RÉPUBLIQUE POPULAIRE DÉMOCRATIQUE D'ALGÉRIE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOHAMED BOUDIAF - M'SILA

FACULTIE: MATHEMATIQUES ET INFORMATQUE **DOMAINE:**MATHEMATIQUES ETINFORMATIQUE

DEPARTEMENT: INFORMATIQUE **SPECIALITE:** INFORMATIQUE

N°:…….. **OPTION:** IDO

Mémoire soumis en remplissant partiellement les conditions requises pour le diplôme de MASTER

Par: Zoulikha TITRAOUI

Sujet

L'optimisation multi objectifs pour l'ordonnancement de tâches dans Le cloudcomputing

Défendu publiquement devant le jury composé de:

Mr. Nacereddine MOUHOUB	Université de M'sila	President
Mr. Abdelbasset BARKAT	Université de M'sila	Encadreur
Mr. Mohamed CHATTRA	Université de M'sila	Examinateur

Année universitaire : 2019/2020

Table des matières

Introduction générale

Introduction générale

Récemment, l'informatique dans les clouds est proposée pour fournir une infrastructure informatique à la demande pour un certain nombre d'applications. Les applications fournies en mode cloud computing ne se trouvent pas forcément sur un serveur hébergé par l'utilisateur mais dans un « nuage » formé de l'interconnexion de plusieurs serveurs localisés à des endroits distants.

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 plusieurs avantages que ce nouveau paradigme offre, y compris la réduction des coûts, l'évolutivité rapide, la facilité de développement, le stockage illimité, et l'accessibilité omniprésente. 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.

En effet, la flexibilité et la dynamique offerte par le cloud et certains défis majeurs tels que la réduction du temps d'exécution, augmentent la complexité de l'ordonnancement et rendent difficile à trouver des solutions efficaces à ces problèmes dans un délai raisonnable. Donc, l'application de méthodes d'optimisation exactes s'avère illusoire en raison du temps de calcul.

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.

En tant que solution, les métaheuristiques multiobjectifs constituent une bonne alternative. Cependant, ces méta-heuristiques ainsi que la modélisation du problème doivent être repensées pour prendre en compte à la fois des spécifications et des contraintes du problème de l'ordonnancement sur le cloud.

Dans le présent travail nous proposons un modèle d'optimisation multiobjectifs adressé le niveau d'ordonnancement de tâche dans le cloud computing. Ce travail s'organise en trois chapitres.

Le premier chapitre dresse un état de l'art non exhaustif des domaines de l'optimisation multiobjectifs et les méthodes de résolution utilisées. Nous présentons l'optimisation multiobjectifs tout en introduisant des concepts fondamentaux du problème tels que la multiplicité des solutions et la difficulté d'un problème multiobjectif. Ainsi les principales approches de résolution pour ces problèmes. Et en a terminer le chapitre par une classe des méthodes de l'optimisation multiobjectifs qui est la programmation génétique.

Le chapitre 2 présente la littérature qui couvre les définitions de base concernant le cloud computing. L'objectif est de fournir en détail au lecteur les deux domaines dans lesquels s'intègrent les travaux de cette thèse. Il se compose de deux sections : dans la première seront présenté la définition du cloud computing et ses différents concepts, dans la seconde seront présente un état de l'art sur l'ordonnancement dans le cloud computing on définissant notre problème.

Le chapitre 3 est dédié à la conception de notre approche proposée. Dans un premier temps, nous introduisant le simulateur utilisé cloudsim. En deuxième lieu nous intégrons notre algorithme génétique, et on a terminé par réalisation, exécution et analyse des résultats.

Finalement, nous clôturons ce mémoire par une conclusion générale

Chapitre 1

L’optimisation multiobjectifs

1. Introduction

Dans le monde réel, la plupart des problèmes nécessitent l'optimisation simultanée de plusieurs objectifs souvent dépendants les uns des autres. Tandis que la solution optimale est généralement clairement définie dans l'optimisation monoobjectifs, cela n'est pas le cas pour les problèmes multiobjectifs, il existe un ensemble de solutions qui sont des « compromis ». Ces solutions sont optimales dans le sens qu'aucune autre solution dans l'espace de recherche n'est supérieure à elles, lorsque tous les objectifs sont considérés simultanément.

Dans ce chapitre, un ensemble des définitions liées à l'optimisation combinatoire est présenté au préalable, ensuite les concepts de base l'optimisation multiobjectif sont définis formellement, ceci est suivi par une discussion concernant les méthodes de résolution des problèmes d'optimisation multiobjectifs.

2. Définitions

2.1 Problèmes d'optimisation

Un problème d'optimisation, noté $P(X, f)$, est caractérisé par un ensemble réalisable ou admissible X non-vide et une fonction objectif f qui associe un scalaire dans R à chaque élément x de l'ensemble X. Les éléments de X sont dits solutions réalisables.Résoudre le problème $P(X, f)$revient à trouver parmi les solutions réalisables, une qui minimise ou maximise$f$, c'est-à-dire dans le cas d'un problème de minimisation, trouver une solution $x^* \in X$ telle que $f(x) \geq f(x*)$ pour tout élément x dans X. Une telle solution est dite optimale et sera notée $x(X, f)$.

On peut dire aussi qu'un problème d'optimisation se définit comme la recherche du minimum ou du maximum (de l'optimum donc) d'une fonction donnée. On peut aussi trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont des contraintes à évoluer dans une certaine partie de l'espace de recherche.

2.1.1 La complexité d'un problème d'optimisation

La complexité d'un problème est la complexité du meilleur algorithme qui permet de le résoudre. Si cet algorithme est polynomial, le problème est dit facile, autrement le problème est difficile. Nous présentons ici la complexité en temps. Il existe aussi une mesure de la performance des algorithmes en termes d'espace mémoire dite complexité en espace.

- **Un problème de la classe P :** un problème est dit polynomial s'il existe un algorithme de complexité polynomiale permettant de répondre à la question posée dans ce problème, quelque soit la donnée de celui-ci. La classe P est l'ensemble de tous les problèmes de reconnaissances polynomiaux.
- **Un problème de la classe NP :** un problème de reconnaissance est dans la classe NP si, pour toute instance de ce problème, on peut vérifier, en un temps polynomial par rapport à la taille de l'instance, qu'une solution proposée ou devinée permet d'affirmer que la présence est « oui » pour cette instance. [15]
- **Un problème de la classe NP-hard :** on dit qu'un problème est dans la classe NP-hard si chaque autre problème dans NP est réductible d'une manière polynomiale dans ce dernier.

- **Un problème de la classe NP-complet :** s'il appartient à NP et chaque autre problème dans NP est réductible d'une manière polynomiale dans ce dernier. C'est un problème de décision et NP-hard qui cherche à trouver l'optimum.

2.1.2 La classification des problèmes d'optimisation

On peut classer les différents problèmes d'optimisation que l'on rencontre dans la vie courante en fonction de leurs caractéristiques :

1. Nombre de variables de décision :

– Une ⇒monovariable.

– Plusieurs ⇒multivariable.

2. Type de la variable de décision :

– Nombre réel continu ⇒ continu.

– Nombre entier ⇒ entier ou discret.

– Permutation sur un ensemble fini de nombres ⇒ combinatoire.

3. Type de la fonction objectif :

– Fonction linéaire des variables de décision ⇒ linéaire.

– Fonction quadratique des variables de décision ⇒ quadratique.

– Fonction non linéaire des variables de décision ⇒ non linéaire.

4. Formulation du problème :

– Avec des contraintes ⇒ contraint.

– Sans contraintes ⇒ non contraint.

2.2 L'optimisation combinatoire

Les problèmes d'optimisation combinatoire sont des problèmes d'optimisation dont les ensembles réalisables sont finis mais combinatoires. Aussi, le nombre de solutions réalisables des problèmes combinatoires augmente exponentiellement en fonction de la taille

du problème, et c'est ce qui exclut des méthodes de résolution basées sur l'énumération de toutes les solutions réalisables.

2.3 L'optimisation multiobjectifs

L'optimisation multiobjectifs permet de chercher les valeurs des variables d'un problème qui maximisent ou minimisent un ou plusieurs fonctions objectif. Il peut s'agir par exemple de minimiser un coût de production, de rationaliser l'utilisation de ressources, d'améliorer les performances énergétiques d'un procédé industriel, etc. Elle procède donc par la définition au préalable des critères de qualité de la solution du problème, puis l'algorithme d'optimisation va résoudre le problème en cherchant les meilleures solutions en fonction de ces critères. Ainsi, la formulation du problème d'optimisation comporte les étapes suivantes :

- Exprimer les critères (ou fonctions) objectif d'optimalité

- Choisir les paramètres (ou variables) d'optimisation

- Définir un espace admissible pour les variables d'optimisation

- Définir les contraintes associées (impératives ou indicatives)

Donc l'optimisation multiobjectifs permet de modéliser des problèmes réels faisant intervenir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n'est plus un simple point, mais un ensemble de bons compromis satisfaisant toutes les contraintes [28]**.** Un problème d'optimisation multiobjectifs sous contraintes peut être défini comme suit :

$$C_l(\vec{x})l = \begin{cases} Min|Max\ f_i(\vec{x})i\ =\ 1, \dots, k \\ 1, \dots, m \end{cases}$$

$$x\epsilon\ IR$$

Où « *n* » représente le nombre de variables, « *x* » un vecteur de décision, « *k* » le nombre d'objectifs (critères) et « *m* » le nombre de contraintes du problème. Pour les problèmes d'optimisation multiobjectifs sous contraintes, il faut satisfaire un ensemble de contraintes tout en optimisant plusieurs fonctions objectives. En conséquence, même avec un petit nombre de variables et d'objectifs, le problème ne peut pas être traité simplement avec un algorithme classique de découpe. En effet, l'optimisation d'un problème multiobjectifs est

souvent plus difficile que l'optimisation des problèmes mono objectifs car la solution optimale ne se réduit plus à un seul point, mais à un ensemble de points non dominés [28].

2.4 Problème multiobjectifs

Définition : Un problème multiobjectifs ou multicritère peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectives.

Les problèmes d'optimisation ont en général plusieurs solutions car la définition d'un optimum ne peut pas être établie dans les problèmes multiobjectifs [1].

Une action (ou **un vecteur de décisions)** sera notée :

$x = (x_1, x_2, \dots, x_n)$ avec x_i les variables du problème et n le nombre de variables.

Les contraintes seront notées :

$g_i(x)$ avec $i = 1, \dots, m$ avec m le nombre de contraintes.

Le vecteur de fonction objectif sera noté f :

$f(x) = (f_1(x), f_2(x), \dots, f_k(x))$ avec f_i les objectifs ou critères de décision et « k » le nombre d'objectifs.

Un problème d'optimisation recherche l'action $x*$ telle que les contraintes $g_i(x*)$ soient satisfaites pour $i = 1, \dots, m$ et qui optimise la fonction f :

$$f(x*) = (f_1(x*), f_2(x*), \dots, f_k(x*)).$$

L'union des domaines de définition de chaque variable et les contraintes forment un ensemble « E » que nous appelons l'ensemble des actions réalisables. Nous appellerons « F » l'ensemble des objectifs réalisables comme illustre la figure 1.1.

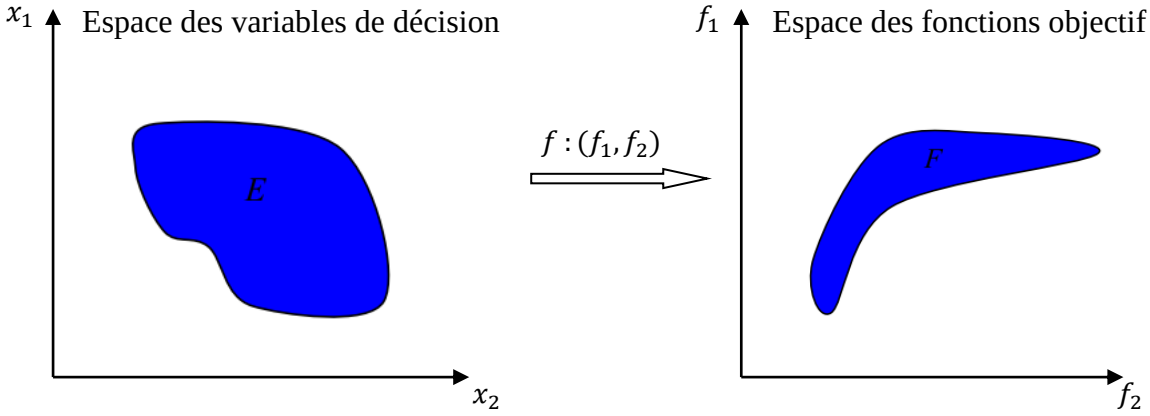

Figure 1.1 L'ensemble des actions réalisables « E » et des objectifs réalisables « F »

Après avoir trouvé les solutions du problème multiobjectifs que d'autres difficultés surviennent : il faut sélectionner une solution dans cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur vis-à-vis des différentes fonctions objectif.

2.4.1 La multiplicité des solutions

Lorsque l'on cherche à obtenir une solution optimale à un problème d'optimisation multiobjectifs donné, on s'attend souvent à trouver une solution et une seule. En fait, on rencontre rarement ce cas de figure. La plupart du temps, on trouve une multitude de solutions, du fait que certains des objectifs sont contradictoires.

Donc, quand on résoudra un problème d'optimisation multiobjectifs, on obtiendra une grande quantité de solutions. Ces solutions, comme on peut s'en douter, ne seront pas optimales, au sens où elles ne minimiseront pas tous les objectifs du problème.

Un concept intéressant, qui nous permettra de définir les solutions obtenues, est le compromis. En effet, les solutions que l'on obtient lorsque l'on a résolu le problème sont des solutions de compromis. Elles minimisent un certain nombre d'objectifs tout en dégradant les performances sur d'autres objectifs.

2.4.2 La difficulté principale d'un problème multiobjectifs

La difficulté principale d'un problème multiobjectifs est qu'il n'existe pas de définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre mais il n'existe pas une solution meilleure que toutes les autres.

Dès lors résoudre un problème multi objectif ne consiste pas à chercher la solution optimale mais l'ensemble des solutions satisfaisantes pour les quelles on ne pourra pas effectuer une opération de classement. Les méthodes de résolution de problèmes multi objectifs sont donc des méthodes d'aide à la décision car le choix final sera laissé au décideur.

2.5 Résolution d'un problème multiobjectifs

Deux types de comportement existent :

Le premier comportement : est de ramener un problème multiobjectifs à un problème simple objectif.

Le second comportement : est de tenter d'apporter des réponses au problème en prenant en compte l'ensemble des critères.

La différence entre ces deux communautés s'exprime dans la figure 1.2.

