République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Centre Universitaire Belhadj Bouchaib d'Aïn-Témouchent

Faculté des Sciences

Département de Mathématiques et de l'Informatique

Mémoire

Pour l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Ingénierie des Données (RID)

Présenté par :

Mr. KHARABI salheddine abder rahmen

Mr. BENALLAL mohamed islam

Ordonnancement Multi-critères dans le Cloud Computing

Encadrant :

Mr. BOUAFIA zouheyr

Maitre Assistant "A" à C.U.B.B.A.T.

Soutenu le 19 juin 2019.

Devant le jury composé de :

Président :	Mr. BENOMAR (M.C.B)	C.U.B.B.A.T.
Examinateurs :	Mme. ABDERRAHIM (M.C.B)	C.U.B.B.A.T.
	Mr. BENOMAR (M.C.B)	C.U.B.B.A.T.
Encadrant :	Mr. BOUAFIA zouheyr (M.A.A)	C.U.B.B.A.T.

Dédicace

Je remercie "Allah" de m'avoir donné la volonté et le courage afin d'accomplir ce modeste travail .

Je dédie ce travail :

- À mes chers parents pour leur soutient et amour permanent.
- À mon frère **KHARABI YOUSSOUF** pour ses conseils et aides permanents.
- À mes amis et mes collègues.
- À tous ceux qui me sont chers.

KHARABI SALHEDDINE

Dédicace

Je tiens à remercier "Allah", le tout puissant, de m'avoir donné la force et la volonté pour dépasser toutes les difficultés.

Je dédie ce travail

À mes chères parents.

À mes sœurs et frères.

À toute ma famille.

À tous mes amis et mes collègues.

À tous ceux qui me sont chers.

BENALLAL ISLAM

Remerciements

Louange à Allah qui nous a orientées sur le bon chemin tout au long de ce modeste travail, et nous a ouvert les bonnes portes et les justes réflexes, sans sa miséricorde, ce travail n'aurait pas été accompli.

Nous profitons de cette occasion afin de transmettre nos sincères remerciements et nos profondes reconnaissances À :

- Nos formidables parents qui nous ont aider à surmonter les obstacles par leurs encouragements et leurs prières.
- Notre encadrant **Mr. BOUAFIA ZOUHEYR** qui nous a donné l'opportunité de réaliser ce sujet sous sa direction, et ses conseils fascinant ainsi que son temps consacré tout au long du travail.
- Tous nos enseignants qui nous ont permis d'arriver là où nous sommes aujourd'hui.
- Aux membres de jury **Mr. BENOMAR** et **Mme. ABDERRAHIM** qui nous honorent de leur présence en tant qu'examinateurs.

Nous tenons à remercier toutes les personnes qui ont participé de prés ou de loin au bon déroulement de ce travail.

Table des matières

Table des figures

Liste des tableaux

Liste des symboles

CIA Confidentiality Integrity Availability

CIS Cloud Information Service

CSA Cuckoo Search Algorithme

$FIFO/FCFS$ First In, First Out Ou First Come, First Served

$IaaS$ Infrastructure as a Service

IBM International Business Machines

IDE Integrated Development Environment

$MIPS$ Million Instructions Per Second

$PaaS$ Platform as a Service

QOS Quality Of Service

RR Round Robin algorithme

$SaaS$ Software as a Service

SJF Shortest Job First/ la tâche la Plus courte d'abord

SLA Service Level Agreement

VM Machine Virtuelle

Introduction générale

Le Cloud Computing s'impose très rapidement comme étant un standard pour l'hébergement des applications et les services logiciels. La plupart des entreprises, individus et même des corps gouvernementaux se réfugient vers le Cloud en raison de la réduction des prix, la facilité de développement et le stockage illimité. Plusieurs applications dans de nombreux domaines contiennent généralement de nombreuses tâches. Ces tâches requièrent pour leurs exécutions sur le Cloud un ordonnancement, c'est l'affectation des tâches aux ressources disponibles sur la base des exigences et les caractéristiques des tâches. C'est un processus très important pour un fonctionnement efficace du Cloud.

L'ordonnancement de tâche dans un Cloud est un problème difficile. Ce problème est d'autant plus difficile lorsqu'il y a plusieurs facteurs à prendre en compte, donc c'est un problème d'optimisation combinatoire, ou il est possible de trouver la solution optimale en utilisant des algorithmes ou des métaheuristiques simples.

Le problème d'ordonnancement des tâches est largement étudier dans de nombreux travaux. La majorité de ces travaux se sont concentrés uniquement sur l'optimisation de un métrique de qualité de service, souvent, le temps d'exécution.
L'objectif de notre travail est de développer un algorithme d'ordonnancement des tâches basé sur la métaheuristique « cukoo search » pour l'optimisation du temps d'exécution (makespan) , le coût et la consommation d'énergie.

Dans le premier chapitre, nous présenterons quelques notions de base sur le Cloud Computing et quelques concepts sur l'ordonnancement des tâches. Nous définissons aussi les algorithmes de base de l'ordonnancement comme FCFS (First Come First Served), Round Robin, Min-Min et SJF (Short Job First).

Dans le second chapitre, nous présenterons quelques notions fondamentales sur les métaheuristiques notamment leurs définitions, mode de fonctionnement, caractéristiques et une classification. Ensuite nous présenterons de manière détaillée la métaheuristique « cukoo search » utilisée dans notre projet.

Dans le dernier chapitre, nous expliquerons le choix du simulateur utilisé et l'approche adoptée pour la réalisation de notre travail, nous présenterons ensuite les résultats trouvés.

Chapitre I

Introduction sur le Cloud Computing

Sommaire

1 Introduction

Le Cloud Computing (nuage informatique), c'est un domaine qui regroupe un ensemble des techniques dont leur but principal est de permettre l'accès à distance à des ressources matérielles et/ou logicielles à travers des infrastructures réseaux, ce concept rend possibles la distribution des ressources informatiques sous forme de services et selon un modèle économique (payant, gratuit), ces services sont destinés aux domaines des applications scientifiques et commerciales...

Dans ce chapitre, nous allons présenter quelques notions de base sur le Cloud Computing, ensuite nous allons aborder quelques concepts sur l'ordonnancement des tâches et introduire les algorithmes de base dans ce domaine.

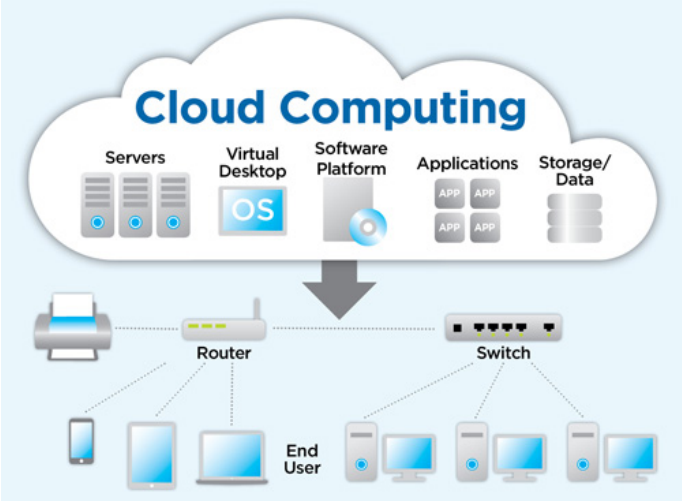

Figure I.1 – Cloud Computing [14]

2 Définitions

Dans cette partie, nous allons présenter quelques définitions trouvées dans la littérature.

L'Institut National des Normes et de la Technique (NIST) a proposé deux définitions pour le Cloud Computing :

- Le Cloud Computing est l'ensemble des disciplines, technologies et modèles commerciaux utilisés pour délivrer des capacités informatiques (logiciels, plateformes, matériels) comme un service à la demande.[18]

- Le Cloud Computing est un modèle d'accès à des ressources informatiques partagées et configurables (serveurs, stockage, applications...), depuis un accès réseau, à la demande, de manière simple, à partir de n'importe quel type d'appareils et depuis n'importe quel endroit.[18]

3 Notions de base

Dans cette partie, nous allons présenter quelques notions de base sur le Cloud Computing nécessaires à la compréhension de ce mémoire, notamment les centres de données, machines physiques, machines virtuelles, tâches, fournisseur...

Centre de données

Un centre de données (Datacenter) est une infrastructure qui regroupe un ensemble d'ordinateurs, d'espace de stockage, des serveurs, des commutateurs, des routeurs, générateur électrique, un système de ventilation et de refroidissement, connexion internet puissante. Cette infrastructure est utilisée comme un outil par les entreprises pour organiser, traiter, stocker leurs grandes quantités de données.[27]

Machine physique

Une machine physique (Hôte, Host) est un matériel physique doté d'une puissance de calcul, capacité de stockage que les machines virtuelles se servent de ses ressources (CPU, RAM, stockage...), pour exécuter leurs tâches.[17]

Machine virtuelle

La machine virtuelle (virtual machine) est le résultat de la virtualisation dans laquelle on crée une version logicielle (virtuelle) d'une entité physique, elle s'applique aux serveurs, système d'exploitation, les machines virtuelles ont pour but de réduire les dépenses, augmenter le gain de productivité.[17]

Tâches

Les tâches (cloudlets) sont l'ensemble des actions que le client souhaite exécuter au sein d'un Cloud Computing, ces actions peuvent être des opérations de stockages, calcul, traitement à distance.

Fournisseur

Un fournisseur (Provider) est une société à caractère privé ou bien public offrant des services Cloud Computing tel qu'une plateforme, infrastructure, application ou de stockage, les clients payent que pour le volume de services cloud qu'ils utilisent.[20]

Service level agreement (SLA)

C'est un document, un accord, ou bien formellement un contrat établi entre le client et le fournisseur de services Cloud, contenant les services que le fournisseur met à disposition du client, la maintenance, ainsi il permet d'assurer aux clients des niveaux de sécurité en ce qui concerne le stockage et la gestion de leurs données confidentielles.[22]

Courtier

Le courtier (Broker) est une entité logicielle qui agit comme un intermédiaire entre les tâches et les datacenters, pour but de distribuer les tâches aux machines virtuelles au sein d'un datacenter. Il joue le rôle d'un négociateur entre les exigences des tâches du client et les ressources disponibles pour trouver une meilleure affectation qui satisfait QOS du client.[24]

Consommateur

Le consommateur (client) est un utilisateur bénéficiaire d'un service fourni par un fournisseur de services Cloud Computing.

Virtualisation

La virtualisation constitue le coeur du Cloud Computing. Elle consiste à ajouter une couche logicielle qui permet de faire l'abstraction entre le matériel et le système d'exploitation dans le but de faire fonctionner plusieurs systèmes d'exploitation sur la même machine physique.[22]

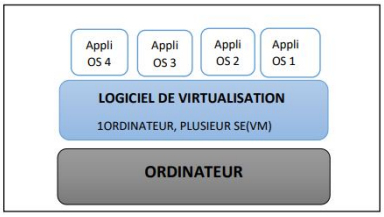

Figure I.2 – La virtualisation

4 Caractéristiques

Le Cloud Computing possède les caractéristiques suivantes :

1. **Accès en libre-service à la demande :** Le Cloud Computing offre des ressources et services aux utilisateurs à la demande. Les services sont fournis de façon automatique, sans nécessiter d'interaction humaine.[15]

2. **Accès réseau universel :** Les services de Cloud Computing sont facilement accessibles au travers du réseau, par le biais de mécanismes standard, qui permettent une utilisation depuis de multiples types de terminaux (par exemple, les ordinateurs portables, tablettes, smartphones).[15]

3. **Mutualisation de ressources (Pooling) :** Les ressources du cloud peuvent être regroupées pour servir des utilisateurs multiples, pour lesquels des ressources physiques et virtuelles sont automatiquement attribuées[15]. En général, les utilisateurs n'ont aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies.[26]

4. **Scalabilité et élasticité :** Des ressources supplémentaires peuvent être automatiquement mises à disposition des utilisateurs en cas d'accroissement de la demande, et peuvent être libérées lorsqu'elles ne sont plus nécessaires (selon le besoin). L'utilisateur a l'illusion d'avoir accès à des ressources illimitées à n'importe quel moment, bien que le fournisseur en définisse généralement un seuil (par exemple : 20 instances par zone est le maximum possible pour Amazon EC2).[26]

5. **Autonome :** Le Cloud Computing est un système autonome, géré de façon transparente pour les utilisateurs. Le matériel, le logiciel et les données au sein du Cloud peuvent être automatiquement reconfigurés en une seule image qui sera fournie à l'utilisateur.[28]

6. **Paiement à l'usage :** La consommation des ressources dans le Cloud s'adapte au plus près aux besoins de l'utilisateur. Le fournisseur est capable de mesurer de façon précise la consommation (en durée et en quantité) des différents services (CPU, stockage, bande passante...), cela lui permettra de facturer l'utilisateur selon sa réelle consommation.[3]

7. **Fiabilité et tolérance aux pannes :** Les environnements Cloud tirent parti de la redondance intégrée du grand nombre de serveurs qui les composent en permettant des niveaux élevés de disponibilité et de fiabilité pour les applications qui peuvent en bénéficier.[6]

8. **Garantie QoS :** Les environnements du Cloud peuvent garantir la qualité de service pour les utilisateurs, par exemple, la performance du matériel, comme la bande passante du processeur et la taille de la mémoire. [28]

9. **Basé-SLA :** Les Clouds sont gérés dynamiquement en fonction des contrats d'accord de niveau de service (SLA)[6] entre le fournisseur et l'utilisateur. Le SLA définit des politiques, telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, la performance, l'exploitation, ou autres attributs du service, comme la facturation, et même des sanctions en cas de violation du contrat. Le SLA permet de rassurer les utilisateurs dans leur idée de déplacer leurs activités vers le Cloud, en fournissant des garanties de QoS [26].

