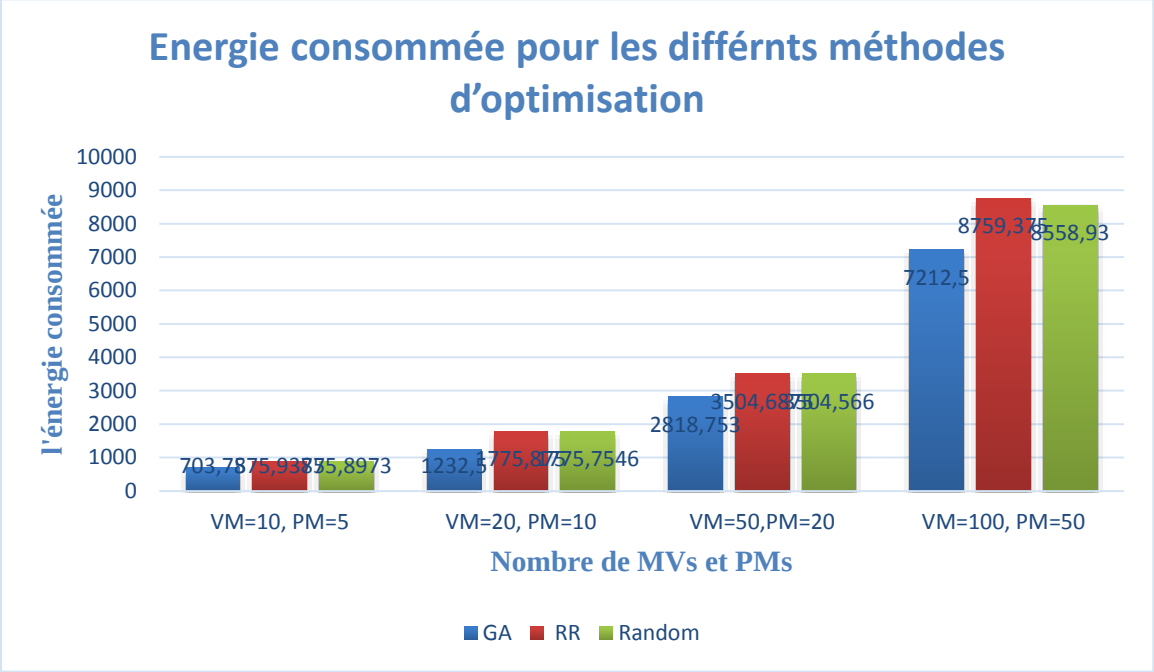


Figure 35.7 Énergie consommée pour les différentes méthodes d'optimisation

4.6. Conclusion

Ce chapitre a été consacré à la présentation de notre implémentation dans le cadre de ce PFE. Nous avons commencé par présenté les outils de simulations utilisés, à savoir, le langage JAVA, l'IDE NetBeans ainsi que le simulateur CloudSim Plus.

Notre approche en ce qui concerne la consommation d'énergie a été montrée par tous les étapes et les résultats de simulation.

Conclusion générale et perspectives

Dans ce mémoire nous avons essayé de résoudre un problème d'optimisation combinatoire de complexité NP-Complet. Il s'agit du problème de placement des machines virtuelles dans les centres de données Cloud. Pour cela, on a utilisé une métaheuristique dont le but est de satisfaire les exigences des fournisseurs du Cloud qui consiste à placer les machines tout en minimisant l'énergie dépensée par leurs centres de données. Nous avons également utilisé l'Algorithme Génétique avec des paramètres et des opérateurs personnalisés.

Ce travail nous a permis d'apprendre d'une manière détaillée le fonctionnement du Cloud Computing, la notion de virtualisation et sa relation avec l'économie d'énergie au sein du Cloud. Nous avons encore appris le fonctionnement du l'outil CloudSim plus ainsi que le rôle de chaque classe et de chaque package pour résoudre ce problème.