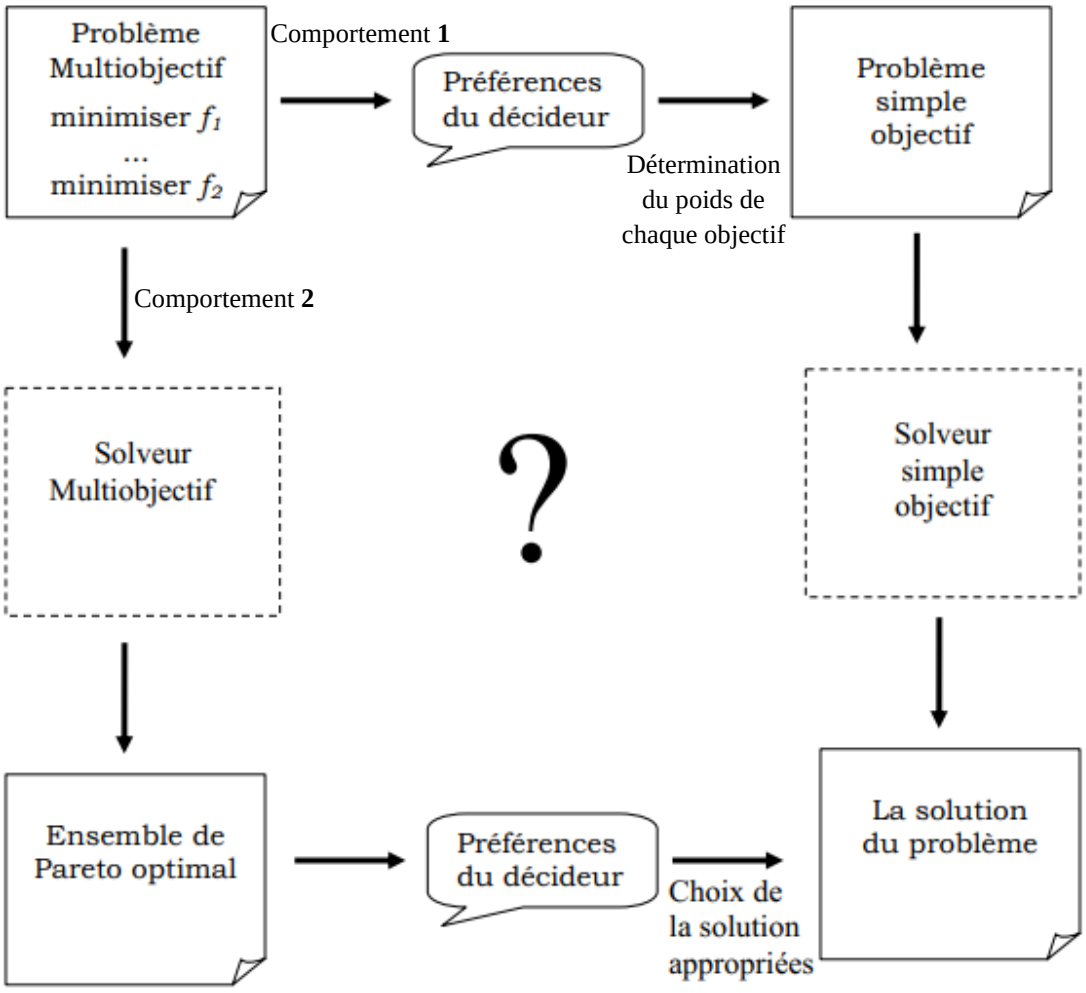

Figure1.2 Quel mode de résolution choisir ?

Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectifs en un problème simple objectif (Comportement 1).

Soit le décideur effectue son choix dans l'ensemble des solutions proposées parle solveur multi objectif (Comportement 2). La principale qualité d'un solveur multiobjectif est donc de rendre les décisions plus faciles en proposant un sous-ensemble représentatif de F (l'ensemble des objectifs réalisables).

2.6 Méthodes de résolution des problèmes d’optimisation

Plusieurs méthodes de résolution existent dans la littérature pour l’optimisation combinatoire. Ces méthodes font partie de deux groupes de nature différente : les méthodes exactes et les méthodes approchées (ou heuristiques).

Les méthodes exactes ont permis de chercher des solutions optimales pour des problèmes de taille raisonnable. Malgré les progrès réalisés, le temps de calcul nécessaire pour trouver une solution optimale risque d’augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante. Ces méthodes sont très exigeantes sur la puissance des ordinateurs.

En revanche, les méthodes approximatives (heuristiques) permettent de trouver une solution dont le coût est proche du coût de la solution optimale. Ils ont l'avantage de permettre en un temps raisonnable de trouver une solution. Ils constituent donc une alternative très intéressante pour traiter les problèmes d’optimisation de grande taille si l’optimalité n’est pas primordiale. Les méthodes approchées sont fondées principalement sur diverses heuristiques, souvent spécifiques à un type de problème.

Une classification des méthodes d’optimisation est illustrée dans la figure 2.1.

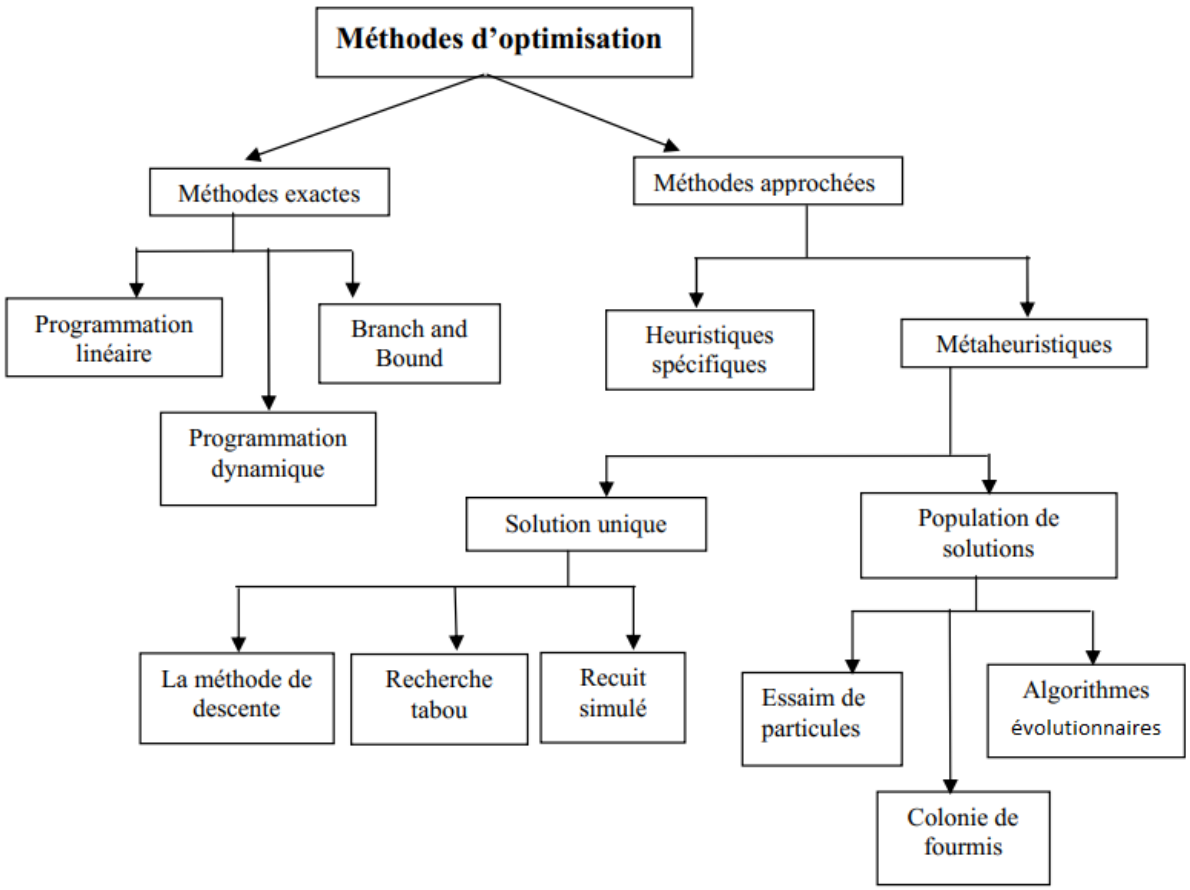

Figure 2.1 Classification possible de méthodes de résolution.

2.6.1 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires représentent une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, énoncée par Charles Darwin en 1859. Le principe fondamental étant que les individus les mieux adaptés à leur environnement survivent et peuvent se reproduire, laissant une descendance qui transmettra leurs gènes. La clef étant l'adaptation des individus face à la pression de l'environnement, l'analogie avec l'optimisation devient claire. Cette adaptation peut alors être assimilée à une optimisation des individus afin qu'ils soient de mieux en mieux adaptés à leur environnement, au fur et à mesure des générations (correspondant aux itérations de l'algorithme).

Nous pourrons alors définir les algorithmes évolutionnaires comme des méthodes faisant évoluer un ensemble de solutions appelé *"population"*. Les solutions, appelées *"individus*", sont représentées par leur génotype, qui s'exprime sous la forme d'un phénotype. Afin d'évaluer la performance d'un individu, on associe au phénotype la valeur de la fonction objectif (ou fonction d'évaluation, fitness). Un algorithme évolutionnaire se décompose en

plusieurs étapes, chacune d'elles étant associée à un opérateur décrivant la façon de manipuler les individus :

- **Évaluation** : on calcule la fitness de chaque individu, par rapport à son phénotype.

- **Sélection** : on sélectionne une partie des descendants en fonction de leur fitness, afin de former une nouvelle population pour la prochaine génération. L'opérateur de sélection le plus simple consiste à prendre les meilleurs individus de la population, en fonction de leur fitness.

- **Croisement** : on croise une partie des individus de la population en "mélangeant" leurs génotypes selon l'opérateur de croisement défini. On obtient ainsi un ensemble de nouveaux individus appelés descendants.

- **Mutation** : les descendants sont mutés, c'est-à-dire que l'on modifie aléatoirement une partie de leur génotype, selon l'opérateur de mutation.

Les algorithmes évolutionnaires englobent différentes familles d'algorithmes qui ne différent que par la structure du génotype des individus ou par les opérateurs utilisés :

- Les stratégies d'évolution: Cette famille de méthodes est chronologiquement la première parmi tous les algorithmes évolutionnaires. Dans ces algorithmes, l'opérateur de mutation utilise une distribution normale pour ajouter une valeur aléatoire à chaque composante du génotype.

- La programmation évolutionnaire: un an après l'apparition des stratégies d'évolution, cette famille voit le jour dans le cadre de nouveaux concepts d'apprentissage en intelligence artificielle, pour la création d'automates à états finis. Chaque individu de la population est vu comme appartenant à une espèce distincte. Lors de la phase de reproduction, chaque individu génère donc un descendant.

- Les algorithmes génétiques: ce sont les algorithmes évolutionnaires les plus connus et les plus répandus. Ces algorithmes se détachent en grande partie par la représentation des données du génotype, initialement sous forme d'un vecteur binaire et plus généralement sous forme d'une chaîne de caractères.

- La programmation génétique: c'est la dernière famille d'algorithmes évolutionnaires à avoir vu le jour. Elle a longtemps été assimilée à un sous-groupe des algorithmes génétiques,

car elle n'en diffère que par la représentation des données. Le génotype est effectivement représenté sous la forme d'un arbre plutôt qu'une chaîne.

3. La programmation génétique

3.1 Définition

La programmation génétique est une technique qui permet aux ordinateurs de résoudre les problèmes sans être explicitement programmés [16].

Dans la programmation génétique les individus dans la population sont des programmes d'ordinateur. Pour détendre le processus de création de nouveaux programmes à partir de deux programmes parents, les programmes sont écrits sous forme d'arbres. Les nouveaux programmes sont produis en supprimant des branches à partir d'un arbre et l'insérer à un autre. Ce processus simple, connu croisement, assure que le nouveau programme est aussi un arbre et syntaxiquement valide.

Les opérateurs utilisés dans le processus de la programmation génétique sont les mêmes utilisés dans les algorithmes génétiques avec le même principe : le croisement, la mutation, la reproduction, la duplication génétique et la sélection. La seule différence est que les chromosomes ici sont des programmes d'ordinateurs.

3.2 Représentation d'une solution de la programmation génétique

Dans un environnement de programmation génétique, les programmes informatiques sont représentés dans des structures arborescentes qui sont évaluées de manière récursive pour produire les expressions multivariées résultantes. La nomenclature traditionnelle indique qu'un nœud d'arbre (ou juste un nœud) est un opérateur [+, -, *, /] et un nœud terminal (ou une feuille) est une variable [a, b, c, d].

Lisp a été le premier langage de programmation appliqué à la programmation génétique basée sur l'arbre, car la structure de ce langage correspond à la structure des arbres. Cependant, de nombreux autres langages tels que Python, Java et C ++ ont été utilisés pour développer des applications de programmation génétique arborescentes.

La programmation génétique basée sur l'arbre a été la première application de la programmation génétique. Il existe plusieurs autres types tels que linéaire, cartésien, et à base de pile qui sont généralement plus efficaces dans leur exécution des opérateurs génétiques.

Cependant, la programmation génétique basée sur l'arbre fournit un moyen visuel d'engager de nouveaux utilisateurs de la programmation génétique, et reste viable lorsqu'il est construit ; La figure 2.2 représente un exemple de programme en structure arboréssante.

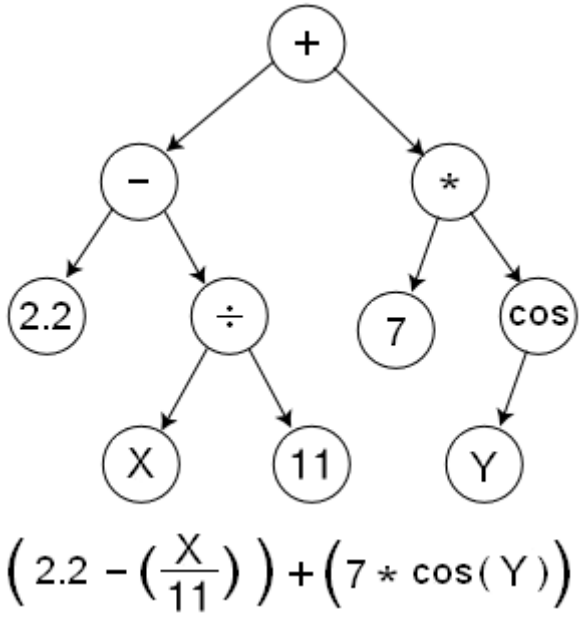

Figure 2.2 Représentation d'un programme en arbre « Lisp ».

3.3 Applications de la programmation génétique

Il y a de nombreuses applications de la programmation génétique :

• **Problèmes de magie noire :** tels que la synthèse automatisée des circuits électriques analogues, des contrôleurs, des antennes, des réseaux des réactions chimiques, et d'autres secteurs de conception.

• **La programmation de l'improgrammable :** comportant la création automatique des programmes machine pour les dispositifs de calcul peu usuels tels que les automates cellulaires, systèmes de multi-agent, systèmes parallèles, rangées de porte field programmable, rangées field-programmable d'analogue, colonies de fourmi, l'intelligence d'essaim, a distribué des systèmes, et des semblables.

• **Nouvelles inventions commercialement utilisables :** comportant l'utilisation de la programmation génétique comme « machine d'invention » automatisée pour créer de nouvelles inventions commercialement utilisables.

• Reconnaissance d'images, classification d'images, traitement d'images satellites, ...

• Prédiction de séries temporelles, génération d'arbres de décisions, datamining, …

• Classification de segments d'ADN, de protéines, ...