5 Modèles de déploiement

Les modèles de déploiement qui se réfèrent à l'emplacement des infrastructures et ressources de la solution Cloud Computing et dans quel objectif. Les principaux modèles de déploiement sont : public, privé, communautaire et hybride.

5.1 Cloud privé

L'ensemble des ressources d'un Cloud privé est exclusivement mis à disposition d'une entreprise ou organisation unique. Le Cloud privé peut être géré par l'entreprise elle même (Cloud privé interne) ou par une tierce partie (Cloud privé externe). Les ressources d'un Cloud privé se trouvent généralement dans les locaux de l'entreprise ou bien chez un fournisseur de services. Dans ce dernier, l'infrastructure est entièrement dédiée à l'entreprise et y est accessible via un réseau sécurisé (de type VPN). L'utilisation d'un Cloud privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.[26]

5.2 Cloud public

L'infrastructure d'un cloud public est accessible à un large public et appartient à un fournisseur de services. Ce dernier facture les utilisateurs selon la consommation et garantit la disponibilité des services via des contrats SLA.[26]

5.3 Cloud communautaire

L'infrastructure d'un cloud communautaire est partagée par plusieurs organisations indépendantes ayant des intérêts communs. L'infrastructure peut être gérée par les organisations membres ou par un tiers. L'infrastructure peut être située, soit au sein des dites organisations, soit chez un fournisseur de service.[26]

5.4 Cloud hybride

L'infrastructure d'un Cloud hybride est une composition de plusieurs Clouds (privé, communautaire ou public). Les différents Clouds composant l'infrastructure restent des entités uniques, mais sont reliés par une technologie standard ou propriétaire permettant ainsi la portabilité des données ou des applications déployées sur les différents Clouds.[26]

Figure I.3 – Modèles de déploiement [13]

6 Modèles de services

Les modèles de services font référence aux différents types de services qu'un fournisseur est capable de proposer, accessible à travers une plateforme de Cloud Computing.

6.1 IaaS

Dans ce modèle, le fournisseur met à la disposition des clients du matériels et ressources et prend en charge la gestion de l'infrastructure, par contre le client se limite à la gestion des systèmes d'exploitation. Nous pouvons citer Amazon Elastic compute Cloud(EC2) qui est un des fournisseurs d'IaaS et qui offre aux clients l'accès à un ordinateur sous la forme d'une image d'une machine virtuelle, sur laquelle il fait ce qu'il veut et la responsabilité de fournisseur se limite à la performance des ressources. Parmi les fournisseurs de service IaaS, nous citons : Amazon EC2, Windows Azure, Oracle, Outscale et OVH.[22]

6.2 PaaS

Dans ce modèle, le fournisseur met à la disposition des clients des plates-formes de développement, des systèmes d'exploitation. Le client a la possibilité de déployer ses propres applications standard, le client gère l'installation et la gestion des applications qu'il implémente et le fournisseur gère l'infrastructure Cloud Computing, le système d'exploitation. Parmi les fournisseurs de PaaS, nous citons : GooglAppEngine, Azure Platform.[22]

6.3 SaaS

Dans ce modèle, le fournisseur propose un environnement complet qui contient même les applications, le client gère l'application à travers un navigateur web. Le client ne se soucie de rien à part la gestion de sa propre donnée. Le fournisseur de services SaaS prend en charge la responsabilité de ce qui se passe de l'infrastructure jusqu'à l'application. Parmi les fournisseurs de service SaaS, nous citons : AWS, Azure, GCP, IBM.[22]

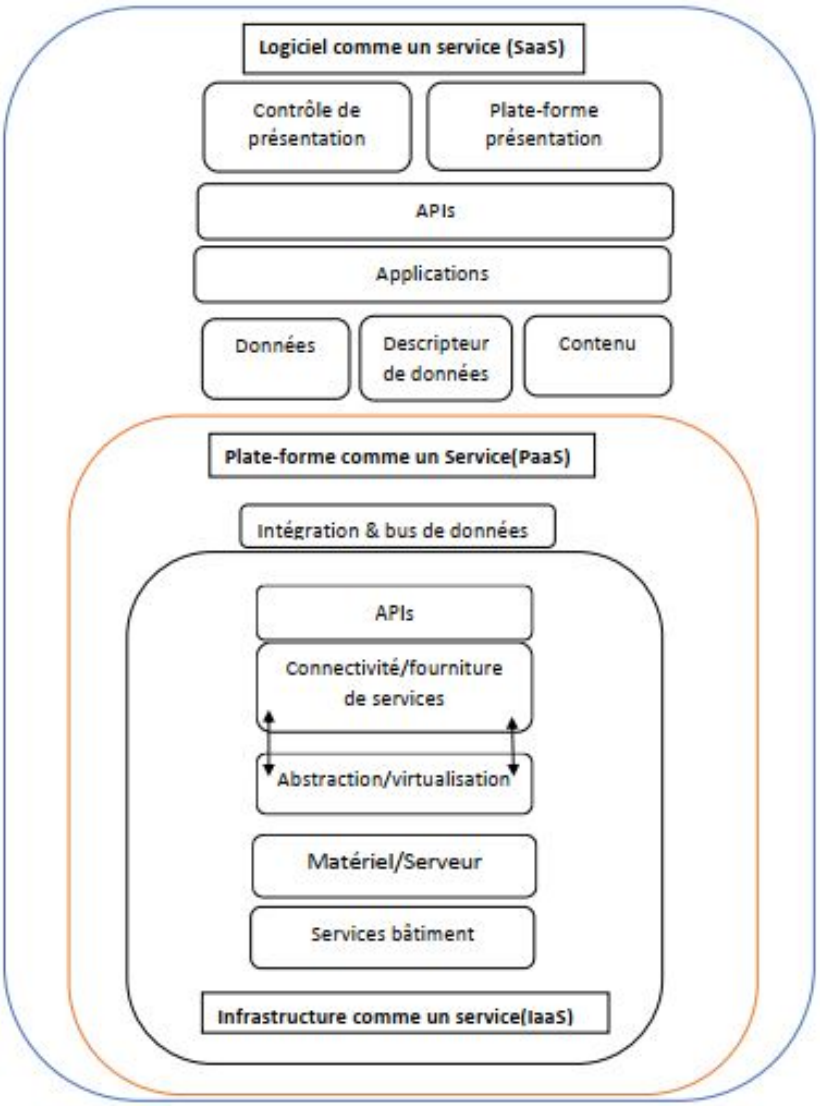

Figure I.4 – Modèle générale de l'architecture de Cloud Computing [22]

7 Apports et risques

7.1 Apports

L'arrivée du Cloud Computing est considérée comme une révolution dans le secteur informatique et économique, nous citons quelques apports de Cloud Computing :

Réduction des coûts

Le Cloud Computing donne accès à des services coûteux à moindre prix et de manière évolutive.

L'accessibilité et Mobilité

Grâce au Cloud, l'utilisateur peut utiliser ces services à distance, à n'importe quel moment et de n'importe quel appareil.

Disponibilité

Le client peut profiter d'un service quand il veut et de manière dynamique (évolutive) et transparente.

7.2 Risques

Les fournisseurs font face à de nombreux risques qui sont définis comme des événements incertains avec une probabilité de se produire et d'avoir un impact négative (menace). La démarche Cloud étant récente, les fournisseurs doivent mettre en place un système pour la gestion des risques du moment que le risque zéro n'existe pas.

CIA

La confidentialité, l'intégrité et la disponibilité des données sont les principales préoccupations des clients lors d'une migration vers une solution Cloud.

1. **La confidentialité :** Il est primordial pour les clients de prévenir le risque de perdre des données confidentielles en imposant à leurs fournisseurs des normes de sécurité.[22]
2. **Intégrité :** L'intégrité de données qui peut-être associée à la non-répudiation des données : le récepteur d'un message a la possibilité de savoir qu'une donnée a été modifiée ou bien consultée a l'aide des mécanismes de sécurité.[22]
3. **Disponibilité :** L'accès aux ressources doit être permanente et évolutive.

La continuité du métier

Le Cloud Computing étant récent, de nombreux fournisseurs sont tous récents aussi et apprennent en même temps que leurs clients. Que se passerait-il si le fournisseur disparaissait ?, la continuité du métier du client est quasiment importante.[22]

Les coûts cachés

Une entreprise doit mener une étude avant de prendre la décision de migrer ses centres de données vers une solution cloud. Une étude de cas proposée par McKinsey indiquait que les coûts augmenteraient de 144 %.[22]

La perte des données

La perte des données est le risque majeur le plus significatif. Il faut donc s'assurer de la bonne sauvegarde des données.

8 Ordonnancement des tâches

L'ordonnancement des tâches dans le Cloud Computing, permet de définir la manière d'allouer des ressources aux utilisateurs afin d'exécuter leurs tâches. La démarche Cloud Computing étant récente, L'ordonnancement des tâches et l'allocation des ressources dans le Cloud Computing sont considérés comme l'un des enjeux essentiels, plusieurs recherches ont été consacrées afin d'atteindre une gestion optimale des ressources au sein d'un Cloud Computing.

À travers cette partie nous allons présenter, quelques définitions, les principaux algorithmes d'ordonnancement des tâches dans le Cloud Computing.

8.1 Définition

L'ordonnancement des tâches dans le Cloud Computing est le processus d'affectation des tâches aux machines physiques (plus précisément les machines virtuelles). Cette attribution joue un rôle très important sur les performances du datacenter, ainsi que d'autres exigences de l'utilisateur décrites dans le SLA. [12]

8.2 Problème d'ordonnancement

Le problème d'ordonnancement consiste à la manière d'organiser dans un temps déterministe, la réalisation de tâches, en prenant en considération des contraintes temporelles (contraintes de délai, contraintes d'enchaînement, ...), ainsi que des contraintes liées à l'utilisation et la disponibilité des ressources.[7]

8.3 Rôle d'ordonnanceur

Le rôle d'un ordonnanceur est de trouver la machine virtuelle appropriée pour traiter les tâches qui lui sont soumises. Les ordonnanceurs se varient de plus simple (Allocation Round Robin) au plus compliqué (selon des contraintes).[23]

8.4 Algorithmes de base sur l'ordonnancement des tâches

Nous allons présenter dans ce qui suit, les principaux algorithmes de base d'ordonnancement des tâches :

Algorithme Min-min

L'algorithme procède comme suit : il calcule le temps d'exécution minimale pour toutes les tâches afin de choisir la valeur minimale entre ces temps minimum, qui représente le temps minimum d'exécution parmi toutes les tâches sur les ressources. Ensuite, en fonction de ce temps minimum, la tâche est ordonnancée sur la machine correspondante. Puis le temps d'exécution pour toutes les autres tâches est mis à jour sur cette machine en ajoutant le temps d'exécution de la tâche assignée a des temps d'exécution des autres tâches sur cette machine/ressource et la tâche assignée est supprimée de la liste des tâches. la même procédure est répétée jusqu'à ce que toutes les tâches soient assignées sur les ressources.[16]

Algorithme Round Robin

Le principe de l'algorithme est simple qui consiste à distribuer de façon équitable les tâches sur les machines virtuelles disponibles, autrement dit le nombre de tâches pour chaque machine virtuelle est le même. L'algorithme Round Robin[19] est implémenté par défaut dans le simulateur CloudSim.[16]

Algorithme FIFO/FCFS

L'algorithme FIFO (First In First Out) ou FCFS (First Come First Served) est l'un des algorithmes les plus simples. L'idée de base est d'ajouter chaque tâche et ressource disponible dans une file et exécuter chaque tâche et ressource par leur ordre d'arrivée.[16]

Algorithme Shortest Job First/Plus court d'abord

Le principe de l'algorithme SJF ressemble au FIFO, on choisit d'exécuter la tâche qui sera plus courte, au lieu d'exécuter dans l'ordre d'arrivée. Le problème se pose dans la détermination du temps d'exécution d'une tâche avant de l'exécuter et pour cela il faut se baser sur une estimation.[16]

9 Conclusion

Dans ce chapitre, nous avons présenté des notions fondamentales sur le Cloud Computing, ensuite nous avons abordé quelques concepts sur l'ordonnancement des tâches. Nous avons aussi mis l'accent sur quelques algorithmes de base de l'ordonnancement comme Min-Min, Round Robin, FIFO et SJF. Malgré l'efficacité de ces algorithmes, ils se concentrent principalement sur l'optimisation d'un ou deux objectif classique. Étant donné que notre mission est de proposer un algorithme d'ordonnancement des tâches multi-objectifs basées sur un algorithme d'optimisation dit métaheuristique. Dans le chapitre suivant nous allons présenter quelques notions fondamentales sur les métaheuristiques.

