REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux & Télécommunications

Par :

BEKKOUCHE Djamel-Eddine & BENEDINE Farouk

Sur le thème

Développement de(s) solution(s) pour l'allocation de ressources dans l'infonuagique.

Soutenu publiquement le 26 / 06 / 2018 à Tiaret devant le jury composé de :

Mr GAFOUR Yacine	Grade MAA	Université Ibn Khaldoun-Tiaret	Président
Mr HATTAB Noureddine	Grade MAA	Université Ibn Khaldoun-Tiaret	Encadreur
Mr DJAAFRI Laouni	Grade MAA	Université Ibn Khaldoun-Tiaret	Examinateur

Dédicaces

À nos familles et nos parents …

À nos frères et nos sœurs …

À nos ami(e)s et nos collègues …

À tous ceux qui nous ont encouragé et aidé …

Remerciements

En préambule à ce mémoire nous remerciant ALLAH qui nous aide et nous donne la patience et le courage durant cette année.

Ce mémoire n'aurait pas été possible sans l'intervention, consciente, d'un grand nombre de personnes. Nous souhaitons ici les en remercier.

Nous tenons d'abord à remercier très chaleureusement Mr. HATTAB qui nous a permis de bénéficier de son encadrement. Les conseils qu'il nous a prodigué, la patience, la confiance qu'il nous a témoignés ont été déterminants dans la réalisation de notre travail de recherche.

Nos vifs remerciements vont également au notre président de jury Mr. GAFOUR ainsi qu'au Mr. DJAAFRI en tant qu'examinateur pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre modeste travail et de l'enrichir par leurs propositions.

Nos remerciements s'étendent également à tous nos enseignants durant les années des études.

On remercie également tous nos amis et amies de la promotion pour leur soutien et les moments agréables que nous avons passé ensemble.

Merci à tous et à toutes.

Table des matières

Liste des figures

CHAPITRE I : APERÇU GÉNÉRAL SUR L'INFONUAGIQUE

CHAPITRE II : ÉTAT DE L'ART SUR ALLOCATION DES RESSOURCES

CHAPITRE III: RÉALISATION & EXPÉRIMENTATION

Liste des tableaux

CHAPITRE I : APERÇU GÉNÉRAL SUR L'INFONUAGIQUE

CHAPITRE II : ÉTAT DE L'ART SUR ALLOCATION DES RESSOURCES

CHAPITRE III: RÉALISATION & EXPÉRIMENTATION

Liste d’abréviations

NIST = National Institute of Standards and Technology. | **SaaS** = Software as a Server.

PaaS = Platform as a Server. | **IaaS** = Infrastructure as a Server. | **GHZ** = Giga Hertz.

API = Application Programming Interface. | **CSIM** = Cloud SIM (SIM = Simulator).

NS2 = Natural Selection 2. | **DVFS** = Dynamic Voltage and Frequency Scaling.

DNS = Domain Name Server. | **MDCSim** = A multi-tier data center simulation.

TCP/IP = Transmission Control Protocol / Internet Protocol | **SJF** = Shortest Job First

UDP = User Datagram Protocol | **GUI** = Graphic User Interface | **ID** = Identifying

HPC = high performance computer | **CPU** = Central Processing Unit | **RR** = Round Robin

RAM = Random Access Memory **SLA** = Service Level Agreement | **SJN** = Short Job Next

FCFS = First Come First Serve | **SRTF** = Shortest Remaining Time First | **GB** = Giga Byte

RRS = Round Robin Sheduling **PSA** = Priority Sheduling Algorithm

Min-Min = Minimum Minimum | **Max-Min** = Maximum Minimum

RASA = Resource Aware Scheduling Algorithm | **GA** = Genetic Algorithm

LCA = League Championship Algorithm | **PSO** = Particle Swarm Optimization

ACO = Ant Colony Optimization | **VM** = Virtual Machin | **JIT** = Just In Time Compiler

JVM = Java Virtual Machin | **JDK** = Java Development Kit

EDI = Environnement de Développement Intégré | **QOS** = Quality Of Service

CDDL = Common Development and Distribution | **CIS** = Cloud Information Services

MB = Mega Byte | **BW** = Band width | **MIPS** = million instruction par second.

Résumé

L’**informatique en nuage** "l’**infonuagique"**, ou encore le ***cloud computing,*** un nouveau paradigme figuré comme une conséquence directe de la virtualisation des environnements distribués classique, un paradigme qui consiste à exploiter la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement Internet. La gestion des ressources est l'un des problèmes ouverts dans ce paradigme effervescent. Il s'agit de trouver des solutions efficaces en terme de consommation de ressources (calcule, stockage, bande passante,…etc.), intérieurement l’allocation des ressources et la planification des taches deviennent un sujet capital et décisif pour l'avenir et la réputation de toute entreprise adoptons ce paradigme. L’objectif ultime de ce travail est de faire une réalisation et une implémentation et par la suite une comparaison entre des solutions disponibles et reconnues dans ce sujet avec une focalisation sur la ressource de calcule (CPU) afin de déduire la meilleur solution. Finalement et pour raffiner, valider notre étude expérimentalement, nous avons réalisé une application sur l'environnement "CloudSim".