3.4 Les phases de la programmation génétique

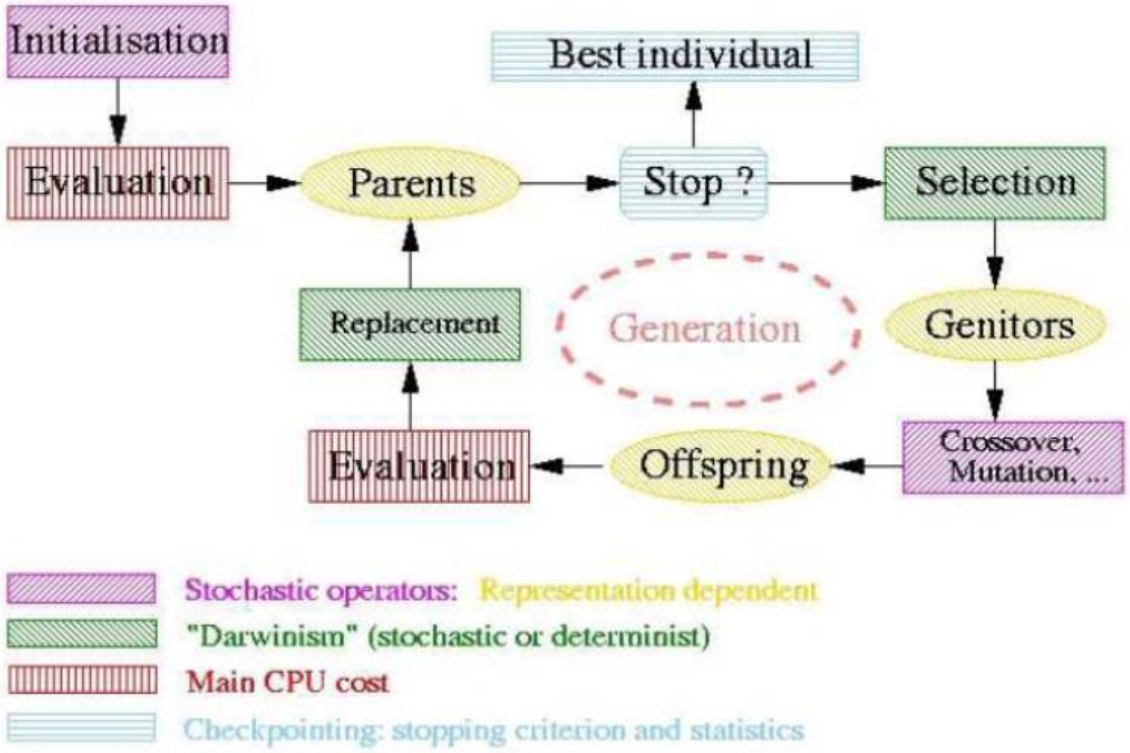

Figure 2.3 Cycle de fonctionnement d'un programme génétique.

La figure 2.3 représente le cycle de fonctionnement d'un programme génétique. Dans un premier temps, il nous faut une base pour pouvoir commencer à générer des programmes (phase initialisation). On obtiendra donc un certain nombre d'individus qui permettront de générer les générations futures. A ce moment, on vérifie si l'une des solutions que nous offrent ces individus est satisfaisante (Evaluation). Si aucune solution ne convient, on va alors procéder à une sélection des meilleurs afin de générer par différentes techniques les descendants. (Phase sélection, crossover et mutation). Enfin, ces descendants vont venir remplacer la génération précédente en étant à leur tour les parents, et le cycle recommence alors par le bloc Evaluation.

4. Conclusion

Comme nous avons pu le voir tout au long de ce chapitre, l’optimisation multiobjectifs n’est pas une tâche facile. L’optimisation multiobjectifs offre à l’utilisateur davantage de degrés de liberté pour modéliser son problème.

Dans ce chapitre, nous avons introduit les concepts généraux de l’optimisation, et l’optimisation multiobjectifs. Ensuite on a également présenté les différentes approches de résolution de problème d’optimisation multiobjectif, et enfin nous avons passé en revue à la programmation génétique comme une approche métaheuristique de résolution, a l’inverse des méthodes exactes, ces méthodes sont très générique, elles peuvent s’adapter à n’importe quel type de problème même si le nombre d’objectif est important.

Chapitre 2

Cloud Computing et ordonnancement de taches

1. Introduction

Le cloud computing est un terme à la mode dans la communauté de recherche, mais l'idée derrière ce concept est plus ancienne que nous le poussons. Les chercheurs ont toujours rêvé à développer des systèmes informatiques qui sont consommés directement par les consommateurs à l'aide d'une connexion entre ces consommateurs et les entreprises propriétaires de ces systèmes, ce principe représente le cœur de fonctionnement de plateforme de cloud computing actuellement.

Dans le présent chapitre on va présenter un état de l'art sur le cloud computing en présentant tous ce qui considéré important dans ce domaine, et on va focaliser sur l'allocation des ressources dans le cloud computing parce que c'est le sujet de cette thèse.

2. Cloud computing

Parce que le cloud computing est nouveau axe de recherche et les chercheurs sont encore mené des recherches dans cette domaine, il n'y pas une définition standard de ce terme qui est accepté par tous les acteurs liés avec le cloud computing. C'est pour cela Vaquero et ces collègues [20] ont essayé de standardisé la définition de cloud en étudions et analysons 22 définitions qui existent dans la littérature. Le résultat de cette effort est la définition suivante ;

Les clouds sont une grande réserve de ressources virtuelles faciles à accéder et à utiliser (comme les matériels, les plateformes de développement). Ces ressources peuvent être reconfigurées dynamiquement pour s'adapter à des situations variables, permettant aussi une utilisation optimale de ces ressources. Typiquement cette réserve de ressources est exploitée par un modèle de paiement à l'utilisation dont laquelle des garanties sont offertes par le fournisseur de l'infrastructure [20].

Une autre définition très intéressant proposé par **George Reese** dans [12] dont lequel l'auteur propose un moyen pour différencier entre un service ordinaire et un service de cloud :

Le cloud n'est pas simplement le dernier terme à la mode pour Internet. Bien que l'Internet soit une base nécessaire au cloud, le cloud est plus que l'Internet. Le nuage est l'endroit où vous allez utiliser la technologie quand vous en avez besoin, aussi longtemps que vous en avez besoin, et non minute plus. Vous n'installez rien sur votre bureau et vous ne payez pas pour le la technologie lorsque vous ne l'utilisez pas.

Le cloud peut être à la fois logiciel et infrastructure. Ce peut être une application à laquelle vous accédez via le Web ou un serveur que vous mettez en service exactement quand vous en avez besoin. Si un service est un logiciel ou matériel, voici un test simple permettant de déterminer si ce service est un service cloud.

Si vous pouvez vous promener dans n'importe quelle bibliothèque ou café Internet et vous asseoir devant n'importe quel ordinateur sans préférence pour le système d'exploitation ou le navigateur et accéder à un service, ce service est basé sur le cloud.

J'ai défini trois critères que j'utilise dans les discussions pour savoir si un service donné est un cloud:

• Le service est accessible via un navigateur Web (non propriétaire) ou une API de services Web.

• Zéro dépense en capital est nécessaire pour commencer.

• Vous ne payez que ce que vous utilisez comme vous l'utilisez.

La définition de cloud computing proposé par l'Institut national des normes et de la technologie (National Institute of Standards and Technology ou NIST) est considéré comme la définition la plus citée dans les littératures, tel que pour le NIST.

Le cloud computing est un modèle Informatique qui permet un accès facile et à la demande par le réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service [24].

2.1 Les caractéristiques de Cloud Computing

Le Cloud Computing modèle définit par le NIST contient essentiellement cinq caractéristiques, à savoir :

- **Accès à la demande par le consommateur**

Les ressources et les services offerts par les fournisseurs sont toujours disponibles à la demande des utilisateurs.

- **Large accès au réseau**

Le cloud computing utilise au possible les technologies les plus standardisées (essentiellement l'Internet), afin de rendre l'accès possible en utilisant n'importe quel appareil technologique.

- **Réservoir de ressources (Resource pooling)**

La virtualisation est une pile principale pour construire une solution cloud computing, alors les ressources d'une plateforme cloud peuvent être physiques ou virtuelles, elles sont partagés et dynamiquement allouées afin de satisfaire les demandes des utilisateurs.

- **Redimensionnement rapide (élasticité)**

En fonction de la demande, les ressources et les capacités peuvent être vendues rapidement et même dans certains cas automatiquement, provisionnées et libérées élastiquement. Pour le consommateur, les capacités disponibles pour l'approvisionnement

semblent souvent illimitées et peuvent être appropriées à n'importe quelle quantité à n'importe quel moment.

- **Paiement à l'usage**

Les services et ressources sont payés à l'usage selon la durée et la quantité d'utilisation. Cette faculté permet de diminuer le coût d'utilisation pour les consommateurs.

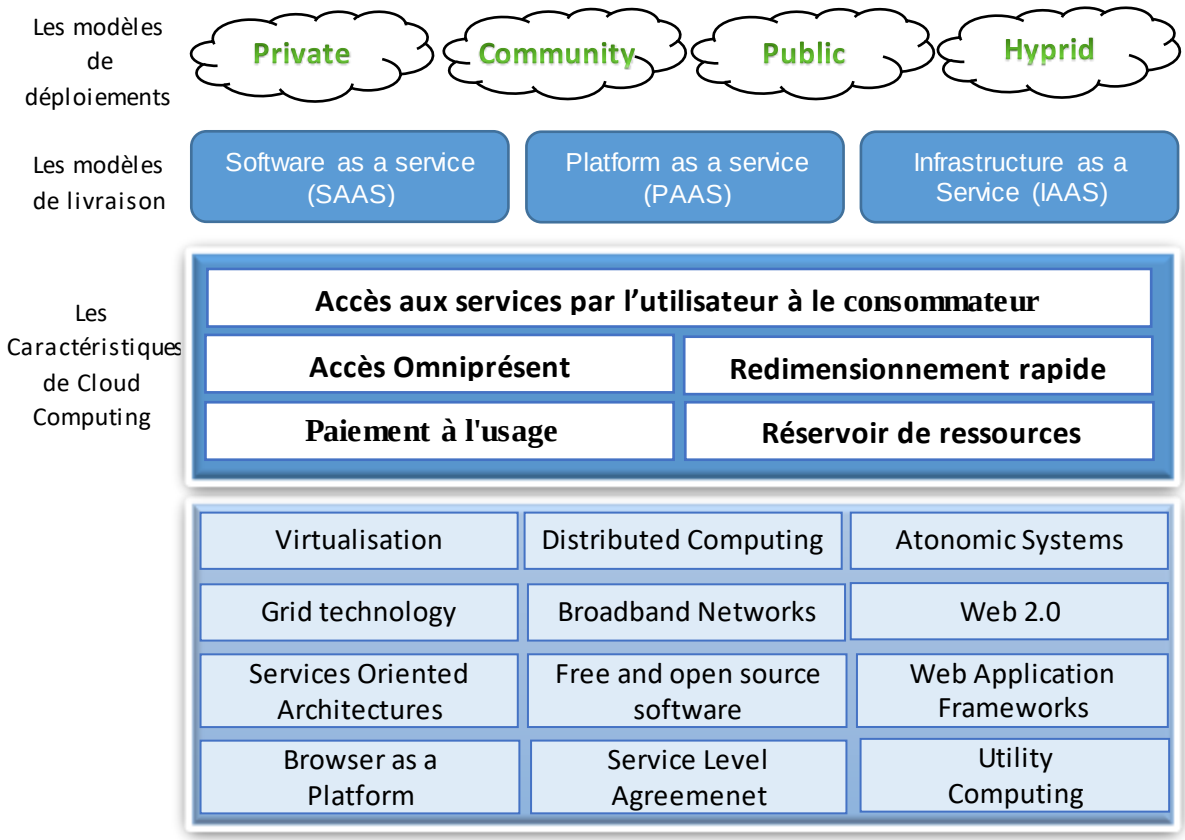

Figure 2.1 Le modèle de cloud computing.

2.2 Les modèles de livraison

Dans le cloud computing les fonctionnalités sont offertes aux consommateurs comme des services. Ces services peuvent être devisés selon la nature du service livré en trois grands groupes (figure 2.1), à suivre :

2.2.1Software as a Service (SAAS)

Sont des services pour les clients de ce cloud. Ces services peuvent être des applications traditionnelles et simples comme le traitement de texte. C'est le modèle de livraison de service le plus utilisé dans le cloud, tel que SaaS qui permet aux consommateurs

d'exploiter des applications à distance du cloud [24]. Par exemple les applications de gestions des emails (Gmail de google, Yahoo mail) sont une sorte de SaaS [24].

Les avantages de SaaS

Coût : Du côté matériel, le coût est notablement réduit car l'entreprise remplace la complexe infrastructure par un simple réseau local connecté avec l'Internet. De côté logiciel, au lieu de payer l'application d'un seul coup, avec SaaS le coût est réglé chaque période de temps et le paiement est à l'usage. Le coût de maintenance est diminué, parce que il est partagé entre tous les clients de ce service [24].

Temps : en comparaison avec les traditionnels modèles, dans SaaS les matériels et les applications (services) sont déjà installés et configurés qui permet de gagner un temps considérable.

Misa à niveau facile : parce que les services sont hébergés chez les fournisseurs, la mise à niveau des applications ne nécessite aucun changement de matériel ou bien téléchargement de logiciels complémentaires chez les clients et toutes les charges sont sur les fournisseurs.

2.2.2 Platform as a Service (PaaS)

PaaS permet aux consommateurs de développer et déployer sur une infrastructure cloud des applications en utilisant des langages et outils de programmation fournis par le fournisseur. Le consommateur ne gère pas et ne contrôle pas l'infrastructure interne de cloud tel que le réseau, les serveurs, les systèmes d'exploitation, le stockage, mais il a un control sur les applications déployées [24].

PaaS offre un environnement de développement pour créer des applications pour les utiliser sur le cloud. La différence essentielle avec un SaaS est que le SaaS est destiné aux consommateurs, par contre le PaaS est destiné aux développeurs. Les fournisseurs de PaaS les plus connus sont : Google App Engine et Microsoft Azure

2.2.3 Infrastructure as a Service (IaaS)

C'est la possibilité pour les consommateurs d'utiliser des ressources liées aux traitements, le stockage, les réseaux et d'autres ressources informatiques fondamentales. Le consommateur sera capable de déployer et d'exécuter des logiciels arbitraires, ce qui peut inclure des systèmes d'exploitation et des applications. Le consommateur ne gère pas et ne contrôle pas l'infrastructure de cloud, mais il contrôle le système d'exploitation, le stockage et

les applications déployées avec un contrôle limité des composants du réseau sélectionnés (par exemple, des pare-feu hôtes) [24].

2.3 Les modèles de déploiement

Le modèle de déploiement est habituellement divisé en quatre modes : public, privé, communautaire et le cloud hybride. Leurs descriptions seront présentées dans ce qui suit :

2.3.1 Le Cloud Privé

L'infrastructure du cloud privé est provisionnée pour une utilisation exclusive par une seule organisation avec plusieurs consommateurs. Le cloud privé peut être appartenir, être géré et être exploité par l'organisation, par un tiers ou par une combinaison de ces derniers, et il peut exister à l'intérieur ou à l'extérieur des locaux de l'organisation [24].

2.3.2 Le Cloud public

L'infrastructure de cloud public est accessible par l'Internet. Elle est ouverte au public ou à de grands groupes industriels. Il peut appartenir, être géré et être exploité par une entreprise de commerce, par une organisation académique, ou par une organisation gouvernementale. Il existe sur les lieux du fournisseur du cloud public [24].