Chapitre II

Les Métaheuristiques

Sommaire

1 Introduction

Dans l'ingénierie, des nouveaux problèmes combinatoires sont apparue, qu'on ne peut pas résoudre avec les méthodes existantes, ce qui a motivé les scientifiques et les chercheurs à concevoir de nouvelles méthodes en s'inspirant par exemple de phénomène naturel et de comportement collectifs de quelques insectes afin de développer des algorithmes d'optimisation combinatoires, ces méthodes sont appelées :**Métaheuristiques**[31].
Dans ce chapitre, nous allons présenter quelques notions fondamentales sur les métaheuristiques notamment leurs définitions, mode de fonctionnement, caractéristiques et une classification. Ensuite nous allons présenter de manière détaillée l'algorithme Cuckoo search.

2 Définition

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 objectif.[5]

L'idée de base est d'avoir une flexibilité en échange de réduire les exigences vise à vis la qualité de la solution trouvée, leur but est de trouver des solutions bonnes en temps acceptable mais aucune garantie de la solution trouvée, il peut exister d'autres solutions optimales dans l'espace de recherche. Parmi les métaheuristique existants, nous citons : la recherche taboue[5], le recuits simulé[5], les colonies de fourmis [5] et les algorithmes révolutionnaires[5].

3 Caractéristiques

Les caractéristiques essentielles des métaheuristiques sont les suivantes :

1. Les métaheuristiques sont basées sur des paramètres qui guident l'exploration et affectent la qualité de la solution.[5]
2. Les métaheuristiques nécessitent la spécification d'un point de départ pour l'exploration, qui est choisie aléatoirement.[5]

3. Les métaheuristiques nécessitent la spécification d'une condition d'arrêt, par exemple un temps de CPU maximum, nombre d'itération.
4. Les métaheuristiques sont gourmands en matière de CPU mais simple à implémenter.
5. Les métaheuristiques parcourt un espace de recherche en se basent sur la combinaison de deux actions :

   Intensification (exploitation) :

   Cette action consiste à procéder à une recherche intensifiée autour d'une solution déjà existante dite prometteuse.

   La diversification (exploration) :

   Cette action consiste à explorer de nouvelles régions de l'espace de recherche (souvent par randomisation) qui peut être pas encore visitée.

6. Les métaheuristiques ne font généralement aucune garantie de la qualité de la solution trouvée, leur traitement est souvent stochastique et peuvent donner des différents résultats sur une même instance du problème, plus on augmente le nombre des itérations plus on a de chance d'obtenir une meilleure solution.

4 Principe de base

Les métaheuristiques sont basées généralement sur des principes communs qui sont les suivants :

1. On a un espace de recherche nommé S et une fonction fitness $f : S \to R$.
2. On définit un voisinage $V(x)$ pour chaque $X \in S$, qui est l'ensemble des points $Y \in S$ que X peut atteindre, ce voisinage est spécifie par un ensemble de transformation Ti, appelée aussi mouvement qui permet d'obtenir Y à partir de X, donc Si $Y \in V(x)$ alors il existe un i tel que $Y = Ti(x)$.[5]
3. On spécifie un opérateur dit d'exploration U qui a pour but de choisir à partir d'un $X0$ le prochain point $X1 \in V(X0)$ de la trajectoire de la recherche, l'opérateur va se baser sur l'utilisation des valeurs de la fonction fitness (objectif) dans le voisinage de $V(x0)$. Ce processus d'exploration : $X0 \to X1 \in V(X0) \to X2 \in V(X1)\ldots$, continue jusqu'à l'obtention d'un critère d'arrêt (nombre d'itération, temps CPU).[5]

5 Classification des métaheuristiques

Les métaheuristiques peuvent être classés selon plusieurs paramètres, nous citons les suivantes :

5.1 Fonction objectif

C'est-à-dire la manière d'utiliser la fonction objectif 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.[25]

5.2 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).

Métaheuristiques à base de 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[5], le recuit simulé[5].

Métaheuristiques à base de population de solutions

Dans cette famille de métaheuristique, le processus de recherche débute avec un ensemble de solutions dit populations, et durant les itérations du processus de recherche ils essayent progressivement d'améliorer leurs qualités afin d'aboutir à des solutions avec de meilleures performances. 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[5], la recherche coucou (cuckoo search).[9]

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

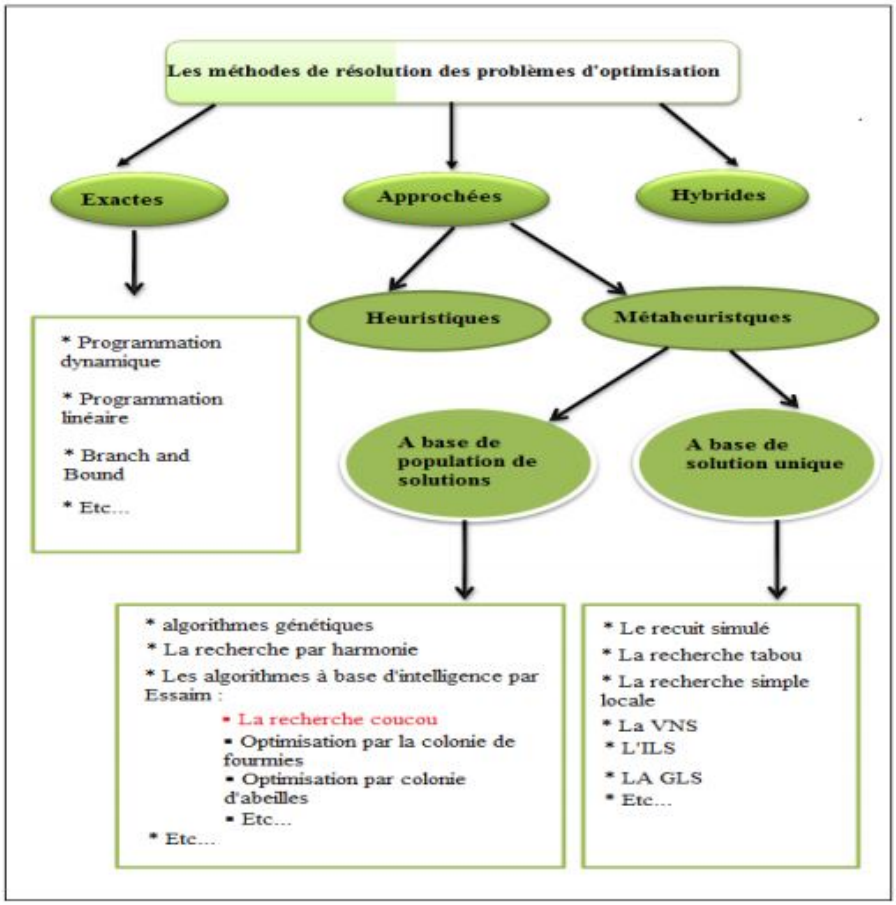

Figure II.1 – Classification des méthodes de résolution de problèmes d'optimisation [10]

Différence entre métaheuristique à base d'une solution et population de solution

La différence entre les deux familles est que dans les métaheuristiques à population, l'espace de recherche devient le produit cartésien $\overset{N}{S} = S \times S \times S \times S...$, où N est le nombre d'individus de la population et S est l'espace de recherche d'un individu.[22]

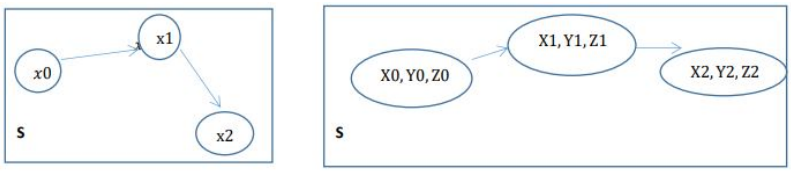

Figure II.2 – Espace de recherche dans les deux familles

5.3 Suivant la source d'inspiration

Nous pouvons distinguer les métaheuristiques qui s'inspirent des phénomènes naturels et de comportement collectifs de quelques insectes tels que la recherche cukoo, les algorithmes d'optimisation par colonies de fourmis, et d'autres métaheuristiques, comme la recherche taboue et la recherche d'harmonie.[9]

6 Algorithme cukoo search

Afin de mieux décrire la métaheuristique recherche coucou (cuckoo search), nous allons d'abord présenter le comportement de quelques espèces de la race coucou et la méthode vol de lévy.

6.1 Comportement d'élevage du cukoo

Cette métaheuristique récente est inspirée du comportement de certaines espèces d'oiseaux, qui sont les coucous qui déposent leurs œufs dans les nids d'oiseaux étrangers, ils développent et mettent en œuvre des techniques pour faire en sorte que les parents hôtes s'occupent de leurs œufs à leur place. Les coucous ont une stratégie de reproduction opportuniste néanmoins certains oiseaux découvrent les œufs intrus et s'en débarrasse ou bien construisent un notre nid ailleurs.[30]

6.2 Description de l'algorithme

La métaheuristique recherche coucou fait sa démarche selon les étapes suivantes :

- Chaque coucou pond un œuf (une solution) à la fois et la dépose dans un nid choisi au hasard sachant que le nombre de nids est fixe, on choisit une fraction parmi les nids contenant les meilleurs œufs (solutions) qui passent à la génération suivante.

- Il y a une probabilité $Pa \in [0, 1]$ que l'hôte découvre l'œuf intrus et par conséquent il abandonne le nid et construit un autre ailleurs ou bien il se débarrasse de l'œuf. Cette action est simulée en remplaçant la fraction Pa (que l'hôte découvre l'œuf) de nids choisis parmi ceux qui contiennent les moins bonnes solutions par de nouveaux nids contenant des solutions choisies au hasard par une marche dite vol de Lévy que nous allons détailler par la suite.

- L'algorithme commence par établir N nids contenant les N solutions initiales du problème au hasard puis il évalue la fitness (fonction objectif) de chaque solution, ensuite l'algorithme entre dans une boucle qui s'arrête selon des critères prédéfinis, comme le nombre maximum d'itération.

- Dans la boucle, en exécutant un vol de Lévy en partant de l'œuf choisi au hasard, on génère une nouvelle solution XI (œuf de coucou), on évalue ensuite la nouvelle solution si elle est meilleure que la précédente on la dépose dans le nid choisi au hasard.

- La dernière étape de l'algorithme consiste à remplacer les fractions Pa des N nids contenant les moins bonnes solutions en construisant des nouveaux nids par un vol de Lévy.

À la base des étapes décrites ci-dessus que l'algorithme cuckoo search fait sa démarche de recherche, qui peut être résumée comme indiqué dans le pseudo-code ci-dessous.