Les résultats de simulation obtenus étaient assez satisfaisantes vu la crédibilité de l'algorithme génétique pour résoudre les problèmes d'optimisation. Dans notre étude, nous avons traité un seul objectif qui était la minimisation de l'énergie consommée dans les centres de données Cloud. Dans les travaux futurs, nous visons à étendre notre étude afin d'inclure d'autre objectifs. Il s'agit de la technique d'optimisation multi-objective qui consiste à traiter simultanément un ensemble d'objectifs. C'est une technique assez motivante, particulièrement lorsque les objectifs sont contradictoires.

Reference

[1] V. Rajaraman, « Cloud computing », *Resonance*, vol. 19, n° 3, p. 242-258, 2014.

[2] Y. Jadeja et K. Modi, « Cloud computing - concepts, architecture and challenges », in *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, mars 2012, p. 877-880. doi: 10.1109/ICCEET.2012.6203873.

[3] J. Brodkin, « Gartner: Seven cloud-computing security risks », *Infoworld*, vol. 2008, p. 1-3, 2008.

[4] K. Stanoevska-Slabeva et T. Wozniak, « Cloud basics–an introduction to cloud computing », in *Grid and cloud computing*, Springer, 2010, p. 47-61.

[5] R. Smith, « Computing in the cloud », *Res.-Technol. Manag.*, vol. 52, n° 5, p. 65-68, 2009.

[6] P. Mell, T. Grance, et others, « The NIST definition of cloud computing », 2011.

[7] F. Liu *et al.*, « NIST cloud computing reference architecture », *NIST Spec. Publ.*, vol. 500, n° 2011, p. 1-28, 2011.

[8] B. Jacob, M. Brown, K. Fukui, N. Trivedi, et others, « Introduction to grid computing », *IBM Redb.*, p. 3-6, 2005.

[9] L. Malhotra, D. Agarwal, et A. Jaiswal, « VIRTUALIZATION IN CLOUD COMPUTING », p. 6, 2014.

[10] R. Jain et S. Paul, « Network virtualization and software defined networking for cloud computing: a survey », *IEEE Commun. Mag.*, vol. 51, n° 11, p. 24-31, nov. 2013, doi: 10.1109/MCOM.2013.6658648.

[11] N. Grassa, « Cours Virtualisation et Cloud », p. 26.

[12] L. Qian, Z. Luo, Y. Du, et L. Guo, « Cloud computing: An overview », in *IEEE international conference on cloud computing*, 2009, p. 626-631.

[13] T. Diaby et B. B. Rad, « Cloud computing: a review of the concepts and deployment models », *Int. J. Inf. Technol. Comput. Sci.*, vol. 9, n° 6, p. 50-58, 2017.

[14] P. Sareen, « Cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud », *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, n° 3, 2013.

[15] M. Malathi, « Cloud computing concepts », in *2011 3rd International Conference on Electronics Computer Technology*, avr. 2011, vol. 6, p. 236-239. doi: 10.1109/ICECTECH.2011.5942089.

[16] A. Apostu, F. Puican, G. Ularu, G. Suciu, G. Todoran, et others, « Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud », *Recent Adv. Appl. Comput. Sci. Digit. Serv.*, vol. 2103, 2013.

[17] T. Dillon, C. Wu, et E. Chang, « Cloud Computing: Issues and Challenges », in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, Australia, 2010, p. 27-33. doi: 10.1109/AINA.2010.187.

[18] X.-S. Yang et S. Deb, « Engineering optimisation by cuckoo search », *Int. J. Math. Model. Numer. Optim.*, vol. 1, n° 4, p. 330-343, 2010.

[19] J. Puchinger et G. R. Raidl, « Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification », in *International work-conference on the interplay between natural and artificial computation*, 2005, p. 41-53.

[20] A. H. Land et A. G. Doig, « An automatic method for solving discrete programming problems », in *50 Years of Integer Programming 1958-2008*, Springer, 2010, p. 105-132.