2.3.3 Le Cloud communautaire

L'infrastructure du cloud communautaire est provisionnée pour une utilisation exclusive par une communauté spécifique de consommateurs qui partagent les mêmes domaines d'intérêts. Il peut appartenir, être géré et être exploité par une organisation ou plus dans la communauté, par un tiers ou par une combinaison d'eux [24].

2.3.4 Le Cloud hybride

Le cloud hybride est une composition de deux ou plusieurs types de clouds (privé, public et communautaire) [24].

2.4 Un modèle en couches de Cloud computing

D'une manière générale, l'architecture d'un environnement cloud peut être divisé en 4 couches: le matériel (couche de Datacenter), la couche infrastructure, la couche plate-forme et la couche d'application, comme indiqué sur la Figure 2.2 [25].

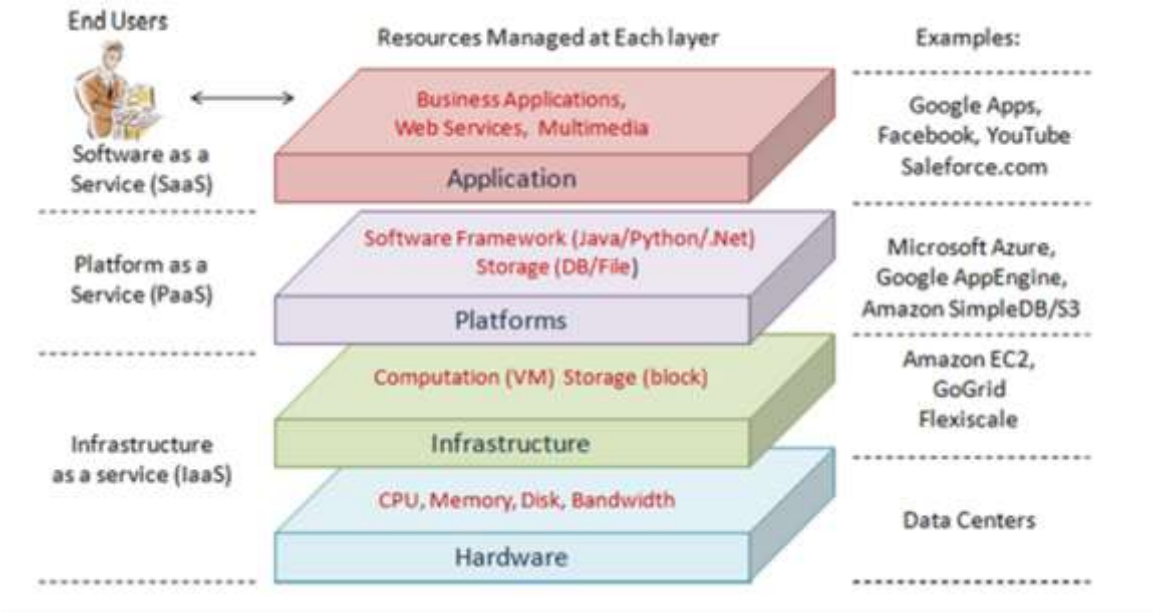

Figure 2.2 : Modèle en couches de Cloud computing

2.4.1 La couche matérielle

Cette couche est responsable de la gestion des ressources physiques du cloud, y compris les serveurs physiques, les routeurs, les commutateurs, les systèmes d'alimentation et de refroidissement. En pratique, la couche matérielle est généralement implémentée dans les Datacenters. Un Datacenter contient généralement des milliers de serveurs organisés en racks et interconnectés à travers des commutateurs, des routeurs ou d'autres tissus. Les problèmes fréquemment rencontrer à la couche matérielle comprennent la configuration matérielle, la tolérance aux pannes, la gestion du trafic, la gestion de l'énergie et refroidissement des ressources.

2.4.2 La couche infrastructure

Aussi appelée la couche de la virtualisation, la couche infrastructure crée un pool de stockage et des ressources informatiques en partitionnant les ressources physiques utilisant des technologies de virtualisation telles que Xen[5], KVM [6] et VM ware [11]. La couche infrastructure est un composant essentiel de cloud computing, car de nombreuses fonctionnalités telles que l'affectation dynamique des ressources sont disponibles via les technologies de virtualisation.

2.4.3 La couche de plate-forme

Construite sur la couche infrastructure, la couche plate-forme se compose de systèmes d'exploitation et cadres d'application. Le but de la couche de plate-forme est de minimiser la charge de déploiement d'applications directement dans des conteneurs VM. Par exemple, Google App Engine fonctionne au niveau de la couche plate-forme pour fournir un support API pour la développent des applications cloud.

2.5 Quoi de neuf dans le cloud?

La question la plus posé quand on lit ou bien on entend pour la première fois quelqu'un parle le cloud computing, le cloud computing peut être considéré comme un partage de temps « time sharing » ou la capacité de partager des ressources informatiques entre de nombreux utilisateurs différents. Mais au début de l'informatique plusieurs entreprises exploit cette possibilité de partager un seul ordinateur par ces employées, cet ordinateur peut allouer et gérer les ressources selon les besoins de ces employées, et cela nous revois à la question précédente : pour quoi le cloud computing est considéré comme une nouvelle idée ?

Selon [8] la réponse à cette question peut être clarifiés sur trois point :

- Le premier est la capacité à exploiter des composants de différentes ressources de cloud et combinez les solutions que vous recherchez. Vous pouvez tirer parti de storageas- un service d'un fournisseur, la base de données en tant que service d'un autre, et même une plate-forme complète de développement et de déploiement d'applications à partir d'un troisième. Cette capacité à exploiter uniquement les ressources dont vous avez besoin parmi les solutions que vous souhaitez proposer, ainsi que dans les quantités appropriées, constitue une valeur évidente de cloud computing moderne.
- Deuxièmement, la banalisation de la bande passante permet aux entreprises tirer parti des ressources informatiques en nuage comme si elles étaient locales. Ainsi, vous pouvez tirer parti les ressources de stockage et d'exécution comme si elles existaient dans votre centre de données, quelque chose qui était difficile il y a quelques années.
- Enfin, il y a la disponibilité de fournisseurs de cloud computing très innovants. Bien que l'architecture et le modèle de cloud computing ne soient pas nouveaux, les acteurs du cloud computing qui fournissent les services sont, y compris l'infrastructure- des acteurs en tant que service, tels que l'EC2 d'Amazon et la plate-forme en tant que

service des joueurs tels que Google App Engine. Avec le cloud computing de plus en plus par à pas de géant, des services de cloud computing de meilleure qualité et plus innovants sont en cours de construction et libéré en permanence.

2.6 Datacenter

Le terme «Datacenter» signifie différemment pour différentes personnes. Parmi les noms utilisés, citons centre de données, hall de données, ferme, entrepôt de données, salle informatique, salle de serveur, R & D laboratoire logiciel, laboratoire haute performance, hébergement, colocation, etc. L'Environment Protection Agency des États-Unis définit un Datacenter comme:

- Principalement les équipements électroniques utilisés pour le traitement des données (serveurs), stockage de données (équipement de stockage), et communications (équipement de réseau). Collectivement cet équipement traite, stocke et transmet des données numériques (information) [13].
- Équipements spécialisés très puissant de conversion et de restauration maintenir une alimentation fiable et de haute qualité, ainsi qu'équipement de contrôle de l'environnement pour maintenir la température et humidité appropriées pour les équipement électronique [13].

2.6.1 L'architecture de Datacenter

Le Datacenter héberge la puissance de calcul, le stockage et les applications nécessaires pour soutenir une entreprise. L'infrastructure du Datacenter est au cœur de l'architecture informatique, à partir de laquelle tout le contenu est extrait ou transmis. Une planification adéquate de la conception de l'infrastructure Datacenter est essentielle, et les performances, la résilience et l'évolutivité doivent être soigneusement prises en compte.

Un autre aspect important de la conception du centre de données est la flexibilité permettant de déployer et de prendre en charge rapidement de nouveaux services. Concevoir une architecture flexible capable de prendre en charge de nouvelles applications dans un délai court peut offrir un avantage concurrentiel significatif. Une telle conception nécessite une planification initiale solide et une réflexion approfondie sur les domaines de la densité de ports, de la bande passante de liaison montante de la couche d'accès, de la capacité réelle du serveur et de la sursouscription, pour n'en nommer que quelques-uns.

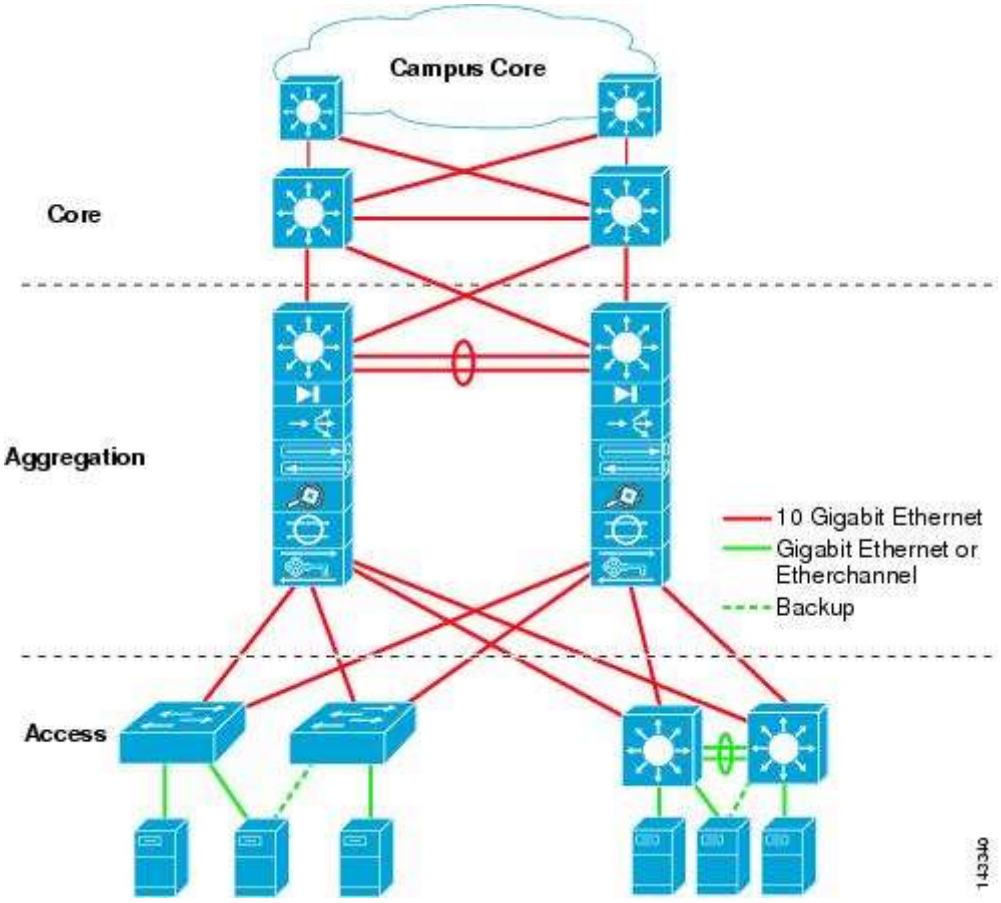

Figure 2.3 : Conception en couches de base de Datacenter.

La conception du réseau de centres de données repose sur une approche en couches éprouvée, qui a été testée et améliorée au cours des dernières années dans certaines des plus grandes implémentations de centres de données au monde. L'approche en couches est la base de la conception du centre de données qui vise à améliorer l'évolutivité, les performances, la flexibilité, la résilience et la maintenance. La figure 2.3 illustre la conception en couches de base [6].

2.7 La virtualisation

La virtualisation est un outil important de conception de système informatique, aujourd'hui plusieurs domaines utilisent les machines virtuelles comme les systèmes d'exploitation, les langages de programmation et l'architectures de processeur. La virtualisation donne aux utilisateurs et aux développeurs plus de flexibilité en les libérant de l'interface traditionnelle et des contraintes de ressources,

2.8 L'ordonnancement de taches dans le cloud computing

2.8.1 Définition de problème

Le cloud computing est un nouveau modèle de calcul, dans lequel les ressources sont partagées en utilisant des technologies de virtualisation sur Internet. Il rend les ressources nécessaires du travail manifestes sous la forme d'une machine virtuelle. Le logiciel de virtualisation open source tel que Xen est utilisé pour le Cloud fournissant une infrastructure informatique virtualisée afin que plusieurs applications puissent partager les mêmes ressources et chaque application s'exécute dans un nœud de cloud qui est un ensemble de ressources de calcul avec des capacités limitées. Chaque application est complètement différente et indépendante et n'a aucun lien entre elles. Les nœuds cloud sont également des indépendants. Chaque nœud cloud est planifié comme une unité d'indépendance

Le cadre général du cloud est illustré dans la **figure 2.4**. Les utilisateurs soumettent leurs travaux au Job Broker qui gère les métadonnées des travaux, telles que les exigences de traitement. Le gestionnaire de nœuds cloud surveille les nœuds cloud, en même temps, le gestionnaire de nœuds cloud maintient les états des nœuds cloud et les attributs des nœuds cloud, tels que la vitesse de calcul. Le gestionnaire de tâche gère les états des taches et les informations de planification qui sont calculés par les exigences de traitement des tâches et la vitesse de calcul des nœuds cloud. Il est généralement facile d'obtenir des informations sur la vitesse de calcul des nœuds cloud, mais il est assez difficile de déterminer les exigences de traitement des tâches. Pour résoudre ce problème, les exigences de traitement des tâches sont estimées de manière dynamique en fonction des longueurs des tâches à partir des spécifications de l'application utilisateur ou des données historiques. Le planificateur de travaux mappe le travail à des intervalles de temps spécifiques sur les nœuds cloud en fonction des états des travaux, des états des nœuds cloud, des informations de planification des tâches. La planification des tâches consiste à mapper la tâche au nœud cloud en fonction de l'objet de l'optimisation. Il existe plusieurs critères utilisés pour évaluer l'efficacité et l'efficacité de l'algorithme de planification des tâches. Les plus populaires sont makespan et flowtime (temps d'écoulement).