Algorithm 1 Cuckoo search

Initialiser une population de n nids d'accueil *xi* (I = 1, 2, ..., n).
while *(t <NbrMaxGéneration)* **do**
    Choisir une solution j au hasard
    et générer une solution par vol de Lévy
    Evaluation de la fitness fi de la solution
    if *(Fi >Fj)* **then**
        Remplacer j par la nouvelle solution
    end
    On classe les solutions par leur fitness et on trouve les meilleures solutions courante.
    la fraction *pa* des moins bons solutions sont abandonnés et une quantité équivalente de nouvelles solutions est générée par le vol de Lévy
    Les meilleurs solutions passent a l'itération suivante
end

6.3 Lévy Flight

Dans la nature, la recherche des animaux pour la nourriture se fait d'une manière aléatoire ou quasi aléatoire. En général, le chemin de recherche de nourriture d'un animal est effectivement une marche aléatoire parce que le prochain mouvement est basé sur la localisation (état actuel), et la probabilité de transition au prochain emplacement. La direction choisit dépend implicitement sur une probabilité qui peut être modélisée mathématiquement. D'ailleurs, diverses études ont montré que le comportement de nombreux animaux et insectes a démontré les caractéristiques typiques du vol de lévy.[30]

Le vol de lévy, nommé d'après le mathématicien français Paul lévy, est une marche aléatoire dans laquelle les étapes sont définies en fonction des longueurs des pas, qui ont une certaine distribution de probabilité, avec les directions des étapes étant isotrope et aléatoire. Le terme «Lévy Flight» a été inventé par Benoît Mandelbrot[2], qui a utilisé cela pour une définition précise de la distribution des tailles de *pa*. Les chercheurs Reynolds et Frye[21] ont mené une étude récente qui a montré que les mouches des fruits, explorent leur paysage en utilisant une suite de trajectoires de vol droites ponctuées par un brusque virage à 90 dégrées, conduisant au style intermittent du vol de lévy.[32]
Même la lumière peut être liée à des Lévy flights (Barthelemy et al 2008) [4]. Par la suite, un tel comportement a été appliqué à l'optimisation et la recherche optimale.[30]

La propriété essentielle du lévy est la divergence des écarts de pas dans tous les cas. Il existe aussi un autre mouvement qu'on appelle le mouvement Brownien mais ce qui le différencie du lévy, est que les écarts de pas sont petits.[32]

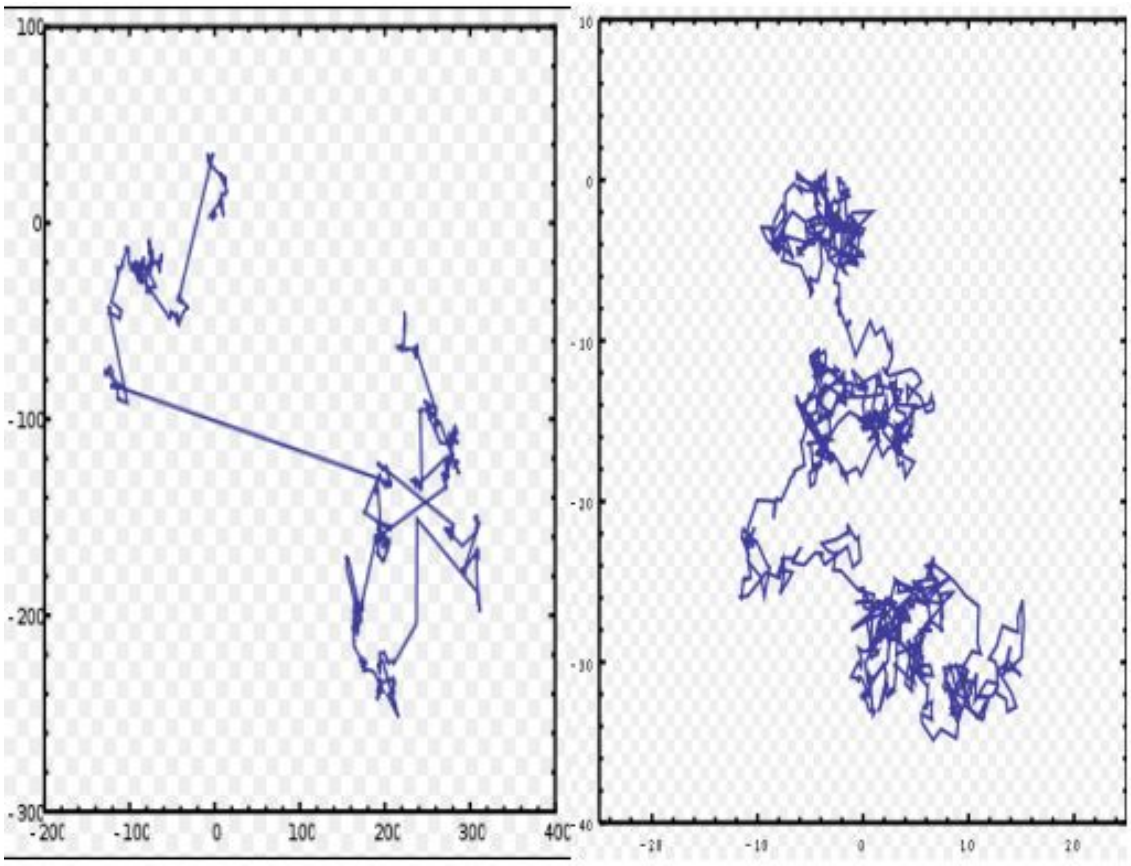

Figure II.3 – Levy flight et Mouvement brownien[32].

Comme il est montré dans la figure ci-dessous, qui représente un exemple de 1000 pas d'un Lévy flight en deux dimensions. L'origine du mouvement est à [0, 0], la direction angulaire est uniformément répartie et la taille de l'étape est distribuée selon une distribution Lévy. On constate la présence de grands sauts par rapport au mouvement brownien (a droite) qui est aussi un exemple de 1000 pas du type de mouvement brownien en deux dimensions. L'origine du mouvement est à [0, 0], la direction angulaire est uniformément répartie et la taille de l'étape est distribuée d'une distribution normale. La génération des solutions $Xi + 1$ se fait par un vol de lévy comme suit :

$$\overset{(t+1)}{Xi} = \overset{(t)}{Xi} + \alpha \oplus lévy(\lambda)$$

Où : α, indique la taille de l'étape. Le vol Lévy garantie une marche aléatoire, ses étapes aléatoires sont tirés de la distribution Lévy avec grand pas[32]. $Lévy \sim u = \overset{-\lambda}{t}$, avec $1 < \lambda < 3$.

Nous avons implémenté le vol de Lévy, en se basant sur les étapes et les formules décrites dans le document[29], qui sont les suivants :
Afin de calculer la taille de pas, nous nous appuyant sur la distribution de Lévy, cette approche calcul le facteur $\hat{\phi}$ suivant :

$$\hat{\phi} = \left(\frac{\Gamma(1+\hat{\beta}).\sin(\frac{\pi.\hat{\beta}}{2})}{\Gamma\left((\frac{1+\hat{\beta}}{2}).\hat{\beta}.2^{\frac{\hat{\beta}-1}{2}}\right)} \right)^{\frac{1}{\hat{\beta}}}$$

où Γ dénote la fonction gamma. Dans l'implémentation d'origine dans le document[1], la valeur de $\hat{\beta} = \frac{3}{2}$, est utilisé et nous faisons la même chose dans notre implémentation. Ce facteur est utilisé pour le calcul de la taille de pas ς :

$$\varsigma = \frac{u}{\mid v \mid^{\frac{1}{\hat{\beta}}}}$$

où u suit la distribution de Lévy donnée par l'équation $\hat{\phi}$, et v = 1. La taille du pas ζ est alors calculée comme :

$$\zeta = 0.01.\varsigma(X - Xbest)$$

où ς est obtenu conformément à l'équation donner précédemment, et x est modifié comme suit : $x \leftarrow x + \zeta.\psi$, où ψ suit une distribution normale N(0,1).

L'algorithme suivant présente une partie de notre implémentation décrivant la marche de lévy.

La méthode lévy flight*/
Variables
x, s, u, v, step, stepsize, best, sigma :Réel double ;
rand :random ;
$beta \leftarrow 1.5$;
calculgama(x) ;
calcullogdegama(x) ;

$sigma \leftarrow Math.pow((gamma(1+beta)*Math.sin(3.1415*beta/2.0)/(gamma((1.0+beta)/2.0)*beta*Math.pow(2.0,(beta-1.0)/2.0))),(1.0/beta))$;
$u \leftarrow (rndm.nextGaussian()) * sigma$;
$v \leftarrow (rndm.nextGaussian())$;
$step \leftarrow u/Math.pow((Math.abs(v)),(1/beta))$;
$stepsize \leftarrow 0.01 * step * (s - best)$;
$s \leftarrow s + stepsize * rand.nextGaussian()$;
if *(s>1||s<-1)* **then**
| $s \leftarrow levy()$;
end
récupérer la valeur de s ;

---

où les fonctions logGamma et Gama sont calculées comme suit :

La méthode logGamma*/
Variables
x, tmp, ser,resultat :Réel double ;
$tmp \leftarrow (x - 0.5) * Math.log(x + 4.5) - (x + 4.5)$;
$ser \leftarrow 1.0 + 76.18009173/(x + 0) - 86.50532033/(x + 1) + 24.01409822/(x + 2) - 1.231739516/(x + 3) + 0.00120858003/(x + 4) - 0.00000536382/(x + 5)$;
$resultat \leftarrow (tmp + Math.log(ser * Math.sqrt(2 * Math.PI)))$;

La méthode Gamma*/
Variables
R, x :Réel double ;
$R \leftarrow Math.exp(logGamma(x))$;

---

7 Conclusion

Dans ce chapitre, nous avons présenté quelques notions fondamentales sur les métaheuristiques, ensuite nous avons proposé une classification des métaheuristiques en se basant soit sur la fonction objectif ou bien sur le nombre des solutions ou suivant la source d'inspiration. Enfin nous nous sommes concentré sur la métaheuristique cuckoo search afin de l'utiliser dans le cadre de notre projet de fin d'études. Cette métaheuristique a été appliquée dans pas mal des problèmes au cours de ces dernières années. Son algorithme est très simple, il donne de bons résultats et peut facilement s'adapter à des problèmes d'optimisation multi-objectifs dans les Clouds. Dans le chapitre suivant, nous allons proposer une nouvelle approche d'ordonnancement des tâches basées sur la métaheuristique cuckoo search.

Chapitre III

Implémentation et simulations

Sommaire

1 Introduction

Ce chapitre, présente notre travail en se basant sur une métaheuristique appelée cuckoo search afin d'optimiser plusieurs objectifs dans l'ordonnancement des tâches à savoir le temps d'exécution, la consommation d'énergie et le coût.

Dans ce chapitre, nous allons présenter le langage java, l'environnement de développement Eclipse. Ensuite le simulateur CloudSim, son architecture, les classes fondamentales.

Par la suite, nous expliquerons notre approche et nous citerons aussi les différentes étapes et les configurations nécessaires avant le lancement de la simulation. Nous conclurons le chapitre par une analyse des résultats.

2 Langage et environnement de développement

Ce projet a été implémenté et simulé dans un environnement qui possède les caractéristiques suivantes :

- Une machine avec un processeur intel(R) core i5, une vitesse de 1.80 **GHZ** avec une capacité de mémoire de 6GB. sous le système d'exploitation windows10 de 64bits.
- Le simulateur CloudSim version 3.0.3 développée avec langage de programmation java.
- L'IDE eclipse.

2.1 Langage de programmation Java

Java est un langage de programmation orientée objet. Il a été crée par James Gosling et Patrick Naughon, deux employés de Sun micro système avec la collaboration de Bill joy(Co- fondateur de Sun Microsystems en 1982), présenté pour la première fois le 23 mai 1995 au Sun World[11].

La particularité principale de langage Java est que, les logiciels écrits avec ce dernier sont facilement portables sur plusieurs systèmes d'exploitation tels que : Unix, Microsoft Windows, Mac OS ou Linux.

2.2 Environnements de développement

Eclipse

Eclipse est un projet organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels, en s'appuyant principalement sur le langage java.[8]

2.3 CloudSim

CloudSim est un kit d'outils de développement pour la simulation de scénarios Cloud, qui offre une plateforme de simulation extensible. CloudSim a été conçu et développé à l'Université de Melbourne en Australie en langage de programmation java. Il permet la modélisation, simulation et l'expérimentation de façon transparente des infrastructures de Cloud Computing, par exemple la création des centres de données, les machines physiques, les machines virtuelles.