[21] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[22] M. Douiri, S. Elbernoussi, et H. Lakhbab, « Cours des méthodes de résolution exactes heuristiques et métaheuristiques », *Univ. Mohamed V Fac. Sci. Rabat*, p. 5-87, 2009.

[23] A. Alaoui, « Application des techniques des métaheuristiques pour l'optimisation de la tâche de la classification de la fouille de données », PhD Thesis, USTO, 2012.

Referance

[24] M. HADJI, « Placement Dynamique de VMs dans le Cloud: Divisez votre Capex par deux! ».

[25] A. Lambert, « Résolution de programmes quadratiques en nombres entiers », PhD Thesis, Conservatoire National des Arts et Métiers (CNAM), 2009.

[26] M. Benomar, « Devant le jury composé de »:, p. 62.

[27] S. Digabel, « Introduction aux métaheuristiques », *Support Cours Ecole Polytech. Montr. Fr.*, 2014.

[28] I. Boussaid, « Perfectionnement de métaheuristiques pour l'optimisation continue », PhD Thesis, Paris Est, 2013.

[29] « Optimisation-de-la-Qos-dans-un-reseau-de-radio-cognitive-en-utilis-lalgorithme-de-la-recherche-cuckoo-search.PDF ».

[30] C. Gagne, « Algorithmes évolutionnaires appliqués à la reconnaissance des formes et à la conception optique », 2005.

[31] J. Koza, « On the programming of computers by means of natural selection », *Genet. Program.*, 1992.

[32] W. Banzhaf, P. Nordin, R. E. Keller, et F. D. Francone, *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., 1998.

[33] S. Binitha, S. S. Sathya, et others, « A survey of bio inspired optimization algorithms », *Int. J. Soft Comput. Eng.*, vol. 2, n° 2, p. 137-151, 2012.

[34] S. Assar, « Vient de paraître: Cloud computing: Décider, concevoir, piloter, améliorer (par R. Hennion, H. Tournier, E. Bourgeois) », *Systèmes Inf. Manag.*, vol. 18, n° 1, p. 5, 2015.

[35] S. Mirjalili, J. Song Dong, A. S. Sadiq, et H. Faris, « Genetic algorithm: Theory, literature review, and application in image reconstruction », *Nat.-Inspired Optim.*, p. 69-85, 2020.

[36] J. F. Frenzel, « Genetic algorithms », *IEEE Potentials*, vol. 12, n° 3, p. 21-24, 1993.

[37] K. Sastry, D. Goldberg, et G. Kendall, « Genetic algorithms », in *Search methodologies*, Springer, 2005, p. 97-125.

[38] S. Mirjalili, « Genetic algorithm », in *Evolutionary algorithms and neural networks*, Springer, 2019, p. 43-55.

[39] A. Shukla, H. M. Pandey, et D. Mehrotra, « Comparative review of selection techniques in genetic algorithm », in *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, 2015, p. 515-519.

[40] P. Sharma et A. Wadhwa, « Analysis of selection schemes for solving an optimization problem in genetic algorithm », *Int. J. Comput. Appl.*, vol. 93, n° 11, 2014.

[41] P. Kora et P. Yadlapalli, « Crossover operators in genetic algorithms: A review », *Int. J. Comput. Appl.*, vol. 162, n° 10, 2017.

[42] M. HADJI, « Placement Dynamique de VMs dans le Cloud: Divisez votre Capex par deux! ».

[43] L. Li et K. Liu, « Guarantee-aware cost effective virtual machine placement algorithm for the cloud », in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2017, p. 506-513.

[44] L. Mashayekhy, M. M. Nejad, et D. Grosu, « Cloud federations in the sky: Formation game and mechanism », *IEEE Trans. Cloud Comput.*, vol. 3, n° 1, p. 14-27, 2014.

[45] M. M. Mamlouk, « Approche heuristique pour le placement des machines virtuelles dans un environnement infonuagique de grande taille », PhD Thesis, Ecole Polytechnique, Montreal (Canada), 2017.