Mots clés : Virtualisation, Infonuagique, Allocation, Planification, Ressources, CloudSim.

Abstract

Cloud computing, a new paradigm appears as a direct consequence of the virtualization of distributed environments classic, a paradigm that is to exploit the computing power or storage of remote computer servers via a network, usually the Internet. Resource management is one of the open issues in this effervescent paradigm. It's about finding efficient solutions in terms of resource consumption (computing, storage, bandwidth, etc.), internally the allocation of resources and the planning of tasks becomes a crucial and decisive topic for the future and the future. Reputations of any companies adopt this paradigm. The ultimate goal of this work is to make an implementation and an implementation and then a comparison between available and recognized solutions in this subject with a focus on the resource of calculation (CPU) in order to deduce the best solution. Finally is to refine, validate study note experimentally, we realized an application on the environment "CloudSim".

Keywords: Virtualization, Cloud, Allocation, Planning, Resources, CloudSim.

ملخص

الحوسبة السحابية، نموذج جديد هو نتيجة مباشرة للافتراضية التقليدية الموزعة ، وهو نموذج يستغل الطاقة الحسابية أو تخزين خوادم الحوسبة البعيدة. عبر شبكة ، عادة الإنترنت. إدارة الموارد هي واحدة من القضايا المفتوحة في هذا النموذج فوارة. إنها تتعلق بإيجاد حلول فعالة من حيث استهلاك الموارد (الحوسبة ، التخزين ، عرض النطاق الترددي ، ... وتخطيط المهام موضوع حاسم للمستقبل، لأنها تعتبر سمعة أي شركة تعتمد الخ)، وفي الداخل، يصبح تخصيص الموارد إن الهدف النهائي من هذا العمل يتمثل في تنفيذ وتطبيق ثم مقارنة بين الحلول المتاحة والمعترف بها في هذا هذا النموذج. وأخيرًا وللتنقيح الموضوع مع التركيز على مورد الحساب (وحدة المعالجة المركزية) من أجل استخلاص الحل الأفضل. والتحقق من دراستنا تجريبيًا، فقد أدركنا تطبيقًا على البيئة "كلاود سيم ".

كلمات مفتاحية : المحاكاة الافتراضية ، الغيمة ، تخصيص ، التخطيط ، الموارد ، كلاود سيم

Introduction Générale

1. Contexte & Motivation

L'infonuagique est une technologie à la mode, consiste à fournir un service [1] plutôt qu'un produit. Des services comme le calcul, le logiciel, l'accès aux données et le stockage sont fournis à son utilisateur sans aucune connaissance préalable sur l'emplacement physique et la configuration du serveur fournissant ces services. Les Data Centres qui compose l'infonuagique possédant plusieurs éléments tels que les serveurs, les équipements réseaux, les systèmes de refroidissement consomment de grandes quantité d'énergie pour fournir des services efficaces et fiables à leurs utilisateurs [2].

Cette forte consommation d'énergie a augmenté les couts d'exploitation pour les fournisseurs ainsi que pour les utilisateurs de services. En outre, une grande quantité de dioxyde de carbone est émise, cela résulte en augmentation de réchauffement de la planète dans un avenir proche. Ce problème de consommation d'énergie par les Data Center n'est pas causé seulement par la grande quantité des ressources physiques, mais réside plutôt dans l'inefficacité d'utilisation des ressources physiques. De ce fait, de nombreuses entreprises sont séduites d'utiliser des ressources disponibles de manière efficace, presque sans limite, tout en minimisant l'énergie consommée des applications et le budget induit par l'utilisation de ces ressources.

2. Problématique:

La gestion des ressources est l'un des problèmes ouverts dans ce paradigme effervescent. Il s'agit de trouver des solutions efficaces en terme de consommation de ressources (calcule, stockage, bande passante,…etc.), intérieurement l'allocation des ressources et la planification des taches devienne un sujet capital et décisif pour l'avenir et la réputation de toute entreprises adoptons ce paradigme.

3. Objectif :

l'objectif de ce travail présenté dans œ mémoire est de réaliser un ensemble des solutions dans une infrastructure infonuagique afin de Planifier les taches en s'appuyant sur l'une des ressources les plus importantes dans ce paradigme qu'il s'agit du ressource de calcule 'CPU', et par conséquent minimisé l'énergie dépensée par le CPU. L'objectif ultime est de précisé la meilleur solution dans ce cadre.