Architecture de CloudSim

La figure ci-dessous montre l'architecture multicouche du logicielle de CloudSim ainsi que ses composants. Le simulateur CloudSim dans ses premières versions à utiliser un moteur de simulation qui est le SimJava qui garantit pas mal de fonctionnalités, comme la création et la gestion des files d'attente, la création des entités de système Cloud (hosts, centres de données, Vms, courtier), le contrôle de l'horloge de simulation. Cette couche a été supprimée dans les versions récentes de CloudSim afin de permettre la réalisation des traitements et des opérations complexes qu'elles s'étaient impossibles avec SimJava. La couche CloudSim fournit des moyens pour la modélisation et la simulation des centres de données, machines physiques, machines virtuelles ainsi que leurs caractéristiques (stockage, bande passante, mémoire, ...). L'affectation des machines virtuelles aux machines physiques se fait dans cette couche. [24]

La couche Usercode (code utilisateur) permet de manipuler les entités de base comme le nombre de machines physiques, le nombre des tâches et leurs exigences, le nombre d'utilisateurs et leur type d'application ainsi que la politique d'ordonnancement de broker (courtier).[24]

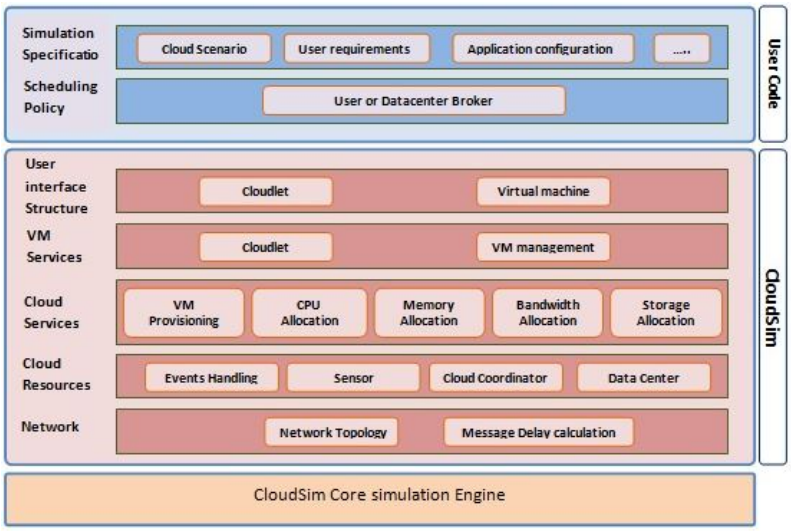

Figure III.1 – Architecture de CloudSim [24]

Classes de CloudSim

Le simulateur CloudSim comprend plusieurs classes, qui peuvent être classées en deux catégories : une catégorie qui contient des classes qui modélisent les entités du système Cloud comme les centres de données, le courtier (broker), les machines physiques (Hosts), les machines virtuelles (Vms), et une autre catégorie qui contient des classes qui modélisent la politique d'ordonnancement. Nous allons détailler quelques classes de base de Cloud.

1. **Classe Datacenter** : Cette classe modélise l'infrastructure qui représente le matériel (hôtes) fourni par le fournisseur Cloud sous forme de service. Il existe aussi la classe PowerDatacenter qui hérite de la classe datacenter, elle est dans le package Power de CloudSim, l'avantage est qu'elle permet une simulation large qui comprend le calcul de l'énergie consommée.

2. **Classe DatacenterBroker :** Cette classe modélise le broker (courtier), qui est l'intermédiaire entre l'utilisateur et le fournisseur de services Cloud. Le broker réagit comme un négociateur pour garantir une meilleure allocation qui satisfait la qualité de service de l'utilisateur. Pour cela il se base sur les CIS (Cloud Information Service) qui contient les caractéristiques des datacenters fournies par le fournisseur.

3. **Classe HOST :** Cette classe modélise les ressources physiques, elle comprend des informations essentielles concernant la mémoire, la bande passante, CPU. Elle garantit une politique d'allocation des ressources (mémoire, stockage,

bande passante) aux machines virtuelles, il y a aussi la classe PowerHost qui hérite de cette classe qui permet une simulation qui prend en charge la notion de l'énergie pour les hôtes.

4. **Classe VM :** Cette classe modélise les machines virtuelles qui sont hébergées aux niveaux des machines physiques. Elle comprend aussi les caractéristiques liées aux machines virtuelles tels que la mémoire, la bande passante, CPU...

5. **Cloudlet :** Cette classe modélise les tâches qui seront assignées aux machines virtuelles pour un traitement bien défini (traitement ou bien stockage).

Politiques d'ordonnancement

Le simulateur CloudSim fournit deux politiques d'ordonnancement :

- **La politique d'ordonnancement SpaceShared (Espace partagé) :**

  Dans cette politique d'ordonnancement, le broker (ordonnanceur) planifie une seule tâche sur une machine virtuelle à un instant donné, et après avoir terminer l'exécution de la tâche, il lance une autre tâche sur la même machine virtuelle. La politique espace partagé suit la même procédure que l'algorithme premier arrivée premier servi. Cette politique est aussi utilisée pour l'affectation des machines virtuelles aux machines physiques (host).
- **La politique d'ordonnancement TimeShared (Temps partagé) :**

  Dans cette politique d'ordonnancement, le broker planifie plusieurs tâches sur la même machine virtuelle. Il partage le temps entre toutes les tâches sur la même machine virtuelle simultanément. Cette politique est aussi utilisée pour l'affectation des machines virtuelles aux machines physiques (hosts).[7]

La différence entre les deux politiques est leurs impacts sur les performances de la simulation, comme il est montré dans la figure ci-dessous. Où, une machine physique (host) qui possède deux coeurs de processeurs et qui reçoivent une demande d'hébergement de deux machines virtuelles. Chacune de ces deux machines nécessite deux coeurs et exécute quatre tâches. Soit :**t1, t2, t3, t4** dans la machine virtuelle **VM1**, et **t5, t6, t7, t8** dans **VM2**.

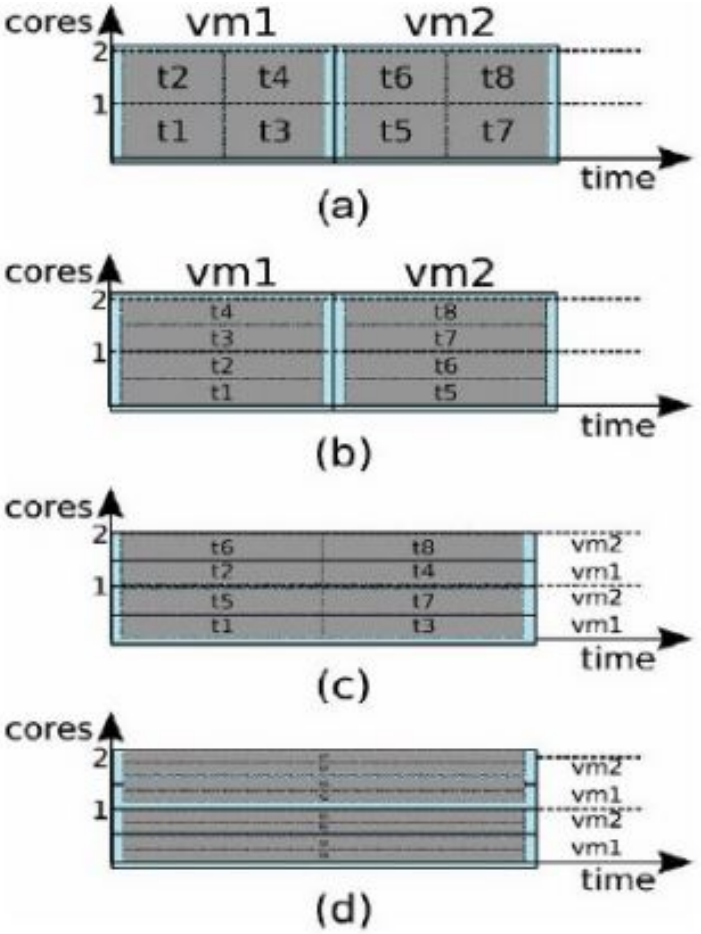

Figure III.2 – Effets des politiques d'ordonnancement sur l'exécution :**(a)** Espace partagé pour VM et tâches, **(b)**Espace partagé pour VM et Temps partagé pour les tâches, **(c)**Temps partagé pour les VMs et Espace partagé pour les tâches, **(d)**Temps partagé pour les VMs et tâches.[7]

Mode de fonctionnement

Le fonctionnement de CloudSim se fait comme suit : la création des machines virtuelles au niveau des machines physiques qui sont hébergées par le datacenter. À son tour, il envoie ses caractéristiques (nombre d'hôtes, machines virtuelles...) à un registre de Cloud information Service (CIS). À l'arrivée des tâches qui sont envoyées par un utilisateur au broker, il récupère les caractéristiques des datacenters de CIS, il agit comme un négociateur afin de choisir les machines virtuelles qui satisfont les exigences de l'utilisateur.

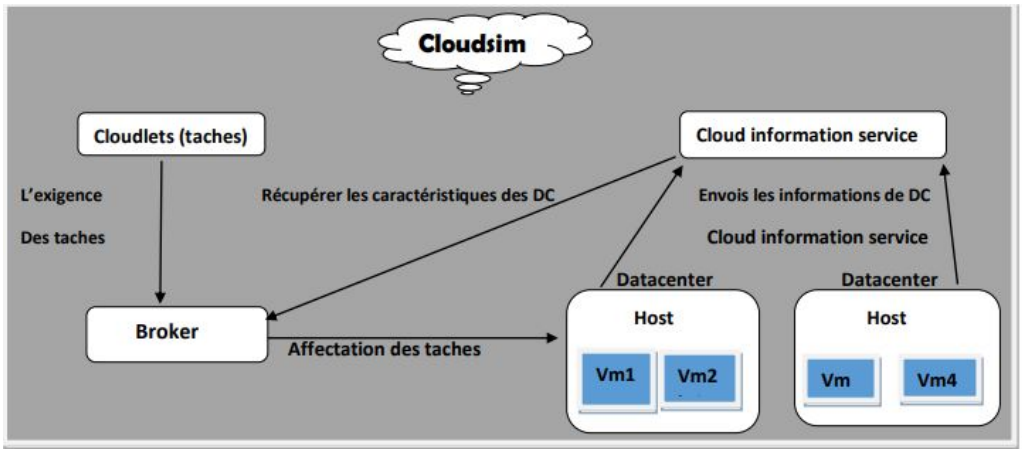

Figure III.3 – Processus de fonctionnement de cloud.

Fonctionnalités de CloudSim

Nous avons choisi le simulateur CloudSim car il offre des fonctionnalités importantes par rapport à ses concurrents (Sim Grid ,Gang Sim...), qui sont les suivantes :

1. Il offre la modélisation et la simulation des composants principaux d'un Système Cloud (centre de données, machines physiques, machines virtuelles...).
2. Il offre des politiques pour le provisionnement des ressources de la machine physique aux machines virtuelles.
3. Il met en disposition des moyens pour se servir dans les calculs d'énergie des centres de données, machines physiques et machines virtuelles.
4. Il offre une gestion dynamique pendant la simulation concernant les opérations de mise à jour (ajout, suppression) sur les ressources, ainsi que la pause et la reprise d'une simulation.

Configuration dans le CloudSim

Pour mettre en œuvre une simulation dans le CloudSim, nous suivons les étapes suivantes :

1. Initialiser le package CloudSim. Il devrait être appelé avant de créer des entités.
2. Créer les centre de données : Datacenter ← Datacentercharacteristics ← liste des hosts ← liste des éléments de traitement (CPU), définit également la politique d'allocation et de planification des machines virtuelles.
3. Création de Broker (courtier).
4. Création de Cloudlets : définit la charge de travail.
5. Création des machines virtuelles (VMs) et définir la procédure de planification des tâches.
6. Lancement de la simulation : c'est un processus automatisé, géré par le moteur de simulation d'événements.
7. Imprimer et analyser des résultats lorsque la simulation est terminée.

3 Approche proposée

3.1 Objectif du travail

L'objective de notre projet, est de proposer un mécanisme d'ordonnancement des tâches dans le Cloud Computing, nous avons proposé une approche qui consiste à gérer les ressources disponibles au sein d'un système Cloud Computing de manière optimale, et cela dans le but d'améliorer les performances du système. Cette approche est basée sur la métaheuristique «cuckoo search», qui prend en considération les trois critères suivants : temps d'exécution, consommation d'énergie, coût de traitement pour chaque tâche.