[46] A. D. Tasci et W. C. Gartner, « Destination image and its functional relationships », *J. Travel Res.*, vol. 45, n° 4, p. 413-425, 2007.

[47] J. Kaplan, W. Forrest, et N. Kindler, « Revolutionizing data center energy efficiency, McKinsey & Company », *Online Httpwww Sallan Orgpdf-DocsMcKinseyDataCenterEfficiency Pdf*, 2008.

[48] E. Naone, « Technology overview conjuring clouds », *Technology Review*, vol. 112, n° 4. TECHNOL REV 1 MAIN ST, 13 FLR, CAMBRIDGE, MA 02142 USA, p. 54-56, 2009.

Referance

[49] Y. Song, Y. Sun, et W. Shi, « A two-tiered on-demand resource allocation mechanism for VM-based data centers », *IEEE Trans. Serv. Comput.*, vol. 6, n° 1, p. 116-129, 2011.

[50] C.-C. Lin, P. Liu, et J.-J. Wu, « Energy-efficient virtual machine provision algorithms for cloud systems », in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 2011, p. 81-88.

[51] A. Beloglazov et R. Buyya, « Energy efficient resource management in virtualized cloud data centers », in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, p. 826-831.

[52] Y. Gao, H. Guan, Z. Qi, Y. Hou, et L. Liu, « A multi-objective ant colony system algorithm for virtual machine placement in cloud computing », *J. Comput. Syst. Sci.*, vol. 79, n° 8, p. 1230-1242, déc. 2013, doi: 10.1016/j.jcss.2013.02.004.

[53] Y. C. Lee et A. Y. Zomaya, « Energy efficient utilization of resources in cloud computing systems », *J. Supercomput.*, vol. 60, n° 2, p. 268-280, mai 2012, doi: 10.1007/s11227-010-0421-3.

[54] S. Rawas, A. Zekri, et A. El Zaart, « Power and Cost-aware Virtual Machine Placement in Geo-distributed Data Centers »:, in *Proceedings of the 8th International Conference on Cloud Computing and Services Science*, Funchal, Madeira, Portugal, 2018, p. 112-123. doi: 10.5220/0006696201120123.

[55] « Java Courses & Tutorials », *Codecademy*. https://www.codecademy.com/catalog/language/java (consulté le 12 mai 2022).

[56] S. Sri Astuti, « PERANCANGAN SISTEM PARKIR MALL DENGAN KONSEP OBJECT ORIENTED PROGRAMMING (OOP) MENGGUNAKAN BAHASA PEMROGRAMAN JAVA DENGAN EDITOR NETBEANS 8.2 ».

[57] T. PRO, « PEMBUATAN SISTEM INFORMASI PEMBAYARAN BARANG PADA TOKO PRO KOMPUTER BERBASIS OBJECT ORIENTED PROGRAM (OOP) MENGGUNAKAN NETBEANS 8.2 ».

[58] O. O. P. Java, « PERANCANGAN SISTEM PEMBELIAN TIKET KONSER MENGGUNAKAN KONSEP PEMROGRAMAN BERBASIS OBJEK DENGAN EDITOR NETBEANS 8.2 ».

[59] « Implementasi_konsep_OOP_Sistem_Informasi_Laundry_dengan_NetBeans_8.2-with-cover-page-v2.pdf ».

[60] J. Byrne *et al.*, « A review of cloud computing simulation platforms and related environments », in *International Conference on Cloud Computing and Services Science*, 2017, vol. 2, p. 679-691.

[61] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, et R. Buyya, « CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms », *Softw. Pract. Exp.*, vol. 41, n° 1, p. 23-50, 2011.

[62] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. M. Inácio, et M. M. Freire, « CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness », in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, p. 400-406. doi: 10.23919/INM.2017.7987304.

[63] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[64] E. Gamma, R. Helm, R. Johnson, R. E. Johnson, et J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.