Afin de formaliser le problème, nous expliquons la définition de la terminologie pertinente à notre problème objectif. Un nœud cloud (CN) est un ensemble de ressources de calcul avec une vitesse de calcul limitée sur le cloud. Une tâche est considérée comme un ensemble unique d'applications utilisateur. Un calendrier est le mappage des tâches à des

intervalles de temps spécifiques sur le CN. On suppose que l'ensemble de tâches est : $J = \{J_1, J_2, \dots, J_n\}$, pour toute *i*, *j* ($i \neq j$), *Ji* et *Jj* ne sont pas pertinents et la préemption n'est pas autorisée. Chaque tâche *Jj* a son correspondant longueur (exigence de traitement). L'ensemble du longueur du tâche est $L = \{L_1, L_2, \dots, L_n\}$, L_j est le longueur du tâche J_j, l'unité de $L_j (i = 1,2, \dots, n)$ est le nombre de cycles. L'ensemble CN est $C = \{C_1, C_2, \dots, C_m\}$. Chaque CN a la vitesse de calcul correspondante. L'ensemble du vitesse du CN est $V = \{v_1, v_2, \dots, v_m\}$, l'unité de v_i $(i = 1,2, \dots, m)$ est le nombre de cycles par unité de temps (CPUT), v_i est la vitesse de CN C_i. Les tâches individuels J_j $(j = 1,2, \dots, n)$ doit être traité jusqu'à son achèvement sur un seul CN. Nous définissons $A = (a_{ij})(i = 1,2, \dots, m;\ j = 1,2, \dots, n)$ comme le temps qu'il faut au CN C_i pour terminer la tâche J_j.

$$a_{ij} = \frac{l_j}{v_j}$$

La matrice d'affectation binaire$X = (x_{ij})$ $(i = 1,2, \dots, m; j = 1,2, \dots, n)$ est utilisée pour désigner les variables de décision.

$$x_{ij} = \begin{cases} 1 & si\ J_j \text{est affecté au nœud de cloud } Ci \\ 0 & \text{autrement} \end{cases}$$

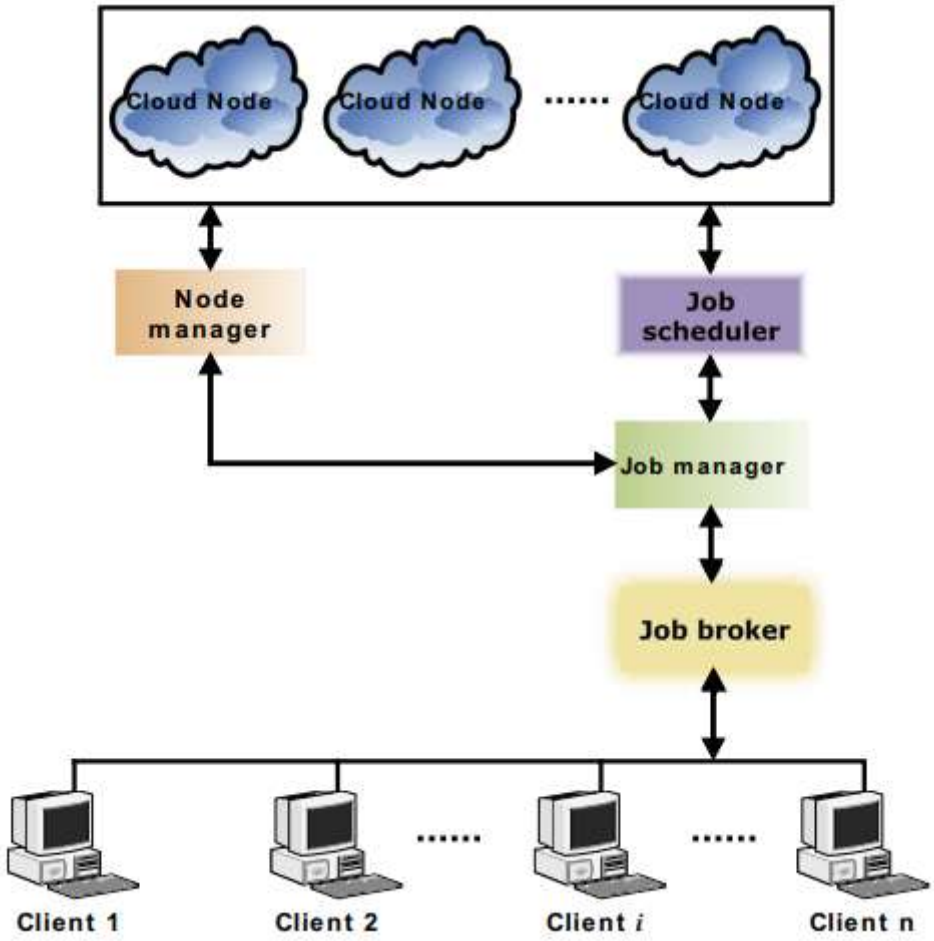

Figure2.4 Cadre général du cloud et des systèmes de tâche.

Étant donné que chaque tâche ne peut être exécutée que sur un seul nœud cloud, l'équation suivante doit être satisfaite.

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

Afin de simplifier le modèle, les temps de configuration et de transfert sont ignorés. La planification de tâches consiste à mapper $J_j(j = 1,2, \dots, n)$ à $C_i(i = 1,2, \dots, m)$ afin de minimiser le temps d'achèvement. Par exemple, nous planifions 13 jobs $(J_1 \sim J_{13})$sur 3 nœuds (C_1, C_2, C_3) L'ensemble de vitesse du CN est $S = \{4,3,2\}$ et l'ensemble de longueur du tâche est $L = \{6,12,16,20,24,28,30,36,40,42,48,52,60\}$. Sur la figure 2.2(a), les tâches J_3, J_4, J_6, J_7, J_8 et J_9 sont alloués sur le nœud cloud 1; J_5, J_{11} et J_{13} sont alloués sur le nœud cloud 2; J_1, J_2, J_{10} et J_{12}sont alloués sur le nœud cloud 3. Le temps d'exécution du tâche $a_{1,3} = 4, a_{1,4} = 5,\ a_{1,6} = 7,\ a_{1,7} = 7.5,\ a_{1,8} = 9,\ a_{1,9} = 10; a_{2,5} = 8,\ a_{2,11} = 16,\ a_{2,13} = 20;\ a_{3,1} = 3, a_{3,2} = 6,\ a_{3,10} = 21,\ a_{3,12} = 26.$ Pour le nœud cloud 1, son temps

d'écoulement nœud unique $\sum_{j=1}^{N} a_{1j} = 42.5$.Les deux autres temps d'écoulement du cloud nœuds sont respectivement de 44 et 56. Dans la planification des tâches, le plus long temps d'écoulement du nœud unique est appelé makespan, et la somme de tous les temps d'écoulement d'un seul nœud est le temps d'écoulement planifié. Ainsi, le makespan est de 56 et le temps d'écoulement de planification est de 142,5.

Une planification optimal est illustrée sur la figure 2.2 (b), dans laquelle les tâches J_3, J_7, J_8, J_{10} et J_{13}sont alloués au cloud nœud 1 ; J_1, J_4, J_5, J_9 et J_{11} sont alloués au cloud nœud 2 ; J_2, J_6 et J_{12} sont alloués au cloudnœud 3 ; le temps d'exécution du tâche $a_{1,3} = 4, a_{1,7} = 7.5,\ a_{1,8} = 9,\ a_{1,10} = 10.5,\ a_{1,13} = 15;\ a_{2,1} = 2,\ a_{2,4} = 6.6667,\ a_{2,5} = 8,\ a_{2,9} = 13.3333,\ a_{2,11} = 16;\ a_{3,2} = 6, a_{3,6} = 14, a_{3,12} = 26.$

Le temps d'écoulement du nœud unique est $\sum_{j=1}^{N} a_{1j} = 46$. Les deux autres temps d'écoulement de cloud nœuds sont également 46. Dans cette solution optimale de planification, le makespan est de 46 et le temps d'écoulement du programme est 138. Makespan *MS* est le plus long parmi tous les temps d'écoulement d'un seul nœud, et le temps d'écoulement planifié *SF* est la somme du temps de finalisation de toutes les tâches. En planification de job shop, une agrégation pondérée est souvent utilisée pour atteindre un équilibre entre ces deux métriques, à savoir:

$$f = w_1 * MS + w_2 * SF$$

Où w_1 et w_2 sont des poids non négatifs et w_1+w_2= 1. Les poids peuvent être fixes ou adaptés dynamiquement pendant le processus d'optimisation. Malheureusement, *MS*≪*SF* dans un environnement cloud à grande échelle. En d'autres termes, *MS* disparaîtrait de *SF* sauf si w_1 est beaucoup plus grand que w_2. Donc w_1= 1 et w_1= 0 pour la planification des tâches dans le cloud computing. Une planification optimale est celui qui optimise le makespan, comme illustré sur la **figure 2.2 (b).** Notre objectif de planification des tâches est de minimiser le makespan.

Pour toute planification *X*, le makespan *MS* est formalisé comme suit :

$$MS(X) = \max_{i \in \{1,2,\ldots,m\}} \sum_{j=1}^{n} a_{ij} \times x_{ij}$$

L'objectif de planification des travaux est de rechercher le programme *X* qui minimise la *MS* tout en satisfaisant les contraintes de faisabilité du programme. C'est-à-dire que le problème est formalisé comme suit :

$$Obj : f(X) = \min \max_{i \in \{1,2,\dots,m\}} \sum_{j=1}^{n} a_{ij} \times x_{ij}$$

$$x_{ij} \in \{0,1\} (i = 1,2,\dots,m; j = 1,2,\dots,n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

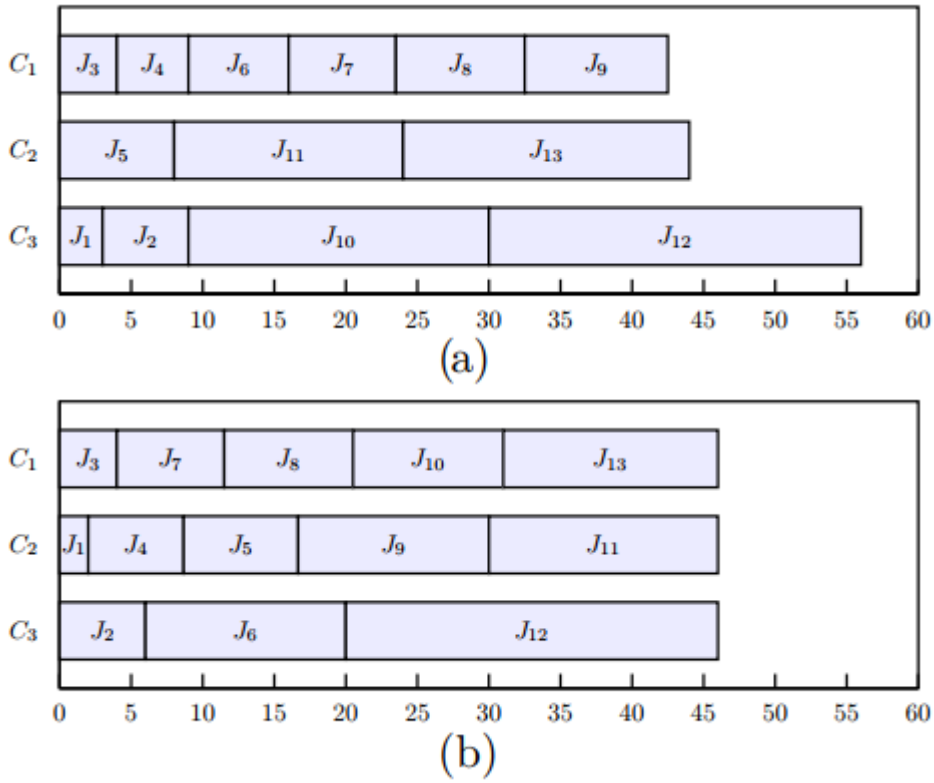

Figure2.5 Un exemple de makespan et flowtime.

3. Conclusion

Le développement du cloud computing passera certainement par son adoption au sein des entreprises, pour qui les offres commencent à être nombreuses. La rentabilité étant l'objectif numéro un.

Arriver à répondre de manière rapides et efficaces aux demandes croissantes des utilisateurs, les entreprises ou fournisseurs de clouds doivent améliorer constamment les algorithmes d'exécution des tâches et améliorer la qualité de services. La théorie d'ordonnancement de tâches et allocation de ressources dans les systèmes de cloud computing suscite une attention croissante avec l'augmentation de la popularité de cloud.

Pour cela, on a introduit dans ce chapitre les notions de base de cloud computing, les définitions, les caractéristiques, les modèles des clouds. Et on terminer par l'ordonnancement de tâches où on a défini le problème étudié.

Chapitre 3

Application de validation

1. Introduction :

Ce chapitre est consacré à l'outil d'évaluation choisi pour cette thèse : le simulateur CloudSim, développé au CLOUDS Laboratory de Melbourne en Australie.

Ce simulateur a été choisi pour ses qualités de modélisation. Au cours de ce chapitre, une partie est consacrée aux évolutions apportées à CloudSim, afin d'y intégrer des fonctionnalités indispensables, et ainsi de faire évoluer ses caractéristiques dans le but d'obtenir un simulateur permettant d'exécuter des simulations qui donne des solutions au problème de répartition des charges. Le principe est d'essayer l'optimisation d'attribution des ressources de cloud par l'application d'un algorithme de répartition des taches sur machines virtuelles (Branch an Bound), l'affectation (équilibrée) automatique et dynamique entre VM pour assurer l'utilisation optimale des machines virtuelles des travaux qui sont conçus au sein de CloudSim afin de rendre ce simulateur complet du point du vue de l'intégration et de l'utilisation des outils de répartition des charges. Tout cela afin de pouvoir mener les phases d'évaluation présentées dans le chapitre suivant avec toutes les fonctionnalités nécessaires. Ainsi, les différentes sections de ce chapitre présentent tout d'abord de manière détaillée l'architecture et les caractéristiques de base de CloudSim, puis une description précise des évolutions qui lui ont été apportées.

2. Le simulateur *CloudSim*

CloudSim, projet développé par le CLOUDS Laboratory de Melbourne en Australie, est un outil de simulation extensible permettant la modélisation et la simulation d'environnement de systèmes de type Cloud Computing de niveau IaaS. Cette section d'introduction aux fonctionnalités de CloudSim est largement inspirée de l'article présentant en détails ce simulateur [26].

2.1 Architecture générale

Illustrée en Figure 3.1, son architecture permet la modélisation des différents composants d'un Cloud: Data Center, machines physiques, machines virtuelles etc... La gestion de toutes ces ressources est gérée par la couche Cloud Services, gérant l'approvisionnement des machines virtuelles, l'utilisation des CPUs, de la mémoire RAM, de la capacité de stockage et de la bande passante des machines physiques. Dans CloudSim, les taches à exécuter sont assimilées à des cloudlets qui sont alloués aux machines virtuelles (couches VM Services et User Interface Structure). La configuration de chacun des composants doit être gérée par l'utilisateur (couche Simulation Spécification) par l'intermédiaire d'un fichier de configuration de simulation. De nombreux paramètres peuvent y être spécifiés :

• Nombre de Data Center, machines physiques, machines virtuelles, taches (cloudlets)

• **Caractéristiques des Data Center :** architecture, système d'exploitation, hyperviseur, coûts de fonctionnement

• **Caractéristiques des machines physiques :** nombre de CPUs, capacité des CPUs, capacité mémoire, Capacité de stockage, bande passante

• **Caractéristiques des machines virtuelles :** capacité CPUs, nombre de CPUs, capacité mémoire, capacité de stockage

• **Caractéristiques des cloudlets :** nombre d'instructions à exécuter, taille des fichiers d'entrée et de Sortie.

2.2 Modélisation du Cloud

CloudSim est dédié à la simulation de services Cloud au niveau Infrastructure (IaaS). Pour ce faire, les simulations sont basées sur la gestion des machines physiques qui

composent le ou les Data Center. À ces machines physiques sont allouées des machines virtuelles selon des politiques de placement qui peuvent être aisément modifiées. Aussi, le cycle de vie des machines virtuelles (création, allocation, migration éventuelle et destruction) détermine le démarrage, le déroulement et la terminaison des simulations.