3.2 Description de l'approche

Cette section décrit l'architecture de la solution proposée. Le but de cette approche est de trouver une meilleure affectation des tâches aux machines virtuelles en matière de temps d'exécution, de coût, et de consommation d'énergie. La démarche principale de notre proposition est focalisée sur deux phases : phase d'initialisation et phase d'ordonnancement.

Phase d'initialisation

Cette phase est consacrée à la configuration de l'environnement Cloud, elle est caractérisée par la définition des différents composants du système (nombre des Datacenters, VMs, Tâches, Hôtes), voir figure ci-dessous.

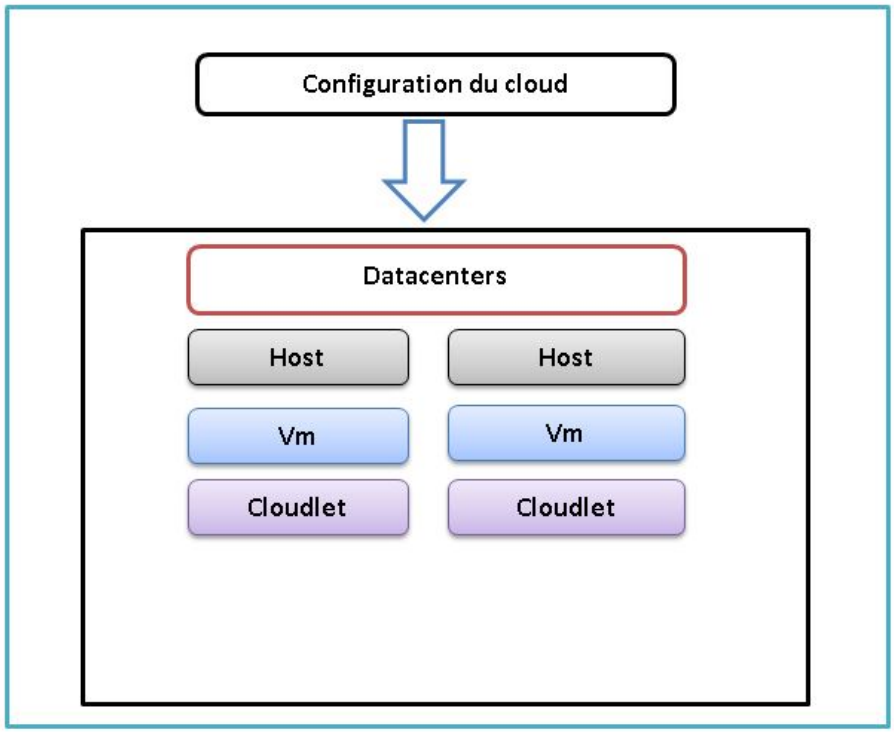

Figure III.4 – Composants du cloud.

Phase d'ordonnancement

Cette phase décrit la stratégie d'ordonnancement que nous avons proposée ainsi que les outils utilisés, notamment la fonction objectif, critères d'ordonnancement. Notre démarche d'ordonnancement se fait selon les étapes suivantes :

- **Étape d'estimation :** Cette étape consiste à calculer les estimations de nos critères afin de calculer la fonction objectif. Les estimations sont obtenues à partir des formules suivantes :
  1. **Temps d'exécution :** Représente le temps nécessaire pour une tâche pour qu'elle s'exécute au niveau de la machine virtuelle. Nous utilisons cette formule :

$$\textbf{Le Temps d'exécution} = \frac{Cloudletlength}{(VMMIPS * VMPEsNumber)}$$

Où :

- Cloudlet length signifie la taille de la tâche.
- Nombre of processeur éléments qui signifie le nombre de coeur de CPU occupée par la machine virtuelle.
- VM MIPS est la vitesse du processeur de la machine virtuelle.

2. **Coût de traitement :** Représente le coût nécessaire pour qu'une tâche s'exécute au niveau de la machine virtuelle. Nous utilisons cette formule :

Coût de traitement= (Prix *Temps d'exécution du tâche)

Où :

- Le prix d'utilisation de CPU par second. Il est fixer a 3.0.
- Temps d'exécution signifie le temps d'exécution d'une tâche.

3. **La consommation d'énergie :** Représente l'énergie consommée au niveau de la machine virtuelle pour faire un traitement d'une tâche. Nous utilisons cette formule :

$$\textbf{énergie} = \left(\frac{\Big((frompower + (topower - frompower)/2) * K \Big)}{nbc} \right)$$

Où :

- frompower signifie l'énergie consommée au début, elle est obtenue grâce à une méthode prédéfinie dans le CloudSim appelées **PowerModel»**, qui permet de récupérer la consommation d'énergie.
- topower signifie l'énergie consommée a la fin de traitement de la tâche en s'appuyant toujours sur la méthode de la classe**«PowerModel»**.
- k est le temps d'exécution de la tâche.
- nbc est le nombre de coeur de CPU occupée par la machine virtuelle.

- **Étape de Fonction objectif :** Nous allons utiliser une approche qui consiste à faire une somme pondérée des fonctions présentées précédemment où chaque fonction est associée à un poids selon son importance. La somme des poids doit être égale à 1. Cette somme représente la fonction objectif (fitness) de notre métaheuristique. Dans nos simulations, nous avons utilisé p1=0.4 pour la consommation d'énergie , p2=0.25 pour le coût et p3=0.35 pour le temps d'exécution. L'algorithme de la fonction objectif est le suivant :

Algorithm 2 Fonction Objectif

Input : F, poids1, poids2, poids3 : Réel double, T :temps d'exécution.
Output : F.

$$\textbf{Fonction objectif=} \begin{cases} Poids1 * cirière1 & + \\ Poids2 * cirière2 & + \\ Poids3 * cirière3 & + \quad T \\ & . \end{cases}$$

où p1=0.4(consommation d'énergie), p2=0.25 (coût) et p3=0.35 (temps d'exécution)

- **Étape d'ordonnancement :** Consiste à faire une pré-simulation, afin de trouver les meilleures affectations des tâches aux machines virtuelles et les envoyer au broker. L'algorithme ci-dessous présente notre approche proposée :

Algorithm 3 Algorithme d'ordonnancement.

Input : T : liste des tâches, V : liste des Vms, F : liste F vide.
Output : liste des affectation des tâches aux machines virtuelles.
for (chaque tâches dans la liste des tâches)
for (i=0 a 10 % ' $\in$ V) choisir Vmi au hasard.
Évaluation(tâche,Vmi).
ajouter Fi dans la liste F.
end for
for (i=0 a 20 % ' $\in$ V) choisir Vmj par le vol de lévy.
Évaluation(tâche,Vmj).
ajouter Fi dans la liste F.
triée la liste F et récupérer le grand Fi.
If (Fj <ù.$Fworst$)
remplacer la Vmi par la nouvelle solution Vmj
end If
end for
Triées F et choisir la meilleur solution.
Affecter la tâche a la meilleur Vm.
end for

- **Étape de simulation :** Après avoir terminé la pré-simulation, l'algorithme envoie les affectations des tâches aux machines virtuelles obtenues au broker afin de les planifier lors de lancements de la simulation.

4 Implémentation

La figure III.5 illustre la page d'accueil de notre application, l'utilisateur se retrouvera devant la fenêtre de configuration des poids de la fonction objectif. La simulation peut être lancée à partir du bouton «start simulation». Avant de lancer la simulation, l'utilisateur doit passer par les étapes suivantes :

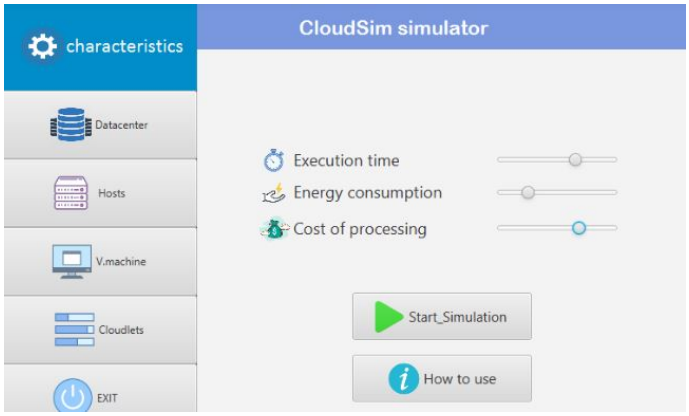

Figure III.5 – Interface principale

1. **La configuration des datacenters :** Dans cette partie et comme il est montré dans la figure III.6, l'utilisateur doit remplir les informations nécessaires pour la création d'un datacenter en glissant sur le bouton «Datacenter» puis sur le bouton «create». Ces informations sont les suivantes : le nom de datacenter, le nombre de datacenter, le prix de base de traitement, le seuil maximum de l'énergie que le datacenter peut atteindre.

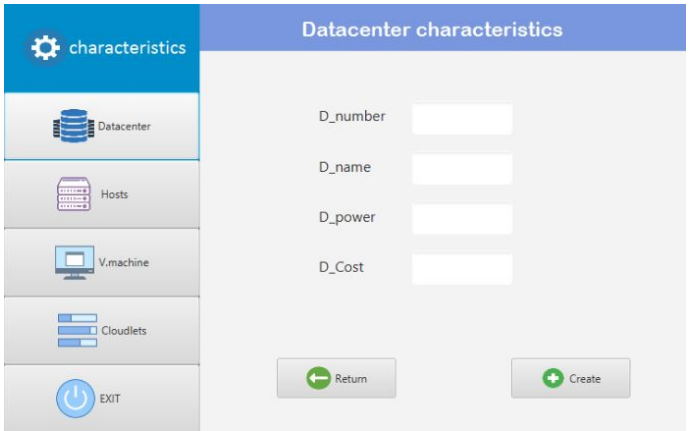

Figure III.6 – Caractéristiques de datacenter

2. **La configuration des machines physiques :** Pour la création des machines physiques, le simulateur ClouSim à besoin de la configuration des hôtes (voir figure III.7), l'utilisateur remplit les informations suivantes : le nombre d'hôtes, le nombre des coeurs de cpu, la vitesse de cpu en MIPS, la ram en MB, stockage en MB, la bande passante en Mbit/s. Puis pour la création, il clique sur «create».

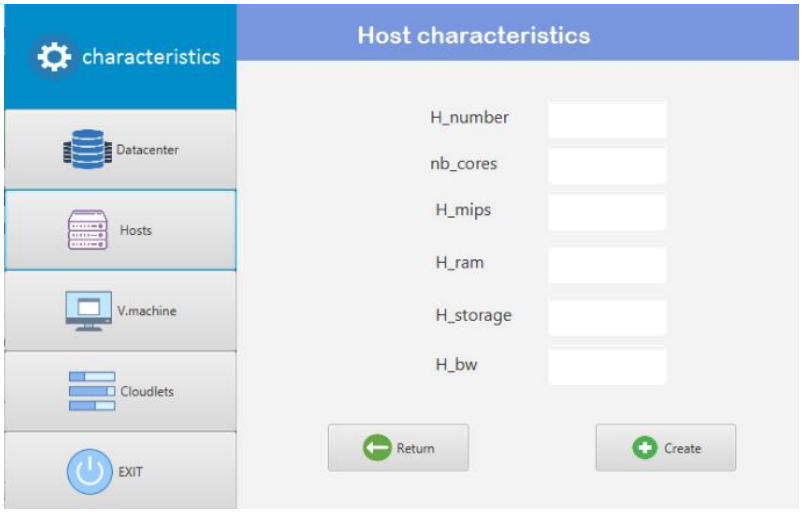

Figure III.7 – Caractéristiques de machines physiques

3. **La configuration des machines virtuelles :** En glissant sur «V.machine» (voir figure III.8) qui permet de configurer les machines virtuelles en remplissant les informations suivantes : le nombre de vm, nombre de coeurs cpu de vm, la vitesse de cpu en mips, la ram, stockage, la bande passante. Pour la création des machines virtuelles, il faut cliquer sur «create».