4 .Organisation du manuscrit

Nous avons structuré notre mémoire selon le plan décrit ci-dessous :

Chapitre I : ce premier chapitre constitue une description des principaux concepts, technologies et caractéristiques du paradigme Infonuagique.

Chapitre II : réservé aux concepts liés à notre sujet qu'il s'agit de l'allocation des ressources et la planification des taches dans l'infonuagique, avec un état de l'art non exhaustive sur les solutions et les mécanismes attaquant ce sujet.

Chapitre III : ce chapitre s'intéressera à l'implémentation, la réalisation et la comparaison entre un ensemble de solutions/mécanismes sélectionnée, afin de déduire la meilleur solution envisageable en terme d'énergie dépensée par la ressource de calcule "CPU".

Ce mémoire se termine par une conclusion générale et par un ensemble des perspectives.

Chapitre I : Aperçu générale sur l'infonuagique

1. Introduction

L'informatique dans le nuage « infonuagique » ou encore plus connue sous sa forme anglo-saxonne : « Cloud Computing » , est un nouveau paradigme orienté service, définit comme un system informatiques basé sur le partage/portage intelligent de ressources, massivement virtualisé . Ce chapitre consiste une synthèse, qui portera sur des définitions et des concepts fondamentaux relatifs à l'infonuagique.

2. Les Concepts de l'infonuagique

Selon l'Institut national des normes et de la technologie, l'infonuagique est un modèle pour permettre un accès pratique à la demande du réseau à un ensemble partagé de ressources informatiques configurables (par exemple, les réseaux, les serveurs, le stockage, les applications et les services) qui peuvent être provisionnés rapidement et libérés avec un effort de gestion minimale ou par l'interaction de fournisseur de services (Figure 1.1). Ce modèle favorise l'accessibilité et est composé de cinq caractéristiques essentielles [6]:

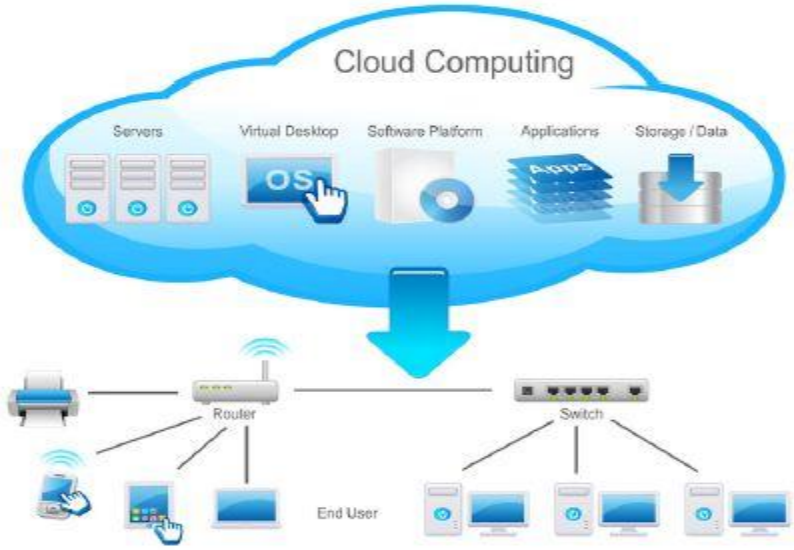

Figure 1.1 L'environnement Infonuagique.

1. La demande libre des services
2. Un accès en diffusion via le réseau
3. La mise en commun des ressources
4. L'élasticité rapide
5. Un service mesuré

Trois modèles de services (SaaS, PaaS et IaaS) et quatre modèles de déploiement (privé, public, communautaire et hybride). Les technologies clés comprennent :

1. Des réseaux rapides,
2. Des ordinateurs bon marché,
3. La virtualisation pour du matériel de base.

Les principaux obstacles à la plus large adoption du Cloud sont :

La sécurité, l'interopérabilité et la portabilité.

Nous récapitulons en termes simples et courts, l'infonuagique est une grande puissance évolutive et personnalisée de calcul disponible par loyer / par heure et accessible à distance. Il peut aider à faire plus de calcul à une fraction de cout.

2.1. La Virtualisation

La virtualisation et l'infonuagique vont de pair pour fournir différents types de services.

Le principe de la virtualisation est de diviser une pièce matérielle physique en plusieurs segments. Ces segments fonctionnent indépendamment bien qu'ils partagent la même ressource physique. De l'autre côté, la virtualisation peut être utilisée pour fusionner une ou plusieurs ressources physiques et leur donner l'apparence d'une seule ressource.