2.3 Modélisation des machines virtuelles et des Cloudlets

La couche de virtualisation géré l'exécution des machines virtuelles sur les machines physiques et la façon dont elles exécutent les cloudlets. En effet, CloudSim propose deux modes d'exécution : temps partagé et espace partagé pour l'exécution des machines virtuelles sur les machines physiques et également pour les cloudlets au sein des machines virtuelles. Ce qui permet donc quatre configurations d'exécution différentes. L'exécution des machines virtuelles est également gérée par l'instanciation de brokers. Un broker peut contenir une ou plusieurs machines virtuelles. La fin de son cycle de vie est déterminée par la durée de vie de chacune des machines virtuelles qui le composent. Ainsi, un broker termine son exécution seulement lorsque toutes les machines virtuelles qu'il contient ont également fini leur exécution.

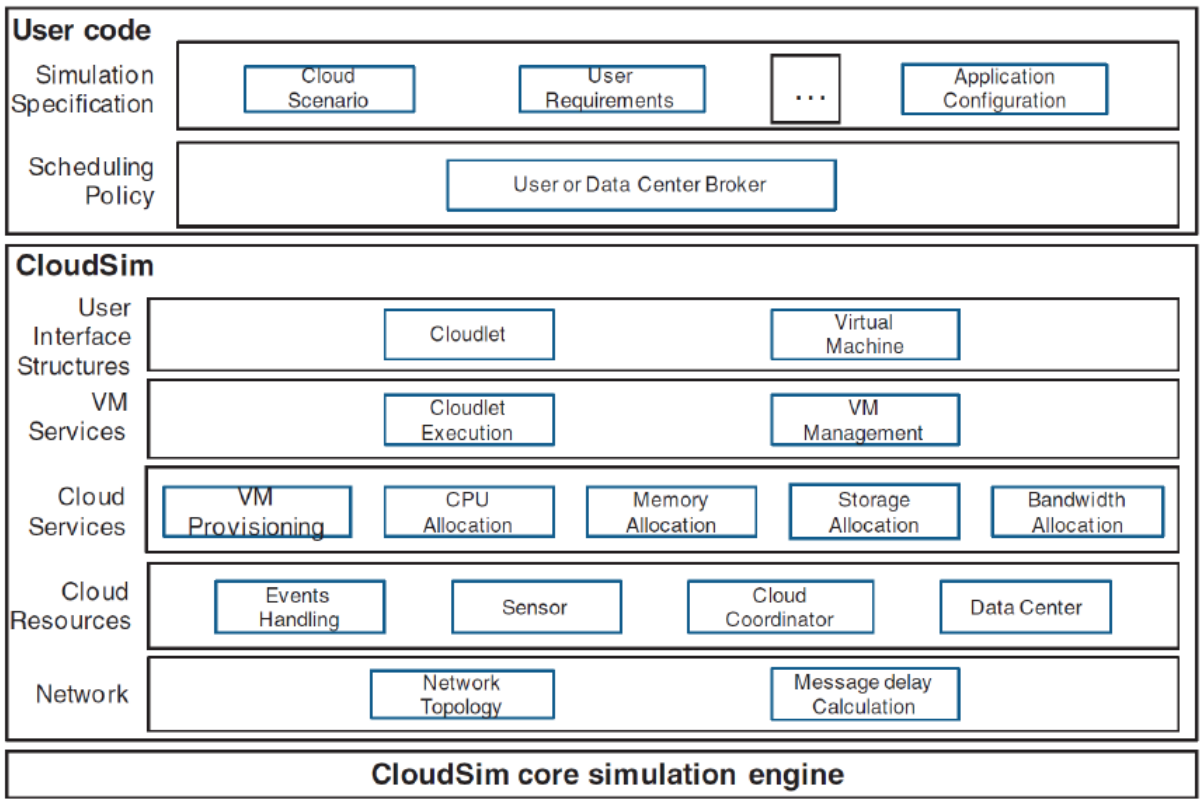

Figure 3.1 Représentation de l'architecture de CloudSim.

2.4 Politiques de placement et de migration

Des méthodes génériques de placement sont également implémentées ce qui permet à l'utilisateur d'intégrer facilement de nouvelles politiques en fonction du type d'études qu'il souhaite mener. Des algorithmes personnalisés de placement de machines virtuelles peuvent être ainsi intégrés en plus des algorithmes de base présents dans le code source du simulateur. De la même manière des classes abstraites peuvent être aisément étendues pour permettre à l'utilisateur d'implémenter ses propres algorithmes de décision de migration de machines virtuelles.

2.5 Conception et implémentation de CloudSim

Le diagramme de conception de classe pour le simulateur est représenté à la figure 3.2. Dans cette section, nous fournissons des détails plus précis liés aux classes fondamentales de CloudSim, qui sont des éléments constitutifs du simulateur.

Datacenter : Cette classe modélise les services au niveau de l'infrastructure de base (matériel, logiciel) offerts par les fournisseurs de ressources dans un environnement de Cloud Computing. Il encapsule un ensemble d'hôtes de calcul (blade servers) qui peuvent être homogènes ou hétérogènes en ce qui concerne leurs configurations de ressources (mémoire, noyaux, capacité et stockage). De plus, chaque composant Datacenter instancie un composant de provisionnement de ressources généralisé qui implémente un ensemble de stratégies pour allouer de la bande passante, de la mémoire et des périphériques de stockage.

DatacenterBroker : Cette classe modélise un courtier qui est responsable de la médiation entre les utilisateurs et les services.

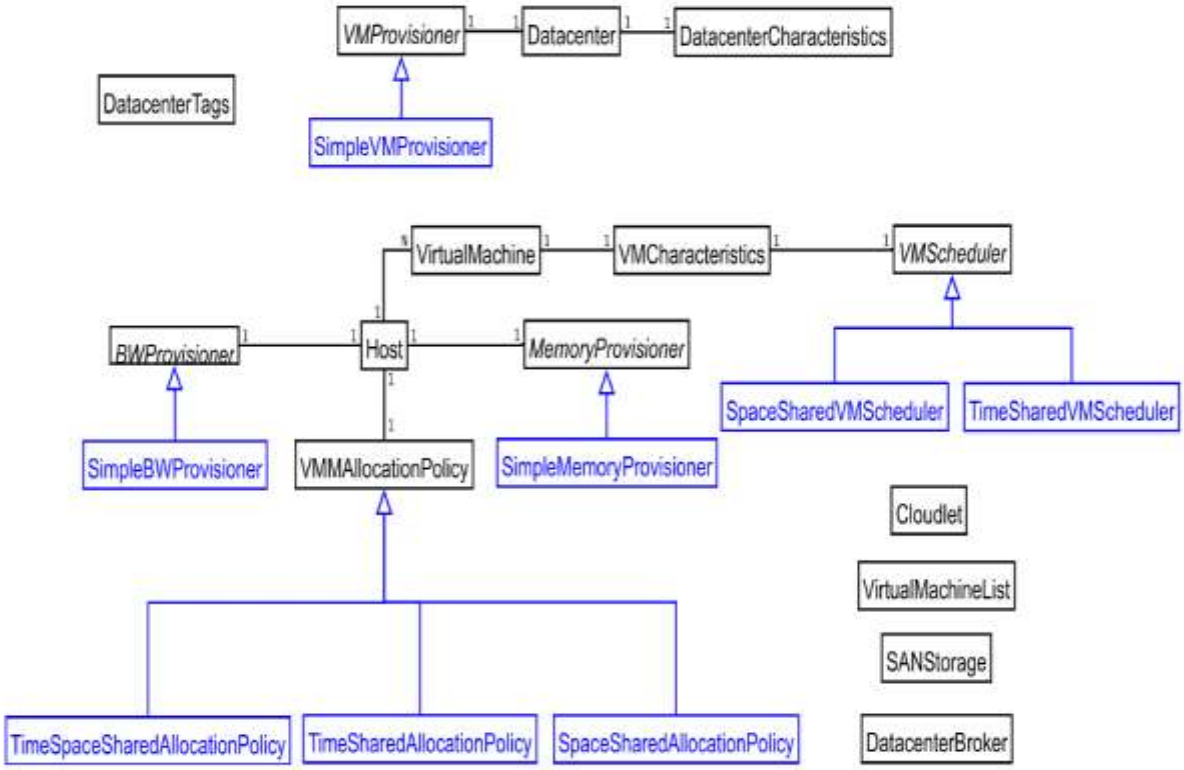

Figure 3.2 Diagramme de conception de la classe CloudSim.

Dépendant des besoins de QoS des utilisateurs et déploie des tâches de service sur Clouds. Le courtier agissant pour le compte des utilisateurs identifie les fournisseurs de services Cloud appropriés par l'entremise du Service d'information en nuage (SIC) et négocie avec eux une allocation de ressources répondant aux besoins de QoS des utilisateurs. Les chercheurs et les développeurs de systèmes doivent étendre cette classe pour effectuer des expériences avec leurs politiques personnalisées de placement d'application développées.

SANStockage : Cette classe modélise un réseau de zone de stockage qui est couramment disponible pour les centres de données basés sur le nuage pour stocker de gros morceaux de données. SANStorage implémente une interface simple qui peut être utilisée pour simuler dès la récupération de n'importe quelle quantité de données, à tout moment sous réserve de la disponibilité de la bande passante du réseau. L'accès aux fichiers d'un SAN au moment de l'exécution entraîne des retards supplémentaires pour l'exécution de l'unité de tâche, en raison du temps écoulé pour le transfert des fichiers de données requis via le réseau interne du centre de données. Machine virtuelle. Cette classe modélise une instance d'une machine virtuelle dont la gestion pendant son cycle de vie est la responsabilité du composant hôte. Comme discuté précédemment, un hôte peut simultanément instancier plusieurs machines virtuelles et

Allouer des noyaux basés sur des politiques prédéfinies de partage du processeur (espace partagé, temps partagé). Chaque composant VM a accès à un composant qui stocke les caractéristiques liées à une machine virtuelle, telles que la mémoire, le processeur, le stockage et la politique de planification interne de la VM, qui est étendue à partir du composant abstrait appelé VM Scheduling.

VMScheduling : Petit nuage, Cette classe modélise les services d'application basés sur le Cloud (diffusion de contenu, réseautage social, flux de travail d'entreprise), qui sont communément déployés dans les centres de données. CloudSim représente la complexité d'une application en termes de ses exigences de calcul. Chaque composant d'application possède une longueur d'instruction pré-affectée (héritée du composant GridSim de GridSim) et la quantité de transfert de données (pré et post-récupération) qui doit être entreprise pour héberger avec succès l'application.

BWProvisioner : Il s'agit d'une classe abstraite qui modélise la politique de fourniture de bande passante pour les machines virtuelles déployées sur un composant hôte. La fonction de ce composant est d'entreprendre l'allocation des largeurs de bande du réseau à l'ensemble des VM concurrentes déployées à travers le centre de données. Les développeurs et chercheurs du système Cloud peuvent étendre cette classe à leurs propres politiques (priorité, QoS) pour refléter les besoins de leurs applications.

MemoryProvisioner : Il s'agit d'une classe abstraite qui représente la politique de provisionnement pour allouer de la mémoire à VMs. Ce composant modélise les stratégies pour allouer des espaces de mémoire physique aux machines virtuelles concurrentes. L'exécution et le déploiement de VM sur un hôte n'est possible que si le composant Memory Provisioner détermine que l'hôte dispose de la quantité de mémoire libre requise pour le nouveau déploiement de VM.

VMProvisioner : Cette classe abstraite représente la politique de provisionnement utilisée par un moniteur VM pour attribuer des VM aux hôtes. La fonctionnalité principale du VMProvisioner consiste à sélectionner l'hôte disponible dans un centre de données, qui répond aux exigences de mémoire, de stockage et de disponibilité pour un déploiement de VM. L'implémentation Simple VMProvisioner par défaut fournie avec le package CloudSim alloue les machines virtuelles au premier hôte disponible qui répond aux exigences précitées. Les hôtes sont considérés pour la cartographie dans un ordre séquentiel. Cependant, des politiques plus compliquées peuvent être facilement mises en œuvre au sein de cette

composante pour obtenir des allocations optimisées, par exemple, la sélection d'hôtes en fonction de leur capacité à répondre aux exigences de QoS telles que le temps de réponse et le budget.

VMMAllocation Policy: Il s'agit d'une classe abstraite implémentée par un composant Host qui modélise les stratégies (espace-partagées, à temps partagé) requises pour attribuer la puissance de traitement aux machines virtuelles. Les fonctionnalités de cette classe peuvent facilement être remplacées pour tenir compte des politiques de partage de processeurs spécifiques à l'application.

2.6 Intégration de l'algorithme Génétique

Le simulateur CloudSim adopté pour cette thèse, contenait déjà plusieurs outils indispensables pour pouvoir effectuer des simulations de répartition des charges. Seule l'implémentation de l'algorithme Génétique ne faisait pas partie des outils disponibles pour étudier de façon plus précise notre problème. C'est pourquoi l'implémentation de cet outil a constitué une étape importante de ces travaux pour ainsi pouvoir exécuter des simulations précises. Celle-ci a nécessité tout d'abord une démonstration de l'algorithme en l'applique à notre problème, mais également une étape de compréhension du code source du simulateur afin de pouvoir intégrer cet algorithme.

3. Qu'est ce que NetBeans?

NetBeans est un projet open source ayant un succès et une base d'utilisateur très large, une communauté en croissance constante, et près 100 partenaires mondiaux et des centaines de milliers d'utilisateur à travers le monde. Sun Microsystems a fondé le projet open source NetBeans en Juin 2000 et continue d'être le sponsor principal du projet. Aujourd'hui, deux projets existent: L'EDI NetBeans et la Plateforme NetBeans. L'EDI NetBeans est un environnement de développement un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java - mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'EDI NetBeans. L'EDI NetBeans est un produit gratuit, sans aucune restriction quant à son usage. Également disponible, la plateforme NetBeans; une fondation modulable et extensible utilisée comme brique logicielle pour la création d'applications bureautiques. Les partenaires privilégiés fournissent des modules à valeurs rajoutées qui s'intègrent facilement à la plateforme et peuvent être utilisés pour développer ses propres outils et solutions. Les deux

produits sont open source et gratuits pour un usage commercial et non-commercial. Le code source est disponible pour réutilisation sous la Common Development and Distribution License (CDDL).