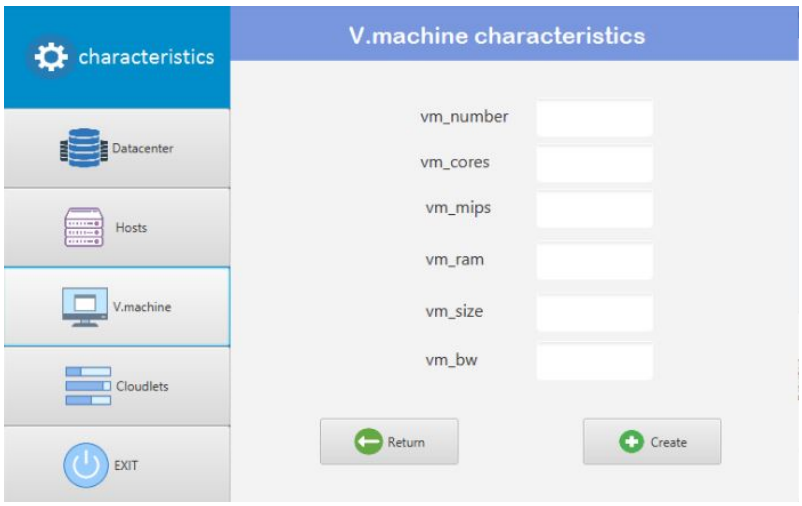

Figure III.8 – Caractéristiques de machines virtuelles

4. **La configuration des tâches :** Le bouton Cloudlets permet la configuration des tâches, pour cela comme c'est illustré dans la figure III.9, en remplissant les champs suivants : le nombre de tâches, le nombre de coeurs cpu, la taille de la tâche, fichier de la tâche en entrée, fichier de la tâche en sortie. Pour la création des tâches, il faut cliquer sur «create».

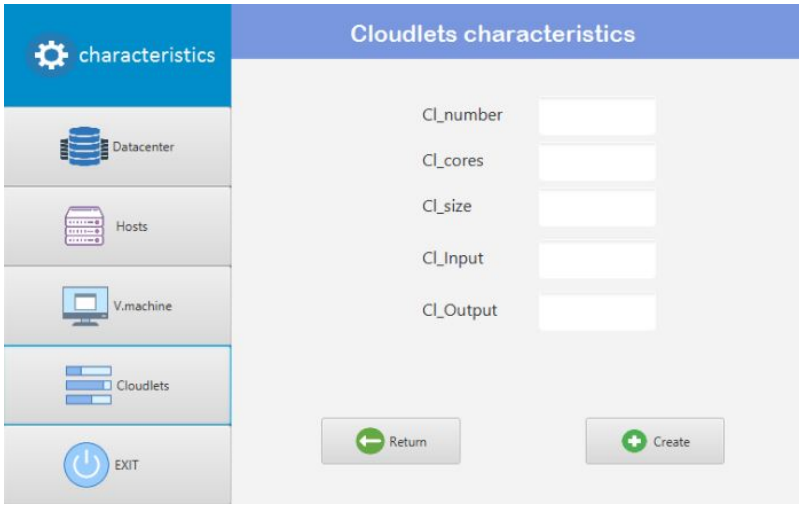

Figure III.9 – Caractéristiques des tâches

5. **Affichage :** Après la terminaison de la simulation, le résultat est affiché dans la console qui contient les informations suivantes : l'ID de la tâche, ID d'hôte, ID de datacenter, ID de la machine virtuelle, le statut de la tâche, le temps d'exécution de la tâche et le temps de début et la fin de la tâche, énergie consommée par la tâche, le coût de traitement de la tâche, la fiabilité, le coût total des tâches, énergie consommée totale des tâches (voir figure III.10).

```
========== OUTPUT ============================================================================================================
Cloudlet ID     STATUS     Data center ID     Host iD     VM ID     Time     Start Time     Finish Time     energy     cost
     1          SUCCESS          2              0          0        13          0              13           62.0       40.0
     0          SUCCESS          2              0          4        13          0              13           62.0       40.0
     3          SUCCESS          2              1          1        20          0              20           91.0       60.0
     2          SUCCESS          2              1          5        20          0              20           91.0       60.0
     4          SUCCESS          2              0          0        13         13              27           62.0       40.0
     5          SUCCESS          2              0          4        13         13              27           62.0       40.0
     6          SUCCESS          2              0          2        40          0              40          178.0      120.0
     7          SUCCESS          2              1          1        20         20              40           91.0       60.0
     8          SUCCESS          2              1          5        20         20              40           91.0       60.0
     9          SUCCESS          2              0          0        13         27              40           62.0       40.0
 fiabilty  100.0%
total cost is 560.3235
total energie consumed is 852.0
CloudSimpremieressaie finished!
```

Figure III.10 – Résultat de la simulation

5 Simulations

5.1 Résultats des simulations

Dans cette partie, nous avons effectué plusieurs simulations à notre algorithme et l'algorithme Round Robin (déjà implémenté dans le Cloud Sim), afin d'étudier l'impact du nombre des tâches sur le temps d'exécution, le coût et la consommation d'énergie, et afin de voir le gain apporté par notre algorithme.
Nous avons utilisé la configuration Cloud suivante :

1. quatre datacenters qui possède les propriétés suivantes :

Dci	Nombre d'hôtes	Nombre de Vms
Dc1	4	16
Dc2	4	16
Dc3	4	16
Dc4	4	16
Totale	16 Hôtes	64 Vms

Table III.1 – Caractéristiques de datacenter

2. 16 machines physiques qui possède les propriétés suivantes :

Hôtes	Coeurs cpu	Mips	Ram	Bande passante	Stockage	politique
16 hôtes	4 coeurs	3000	4098 MB	1000 Mbit/s	100000 MB	SpaceShared

Table III.2 – Caractéristiques des machines physiques

3. 64 machines virtuelles qui sont les suivantes :

Type Vm	Coeurs cpu	Mips	Ram	Bande passante	Stockage
Type 1	1	3000	512 MB	1000 Mbit/s	10000 MB
Type 2	1	2000	512 MB	1000 Mbit/s	10000 MB
Type 3	1	1000	512 MB	1000 Mbit/s	10000 MB
Type 4	1	500	512 MB	1000 Mbit/s	10000 MB

Table III.3 – Caractéristiques des machines virtuelles

4. un certain nombre des tâches qui possède les mêmes caractéristiques suivantes :

Tâches	Taille	coeur cpu	Ram	Bande passante	Stockage	politique
N tâches	40000 MB	1	utilisation totale	utilisation totale	utilisation totale	SpaceShared

Table III.4 – Caractéristiques des tâches

Comme montrer dans la figure , ci dessous, la topologie globale de la simulation.

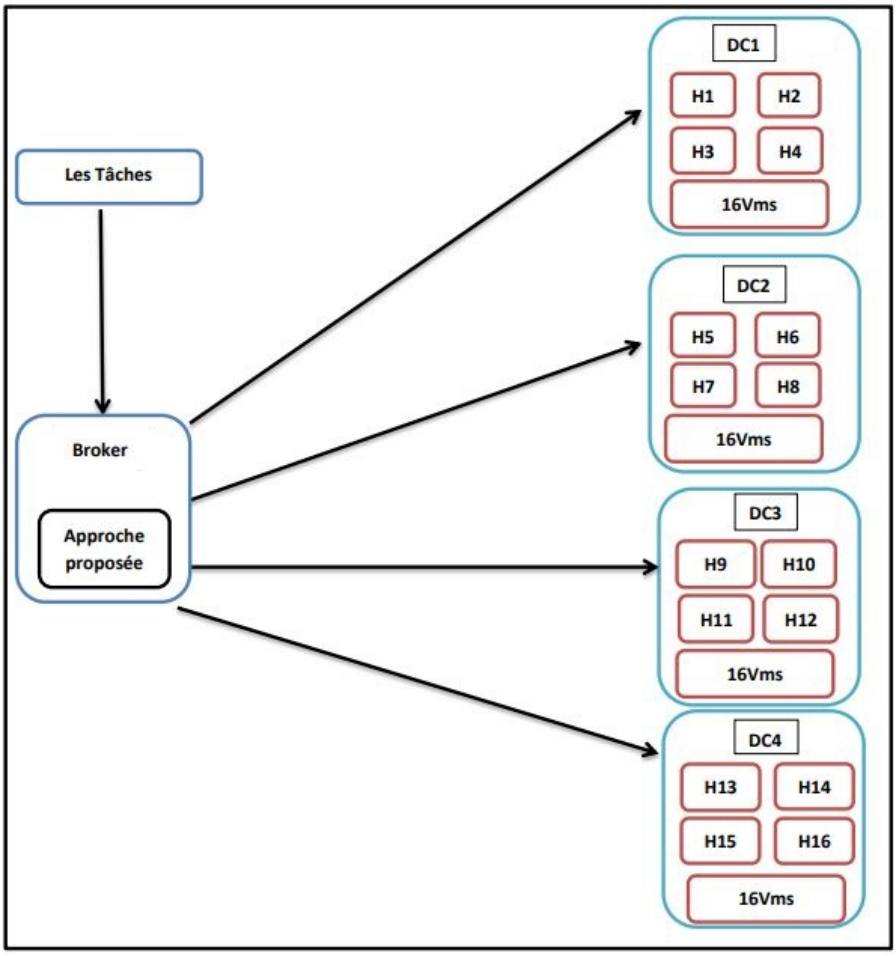

Figure III.11 – Topologie de la simulation

Expérimentation

Cette partie, est consacrée aux résultats obtenu après plusieurs simulations effectuée au niveau du simulateur CloudSim, nous avons obtenu les résultats suivants :

Temps d'exécution

Selon les résultats obtenus, on remarque les deux courbes se croissent de manière progressive, et au bout de 150 tâches, le temps d'exécution des tâches augmente de façon exponentielle dans Round Robin, tandis que la courbe de cuckoo search se croisse de manière progressive jusqu'à il finit les 1000 tâches à 400 secondes par contre l'algorithme Round Robin termine à 1300 secondes.

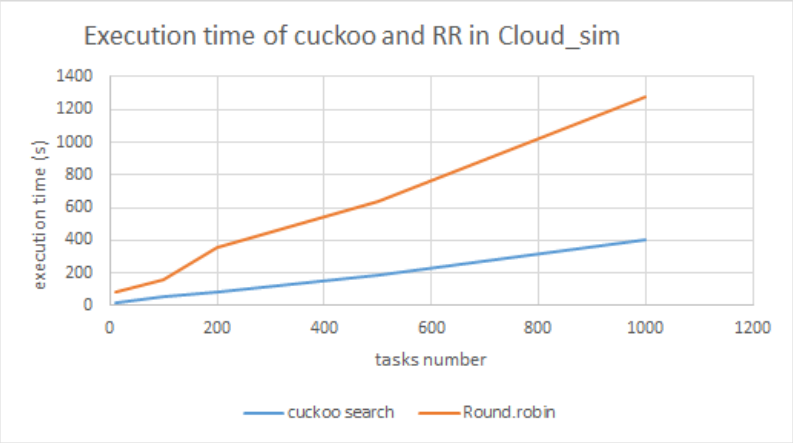

Figure III.12 – Résultat de temps d'exécution sur les deux algorithmes

Consommation d'énergie

D'après le graphe, sur le plan énergie, on constate que les courbes sont croissantes mais cuckoo search reste plus efficace du moment que la consommation d'énergie totale des tâches est á 118 Kw par contre Round Robin qui est á 170 Kw.

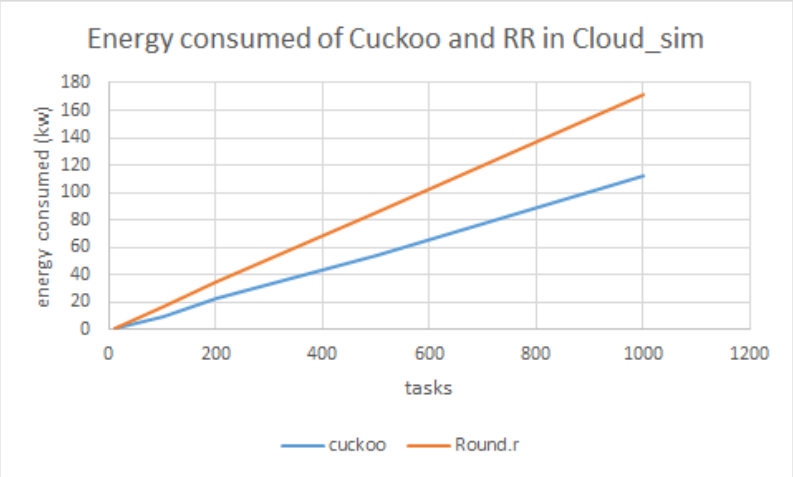

Figure III.13 – Résultat de consommation d'énergie sur les deux algorithmes

Coût de traitement

D'après la figure ci-dessous, on constate que les deux courbes, de 0 à 200 tâches sont en croissance progressive, au bout de 200 tâches, le coût des tâches dans Round Robin augmente de manière exponentielle, il arrive au final à 120000 dollars, ce qui est énorme par rapport à notre approche qui donne un coût total de 80000 dollars.