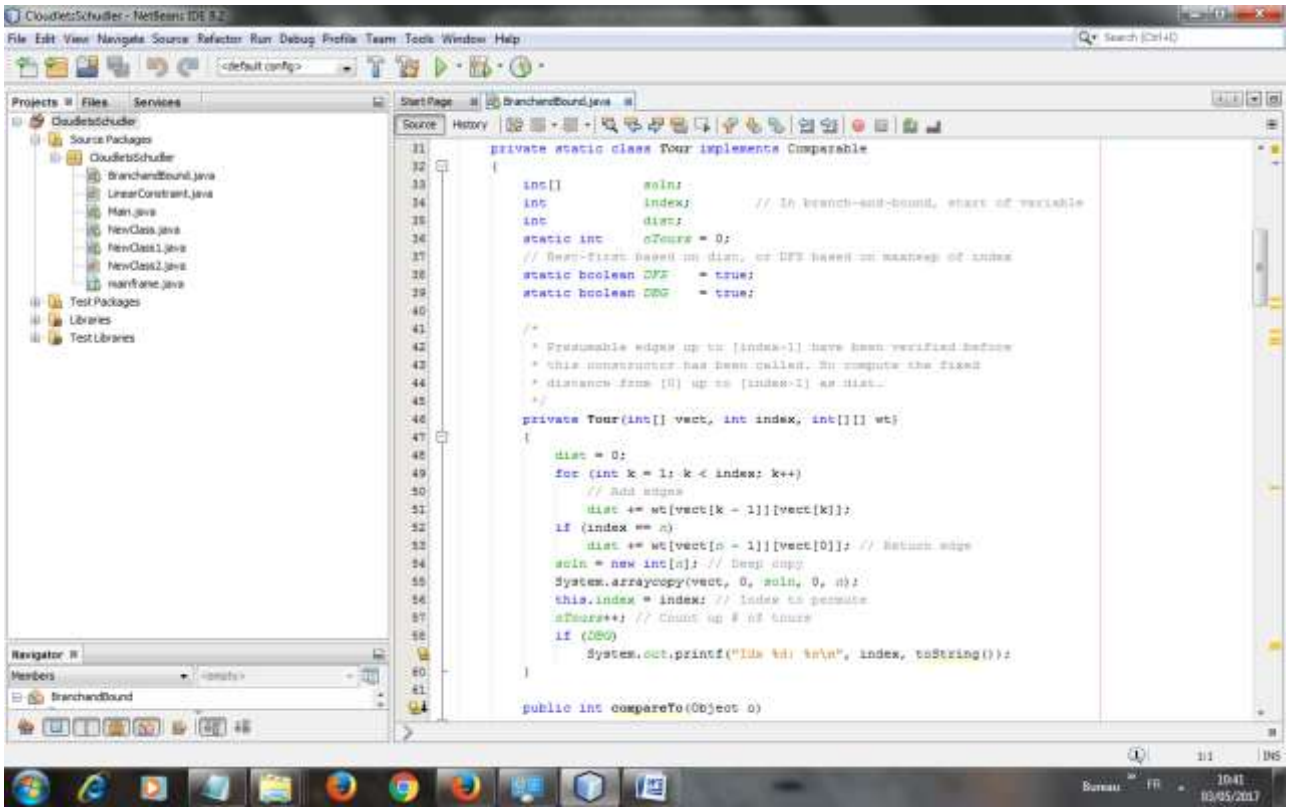

Figure 3.3 NetBeans IDE.

4. Réalisation et résultat d'exécution du projet

4.1 Installation et ajout des package cloudsim 4.0

L'image suivante présente les class important du cloudsim utilisé dans notre travail.

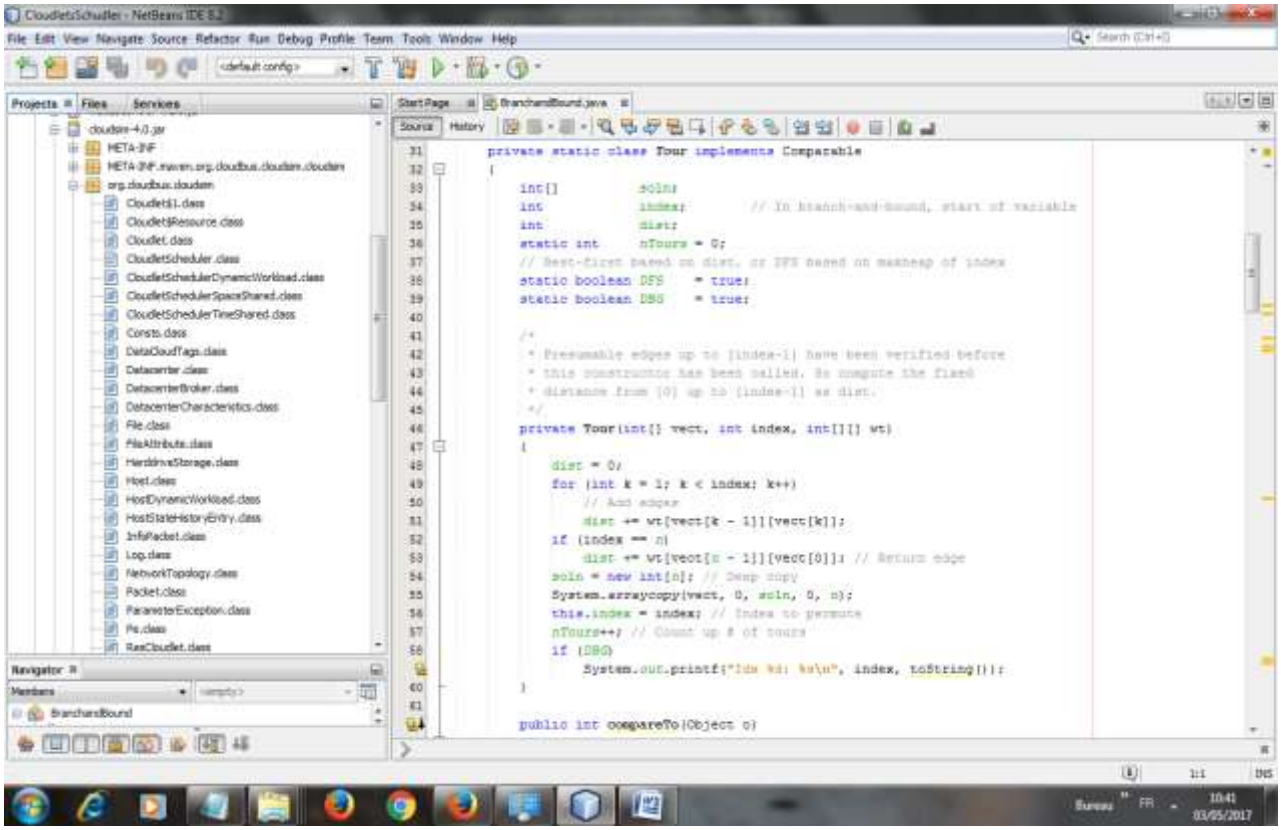

Figure 3.4 Le package cloudsim.

4.2 Fenêtre principale de l'application

Comme démontre la fenêtre nous avons deux panneaux une pour la création des VMs avec ses caractéristiques et l'autre pour la création des cloudlets (srvices) , le tous nous donne la configuration de notre cloud.

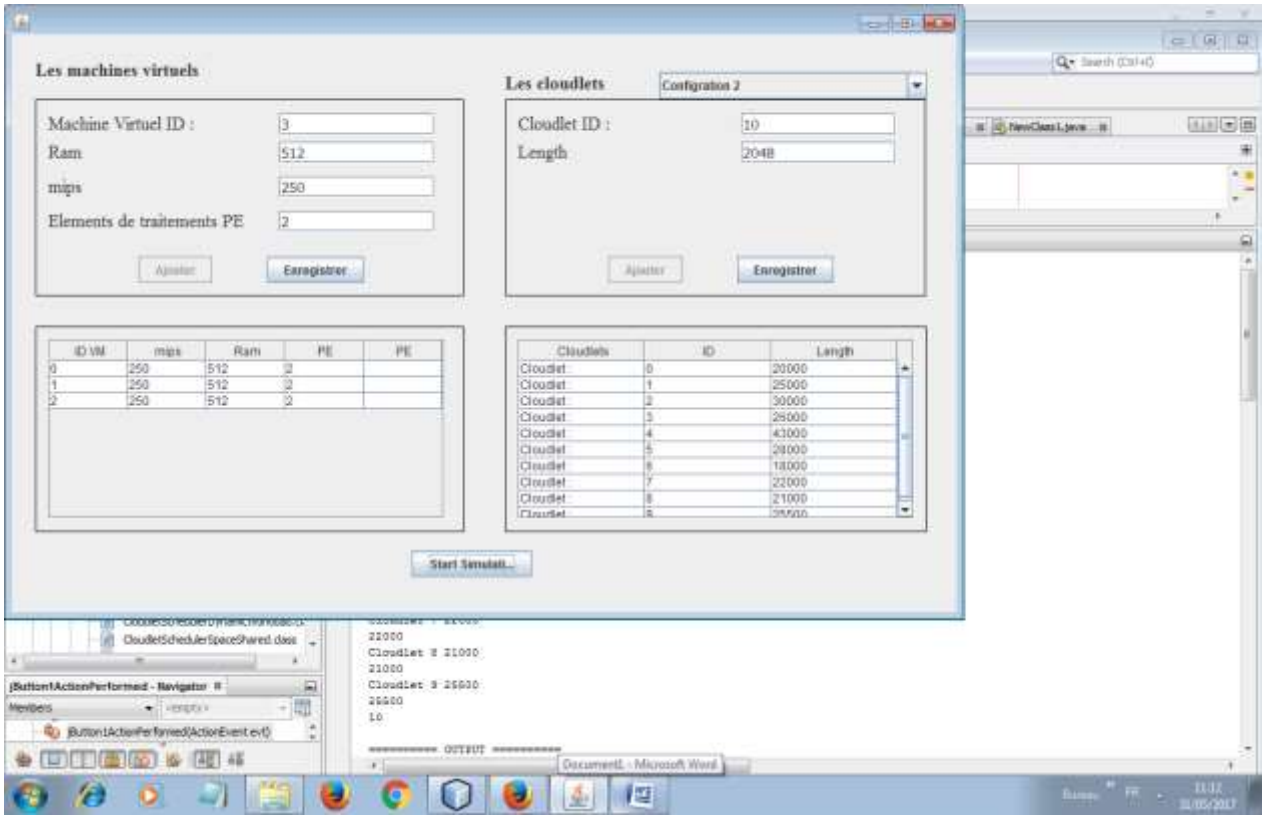

Figure 3.5 Fenêtre principale de l'application.

Une fois la configuration est faite, on lance la simulation à travers le Botton « start simulation ». Les résultats de simulation présentée par les fenêtres suivantes dont on remarque la solution optimale donné par notre application « best solution » puis l'affectation des cloudlets aux VMs après la création de toutes les entités nécessaires (data center, broker …).

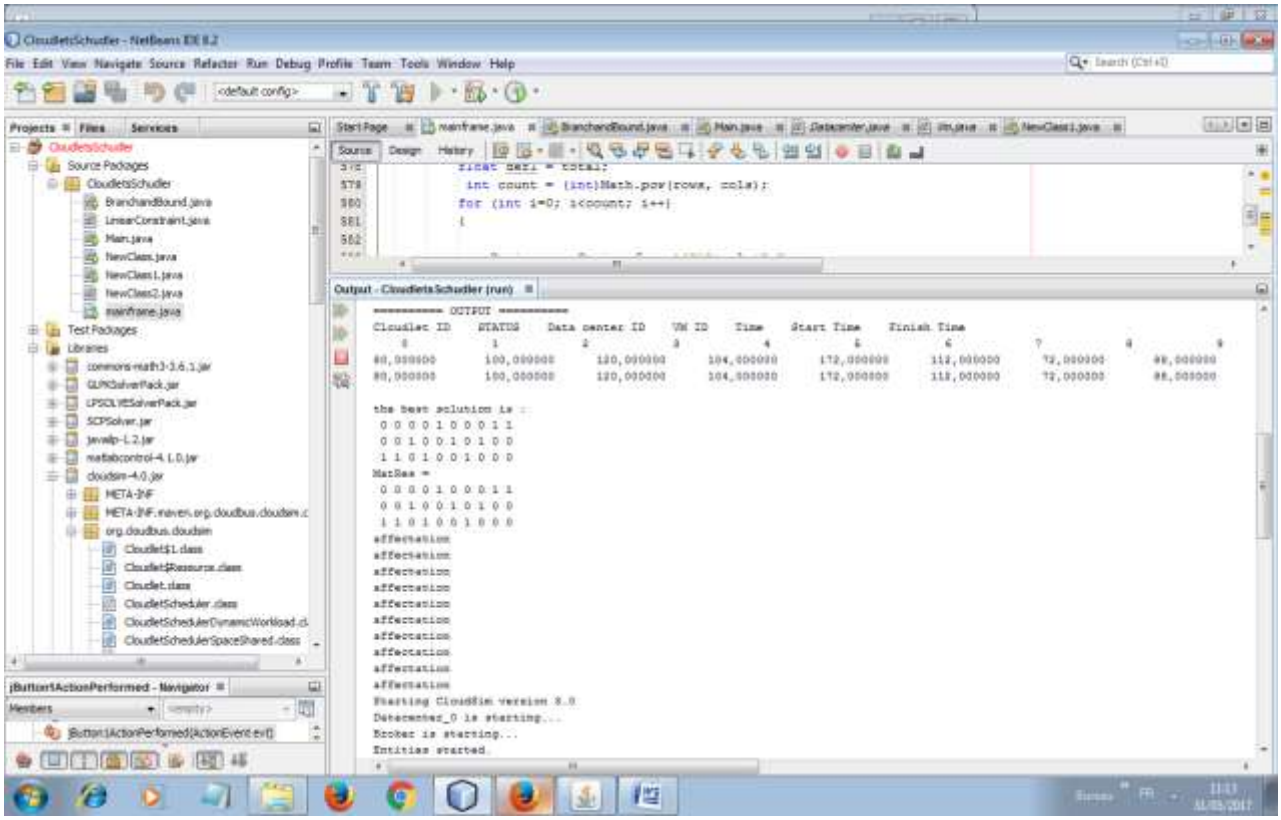

Figure 3.6 La solution optimale.