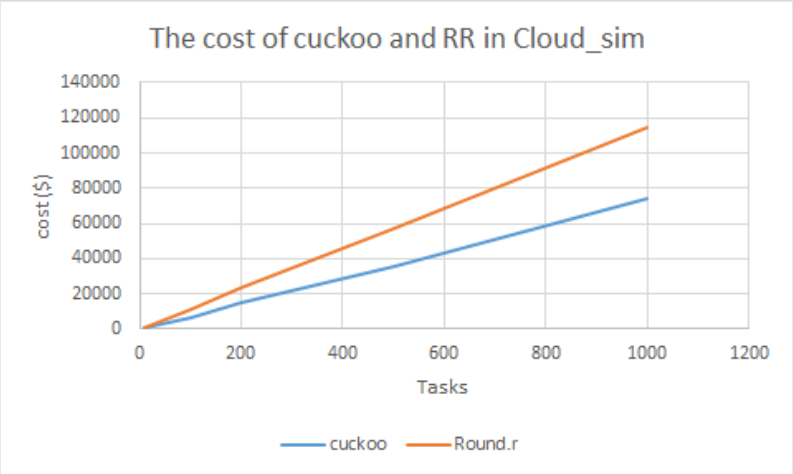

Figure III.14 – Résultat de coût d'énergie sur les deux algorithmes

Nous pouvons conclure que l'algorithme a réussi à améliorer considérablement les trois objectifs à savoir le temps d'exécution, le coût et la consommation d'énergie, et par conséquent offrir une meilleur qualité de service aux utilisateurs (temps et le coût) et un gain pour le fournisseur (consommation d'énergie).

6 Conclusion

Dans ce chapitre, nous avons justifié le choix du simulateur CloudSim qui nous a permis d'intégrer notre algorithme en utilisant une métaheuristique appelée cuckoo search afin d'optimiser plusieurs objectifs dans l'ordonnancement des tâches à savoir le temps d'exécution, la consommation d'énergie et le coût.
Pour implémenter notre approche, nous avons effectué des modifications au simulateur pour l'adapter à notre méthode. Nous avons implémenté les deux phases de notre algorithme à savoir la phase d'initialisation (fenêtres de configuration) et la phase d'ordonnancement (estimation, fonction objectif et ordonnancement). Notre application permet d'afficher les statistiques sur l'ensemble tâches après chaque simulation.
Nous avons remarqué d'après les résultats obtenus par le simulateur CloudSim que notre algorithme a réussi à améliorer considérablement les trois objectifs par rapport à l'algorithme Round Robin. En résumé, nous pouvons dire que notre approche a permis de garantir une meilleure gestion des ressources, qui apporte de bénéfice envers le client en matière de coût et de temps ainsi qu'un gain en matière de consommation d'énergie pour le fournisseur.

Conclusion générale

Le Cloud Computing présente une technologie prometteuse qui facilite l'exécution des applications dans divers domaines. Ces applications contiennent généralement de nombreuses tâches. Ces tâches requièrent pour leurs exécutions sur les machines du Cloud un ordonnancement.

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches. Pour cela, il est très souvent essentiel d'attribuer en même temps les ressources nécessaires à l'exécution de ces tâches.

Dans ce travail, nous avons proposé une approche d'ordonnancement des tâches basée sur la métaheuristique «Cuckoo Search» dans le but d'optimiser plusieurs objectifs à savoir : le temps d'exécution, coût de traitement, consommation d'énergie.

Nous avons remarqué d'après les résultats obtenus après plusieurs simulations par le simulateur CloudSim, que l'algorithme a réussi à améliorer considérablement les trois objectifs par rapport à l'algorithme Round Robin.

Dans nos futurs travaux, nous envisageons plusieurs perspectives, nous comptons perfectionner notre algorithme en ajoutant d'autres objectifs comme l'équilibrage de charge, la consolidation et la fiabilité.

ملخص

تمتلك الحوسبة السحابية تقنية واعدة تجعل من السهل تشغيل التطبيقات في مختلف المجالات. إنه يوفر خدمات مرنة وقابلة للتطوير بناءً على طلب المستخدمين مع جودة خدمة أفضل. أصبحت مهام الجدولة في السحابة تحديا مع تزايد شعبية الحوسبة السحابية. خوارزميات الجدولة الأولى لها هدف واحد ، ألا وهو سرعة المعالجة. في هذا المشروع ، نقترح استراتيجية جدولة مبنية على métaheuristique تسمى الوقواق البحثْ لتحسين عدة معايير مثل وقت تنفيذ المهمة ، واستهلاك الطاقة وتكلفة استخدام الموارد. من النتائج التي حصل عليها محاكي Cloudsim ، تمكنت الخوارزمية من تحسين الأهداف الثلاثة بشكل كبير.

الكلمات المفتاحية : الحوسبة السحابية ، جدولة الوظائف ، الدوال العليا محاكي ثلاثدصم .

Résumé

Le Cloud Computing présente une technologie prometteuse qui facilite l'exécution des applications dans divers domaines. Il fournit des services flexibles et évolutifs à la demande des utilisateurs avec une meilleure qualité de service. L'ordonnancement des tâches dans les Clouds est devenu un véritable défi avec l'augmentation de la popularité du Cloud Computing. Les premiers algorithmes d'ordonnancement visent un seul objectif à savoir la rapidité de traitement. Dans ce projet, nous proposons une stratégie d'ordonnancement basée sur une métaheuristique appelée « cuckoo Search » afin d'améliorer plusieurs critères comme le temps d'exécution des tâches, la consommation d'énergie et le coût d'utilisation des ressources. D'après les résultats obtenus par le simulateur CloudSim, l'algorithme a réussi à améliorer considérablement les trois objectifs.

Mots clés : Cloud Computing, Ordonnancements des tâches, Métaheuristique, CloudSim.

Abstract

Cloud Computing has a promising technology that makes it easy to run applications in various fields. It provides flexible and scalable services at the request of users with a better quality of service. Scheduling tasks in the cloud has become a challenge with the increasing popularity of Cloud Computing. The first scheduling algorithms have one goal, namely the speed of processing. In this project, we propose a metaheuristic scheduling strategy called "cuckoo Search" to improve several criteria such as task execution time, energy consumption and the cost of using resources. From the results obtained by the CloudSim simulator, the algorithm has been able to significantly improve the three objectives.
Keywords : Cloud Computing, Task scheduling in Cloud, Metaheuristic, CloudSim.

Bibliographie

[1] **A. IGLESIAS, A.GÁLVEZ, P.SUÁREZ, M.SHINYA, N.YOSHIDA , C. OTERO , C.MANCHADOAND, V. G.JAUREGUI**. *"Cuckoo Search Algorithm with Lévy Flights for Global-Support Parametric Surface Approximation in Reverse Engineering"*. "March 2018".

[2] **AKYILDIZ, IAN F**. *" Next generation spectrum access/ cognitive radio wirelessnetworks : a survey", computer networks."*. "2006".

[3] **"ARMBRUST, M.FOX, A.GRIFFITH, R.JOSEPH, D. KATZ, R. H., KONWINSKI.** *"Above the clouds : A berkeley view of cloud computing (Vol. 17). Technical Report UCB/EECS-2009-28, EECS Department"*. University of California, Berkeley,2009".

[4] **BARTHELEMY, PIERRE, JACOPO BERTOLOTII, DIEDERIK S. WIERSMA.** *"A Lévy flight for light.""*. "2008".

[5] **BASTIEN CHOPARD, MARCO TOMASSINI.** *"Une introduction aux métaheuristique."*. "Novembre 2017, p(149,169), p(83,103), p(107,114), p(65,73), p(40,50)".

[6] **BUYYA R., YEO C. S., AND VENUGOPAL, S.** *"Market-oriented cloud and atmospheric computing : Hype, reality, and vision. In 10th IEEE Proc International Conference on High Performance Computing and Communications"*. "2008,(pp. 5-13)".

[7] **DJEBBAR ESMA INSAF.** *"optimisation de l'ordonnancement et d'allocation des ressources dans le cloud computing."*. "5-12-2016,p(88-90)".

[8] **ECLIPSE FONDATION.** ***"Eclipse (projet)."***, *note=https://fr.wikipedia.org/wiki/Eclipse_(projet)*, *year="(consulté le 9 mai 2019)"*.

[9] **FERHAT HALIMA.** *"Optimisation de la QoS dans un réseau de radio cognitive en utilisant l'algorithme de la recherche par harmonie"*. "PFE Master RSD,Université de Tlemcen. juin 2015".

[10] **GHERBOUDJ, AMIRA.** *"Méthodes de résolution de problèmes difficiles académiques."*. "Université de Biskra, 2013".

[11] **JAVA LANGAGE.** *"langage de programmation java"*, "(2008) , (cf. p. 48, 49)". http://fr.Wikipédia.org/wiki/Java(langage).

[12] **L. BACCOUCHE.** *"Un Mécanisme d'Ordonnancement Distribué de Tâches Temps Réel. Thèse de doctorat, l'institut national polytechnique de Grenoble"*. "2004".

[13] **LEBIGDATA**. *"Définition de Cloud Computing,(consulté le 29 avril 2019)."*. https ://www.lebigdata.fr.

[14] **MALICK**. *"Cloud Computing."*. "7 novembre 2018, (consulté le 2 mai 2019)". https://www.developpez.com.

[15] **MELL, P, AND GRANCE**. *"The NIST definition of cloud computing"*. 2011.

[16] **MESLEM YAMINA,DEBBAS SOUMIA ISMAHÈN**. *"Ordonnancement et réplication de données dans le Cloud Computing."*. "2017".

[17] **MICROSOFT AZURE**. *"Qu'est-ce qu'une machine virtuelle ?"*. "2016, (consulté le 17 mai 2019)". https://azure.microsoft.com.

[18] **PETER MELL, TIMOTHY GRANCE**. *"The NIST Definition of Cloud Computing"*. "September 2011". https://www.nist.gov/.

[19] **P.PRADHAN, P. BEHERA B, RAY**. *"Modified Round Robin Algorithm for Resource Allocation in Cloud Computing"*. "2010".

[20] **REDHAT**. *"Qu'est-ce qu'un fournisseur de cloud ?"*. "2016, (consulté le 28 mai 2019)". https://www.redhat.com.

[21] **REYNOLDS, ANDY M., AND MARK A. FRYE**. *"Free flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search."*. "2007".

[22] **R.HENNION, H.TOURNIER, E.BOURGEOIS**. *"Cloud Computing"*. "2013".

[23] **RICHARD WOLSKI ,FRANCINE BERMAN**. *"Adaptive Computing on the Grid Using AppLeS, IEEE Transactions on Parallel and Distributed System."*. "2002".

[24] **RODRIGO N. CALHEIROS, RAJIV RANJAN, ANTON BELOGLAZOV, CESAR A. F. DE ROSE AND RAJKUMAR BUYYA**. *"CloudSim a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms"*. "24 août 2010".

[25] **S. DIGABEL.** *"Introduction aux Métaheuristiques , support de cours, Ecole Polytechnique de Montréal."*. "France".

[26] **SONIA YASSA**. *"Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing"*. "23 Jun 2015".

[27] **UTEC MARNE LA VALLÉE, ARTHUR VARY-SINAMALE, IDIR TIR ET MATTHIAS ALI BELKACEM**. *"DATA CENTERS et CLOUD COMPUTING."*. "2016".

[28] **WANG, L.TAO, J.KUNZE, M.CASTELLANOS, C. KRAMER, AND W.KARL.** *"Scientific cloud computing : Early definition and experience. In High Performance Computing and Communications, HPCC'08. 10th IEEE International Conférence"*. "2008,(pp. 825-830)".

[29] **YANG, X.-S.** *"Nature-Inspired Metaheuristic Algorithms, 2nd. ed. ; Luniver Press : Frome, UK."*. "2010".

[30] **YANG, XIN-SHE, AND SUASH DEB.** *"Engineering optimisation by cuckoo search .International Journal of Mathematical Modelling and Numerical Optimisation"*. "2010".

[31] **YANG, XIN-SHE, AND SUASH DEB.** *"Engineering optimisation by cuckoo search."International Journal of Mathematical Modelling and Numerical Optimisation."*. "2010".

[32] **ZERGA HIDEYAT.** *"Optimisation de la QoS dans un réseau de radio cognitive en utilisant l'algorithme de la recherche cuckoo search."*, "Université de Tlemcen, juin 2016".