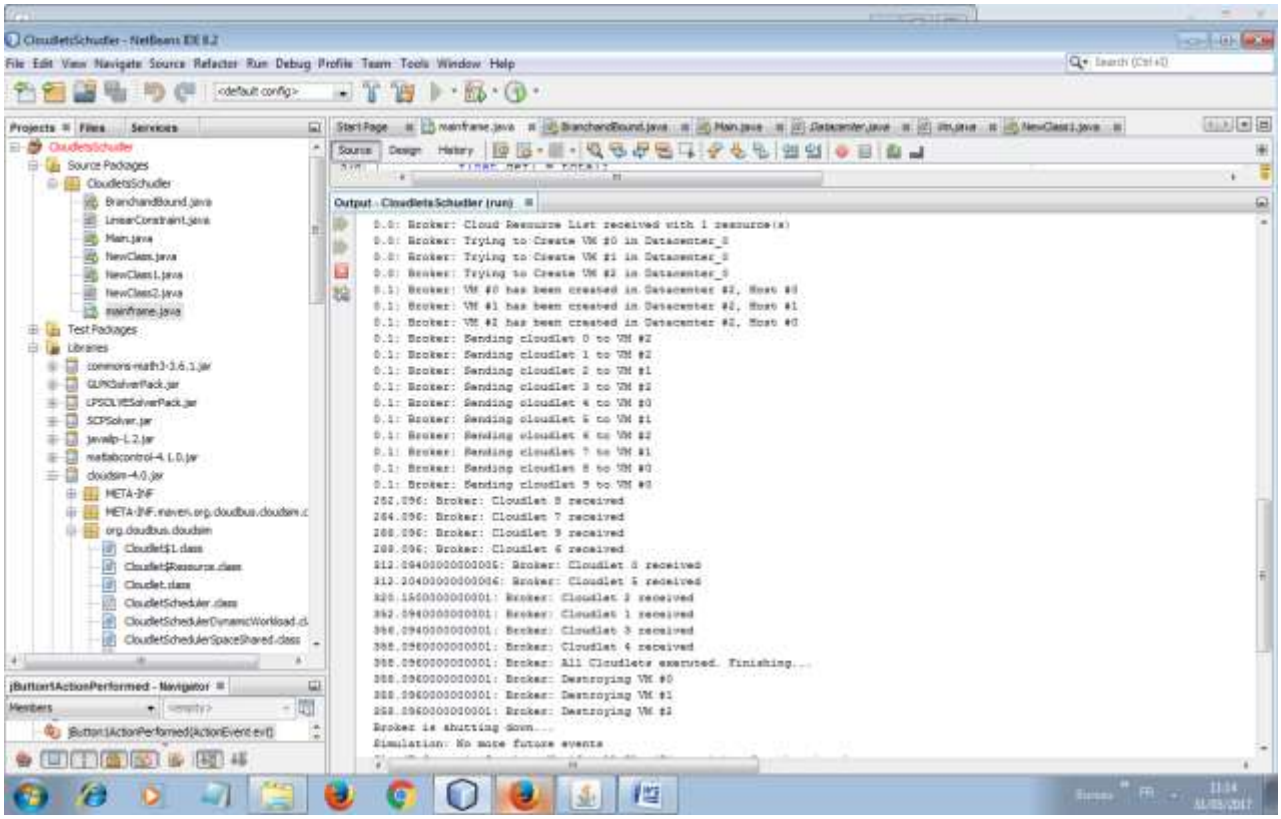

Figure 3.7 Création des entités du *cloud.*

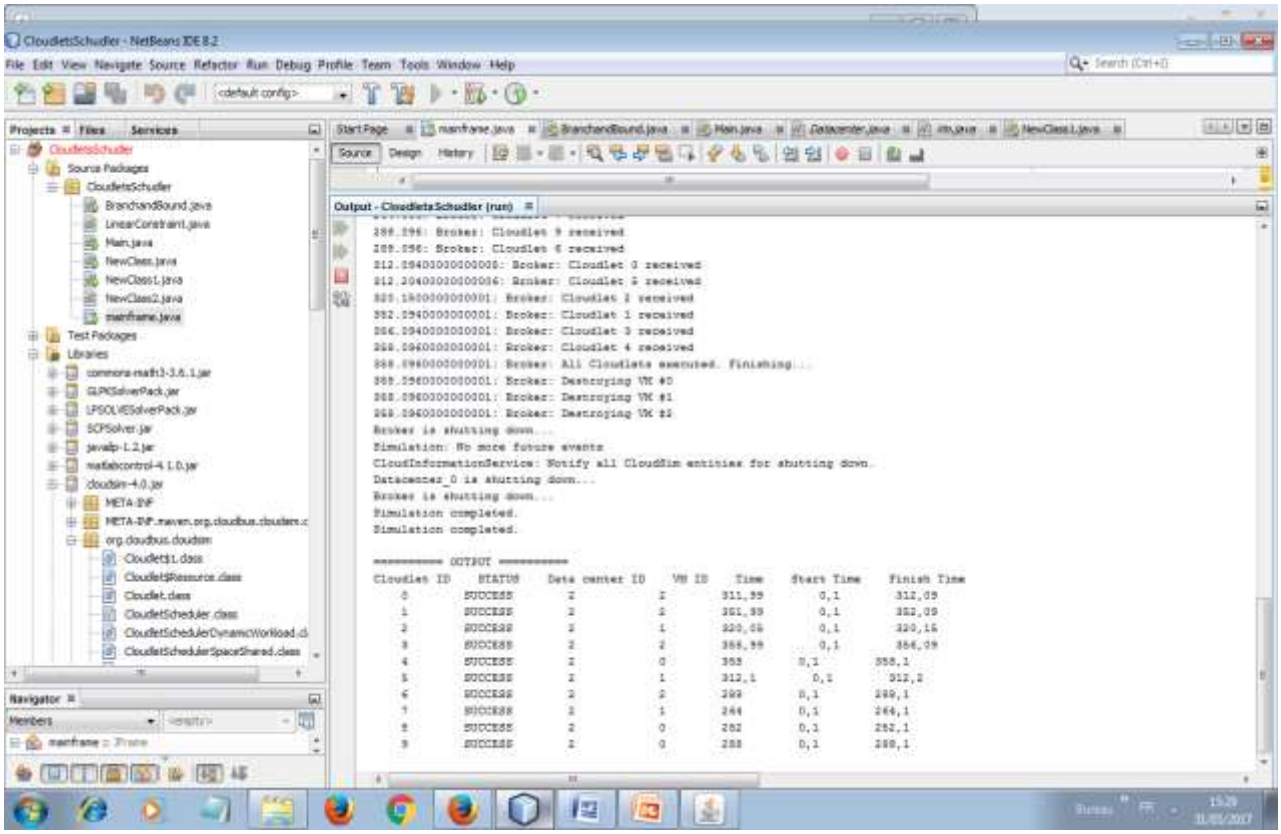

Figure 3.8 Statistique d'exécutions des *cloudlets.*

4.3 Analyse des résultats d'exécution

Les résultats de simulation obtenus par notre solution démontre clairement une allocation efficace en utilisant le Brach and Bound qui permet une optimisation considérable du temps d'exécution de l'ensemble des cloudlet au niveau du cloud par rapport au résultat donnés par la politique d'allocation standard utilisé dans le cloudsim (FIFO), les deux images suivantes présente les résultats obtenus.

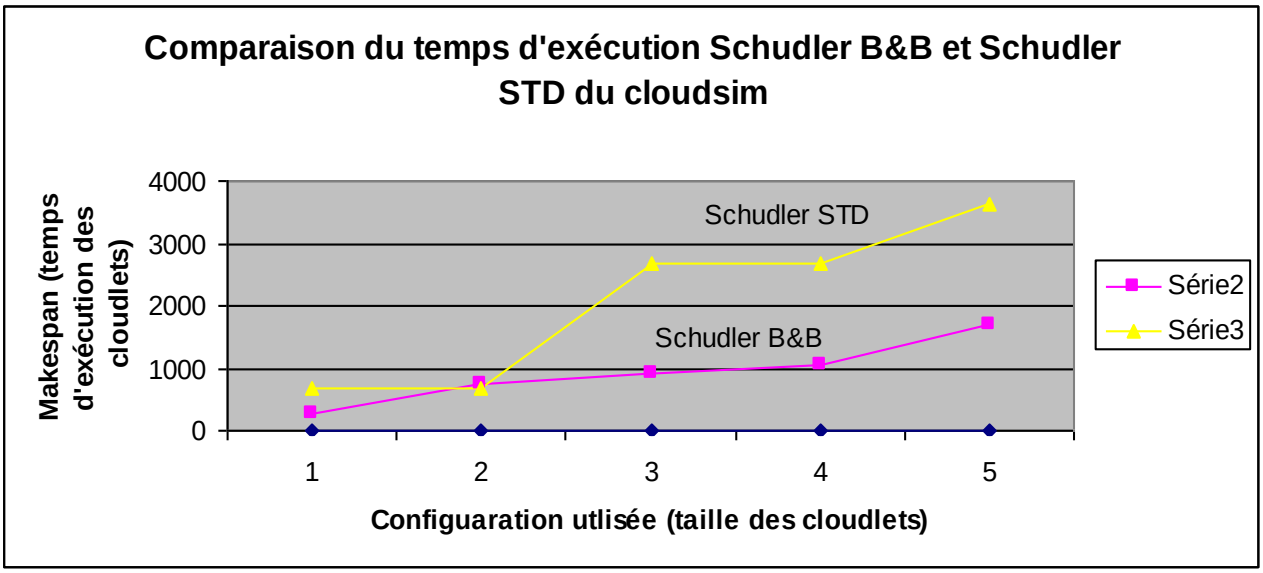

Figure 3.9 Résultat d'exécution (mv de différent configuration).

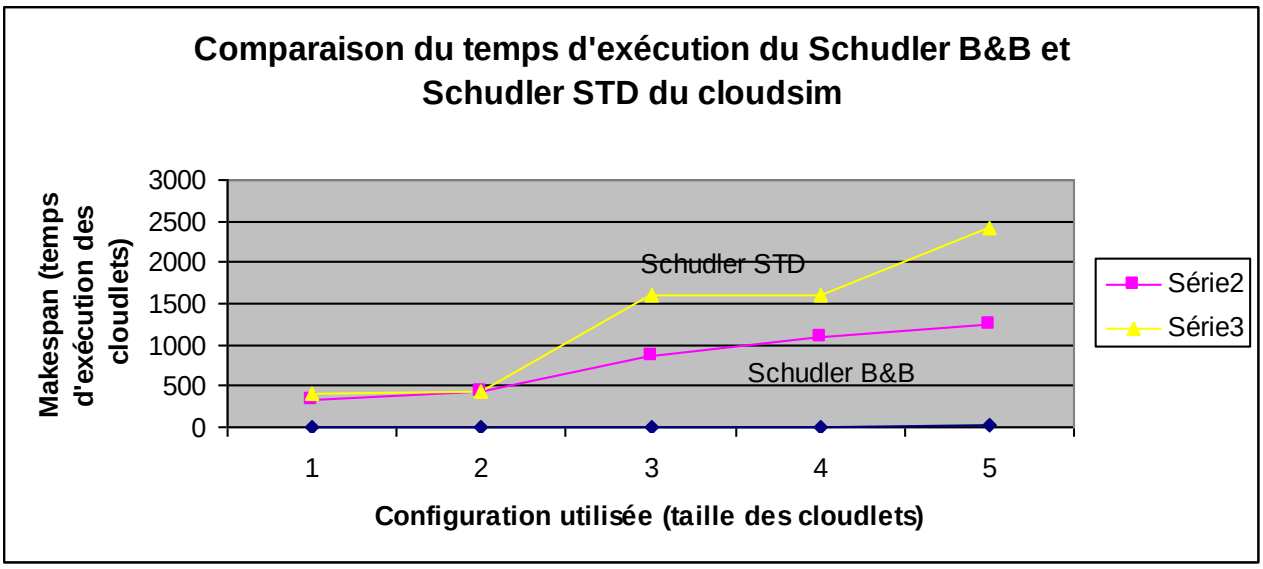

Figure 3.10 Résultat d'exécution (mv de même configuration).

5. Conclusion

Dans ce chapitre, nous avons proposé notre solution sous forme d'application java qui présente une simulation d'un Cloud en intégration la bibliothèque CloudSim, dont on intéresse à la partition scalable de calcul distribué sur notre Cloud par l'implantation du l'algorithme Génétique, qui donne des résultats satisfaisants sans contraintes par rapport au nombre des machines virtuels ou cloudlet (services). La bonne compréhension du problème nous a permet de résoudre le problème de façon simple et efficace.

Conclusion générale

Coclusion générale

L’un des problèmes fondamentaux de cloud computing est l’ordonnancement des tâches d’une façon optimale. Un ordonnancement consiste à déterminer l’ordre dans lequel les différentes tâches des processus doivent s’exécuter et les ressources que chacune d’elles doit utiliser. En outre, les utilisateurs sont confrontés au problème du choix des meilleures ressources (fournisseurs) à utiliser.

Le problème d’ordonnancement reste ouvert dans le domaine du cloud computing. Enoutre, il n’existe aucun algorithme d’ordonnancement traitant le problème de la tâcheconcurrente aux ressources disponibles.

Dans notre étude, nous avons discuté le problème d’ordonnancement de taches dans le cloud computing. Ce problème est considéré comme étant un problème d'optimisation multi-objectif en raison de la nature conflictuelle des métriques de QoS (Quality of Service) à prendre en compte.

Dans ce cadre de cette étude, nous avons proposé une approche évolutive de l'optimisation multiobjectifs pour l’ordonnancement de tâches dans l’environnement cloud. Nous proposant une solution sous forme d’une simulation d’un cloud en utilisant cloudsim.

Bibliographie

[1] Alain Berro, 18 décembre 2001, Optimisation multi objectif et stratégies d'évolution enenvironnement dynamique, Université de Toulouse I

[2] Alain Berro, 4 novembre 2008, Algorithme évolutionnaire pour l'optimisation multi objectif,Université de Toulouse I.

[3] Amira Gherboudj, 2013, Méthodes de résolution de problèmes difficiles académiques, Université de Constantine2.

[4] "Citrix Hypervisor | Open Source Server Virtualization." [Online]. Available: https://xenserver.org/. [Accessed: 30-May-2019].

[5] Claire Monin, 7 Janvier 2014, Optimisation multiobjectif de l'allocation Stratégique par Un algorithme génétique, Université Claude Bernard – Lyon 1.

[6] "Data Center Architecture Overview - Cisco." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCInfra_1.html. [Accessed: 11-May-2019].

[7] Dominique Francisci, mars 2002, Algorithmes évolutionnaires et optimisation multi-objectifs en data mining, Rapport de recherche I3S/RR-2002-12-FR, Laboratoire I3S.

[8] D. S. Linthicum, Praise for Cloud Computing and SOA Convergence in Your Enterprise. 2010.

[9] ElghazaliTalbi , Méthodes d'Optimisation Avancée.

[10] ELMIR Younes, 2017, Optimisation multi objectif par les algorithmes génétiques et approche Pareto de paramètres d'un contrôleur PID, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf.

[11] "ESXi | Bare Metal Hypervisor | VMware." [Online]. Available: https://www.vmware.com/products/esxi-and-esx.html. [Accessed: 30-May-2019]

[12] G. Reese, Cloud application architectures. O'Reilly Media, Inc, 2009.

[13] H. Geng, Data center handbook.

[14] Imed OTHMANI, 20 Mai 1998, Optimisation multicritère fondements et concepts, Université Joseph Fourier – GRENOBLE.

[15] J. Andersson ; A Survey for Multiobjective Optimization in engineeringDesign; Rapport technique, LiTH-IKP-R-1097; 200

[16] John R. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press Cambridge, MA; USA 1992; page 86

[17] J. Régnier, B. Sareni, X. Roboam, System Optimization by Multiobjective Genetic Algorithms and Analysis of the Coupling between Variables Constraints and Objectives, Laboratoire d'Electrotechnique et d'Electronique Industrielle Toulouse.

[18] "KVM." [Online]. Available: https://www.linux-kvm.org/page/Main_Page. [Accessed: 30-May-2019].

[19] Labed Kaouthar, Expérimentation des algorithmes génétiques multi objectifs dans un processus décisionnel multicritère en aménagement du territoire, Université d'Oran Es-senia

[20] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds," ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, p. 50, 2008.

[21] Merdjaoui Brahim, 14 october 2006, Optimisation multi-objectif par algorithms génétiques et approche Pareto des paramètre d'usinage sous contraintes des limitations de production, Université M'hamedBougaraBoumerdes.

[22] MOUELHI Ouael, 17 mars 2010, Contribution à l'optimisation multiobjectif en conception multidisciplinaire,L'institut National Des Sciences Appliquées De Lyon.

[23] Nicolas Jozefowiez, 2013, Optimisation combinatoire multi-objectif : des méthodes aux problèmes de la Terre à (presque) la Lune, Institut National Polytechnique de Toulouse.

[24] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," 2011.

[25] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," J Internet Serv Appl, vol. 1, pp. 7–18, 2010.

[26] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar AF De Rose et Rajkumar Buyya. "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". Dans : Software : Practice and Experience 41.1 (2011) ;

[27] Saadi Leila, 08/07/2007, Optimisation Multi Objectifs par Programmation Génétique, Université de BATNA.

[28] Vincent Barichard-Jin-Kao Hao, 2003, Une approche hybride Pour l'optimisation multiobjectif sous contraintes, Faculté des sciences-Université d'Angers.

[29] Yann Collette, Patrick Siarry, 02/10/2002, Optimisation multi objectif, Editeur : Eyrolles,Collection : Algorithmes.