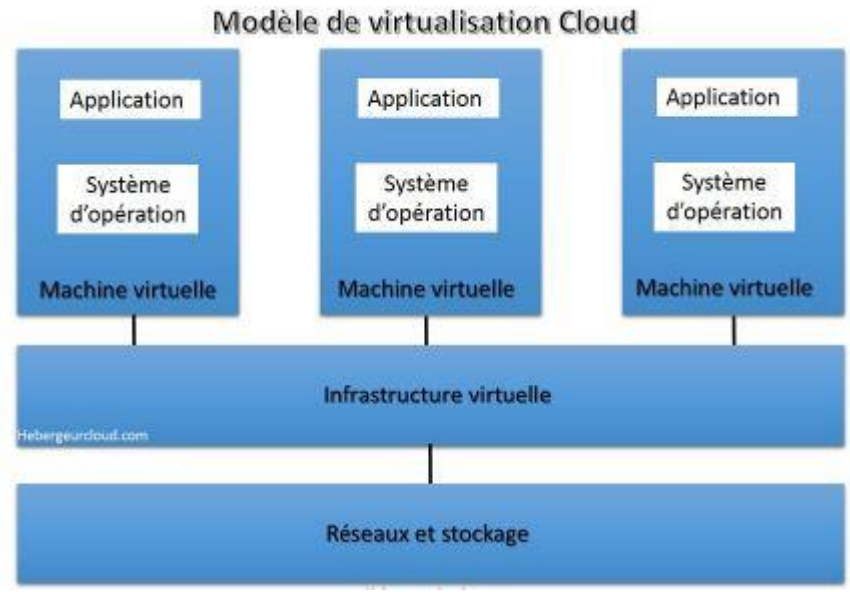

Figure 1.2 La virtualisation dans les environnements infonuagiques.

La virtualisation [19] est le processus de partitionnement de ressources physique avec une telle privatisation. (Application (session); OS (VM)…etc. C'est une évolution qui sert à combiner les ressources matérielles/logiciels du réseau dans une seule unité administrative-nommée un réseau virtuel.

3. Les caractéristiques du Cloud

Dans cette section, nous allons présenter les aspects techniques, qualitatifs et économiques d'infonuagique.

3.1. Les Aspects Techniques :

Dans cette sous-section nous allons présenter les aspects techniques du cloud.

Multi-tenant L'architecture Multi-tenant d'une application est une architecture où une seule instance est partagée par plusieurs utilisateurs. Par contre, une architecture multi instance est une architecture où chaque client possède sa propre instance. Les

bénéfices d'une architecture Multi-tenant sur multi-instances se traduit par une meilleure efficacité de la maintenance de l'application car le développeur de l'application pourra mettre à niveau son application une seule fois au lieu de la faire pour chaque instance comme c'est le cas pour une architecture multi-instance. L'infonuagique rend plus efficace le modèle multi-tenant, grâce à la technologie de virtualisation permet aux utilisateurs utilisant la même instance d'application de contrôler leurs données situées sur la même base de donnée de manière indépendante.

La sécurité[1] est l'une des préoccupations majeures de l'infonuagique non seulement pour le fournisseur mais aussi pour le consommateur. Le fournisseur doit s'assurer que les données de ses clients ainsi que leurs applications sont protégées. Le consommateur de son côté doit sécuriser ses données à travers des mots de passes et des mesures d'authentification élaborées.

L'environnement de programmation Il devrait être capable d'adresser des préoccupations comme les multiples domaines administratifs, les grandes variations de l'hétérogénéité des ressources, la stabilité des performances, la gestion des exceptions dans des environnements très dynamiques, etc. [9] Grâce aux API prédéfinies, les utilisateurs peuvent accéder, configurer et programmer les applications du Cloud.

3.2. Les aspects qualitatifs

[1] La propriété sécuritaire du Cloud a été inspirée de l'article Wikipédia [10].

Parmi les caractéristiques qualitatives du Cloud qui se réfèrent aux propriétés du modèle du cloud et non pas aux exigences technologiques de ce dernier. Cette sous-section a été inspirée de [9].

Élasticité dans le Cloud peut être comparée à la propriété physique de l'élastique, c'est-à-dire sa capacité à s'élargir puis à reprendre sa forme originale. Avec cette propriété, on peut satisfaire les besoins de l'utilisateur de manière dynamique en lui fournissant les ressources requises qui peuvent varier à travers le temps.

La disponibilité fait référence à une capacité pertinente qui répond aux exigences spécifiques des services de la sous-traitance (externalisation). Les indicateurs de qualité de service comme le temps de réponse et le débit doivent être garantis, de sorte à satisfaire les garanties de qualités avancées aux utilisateurs d'infonuagique.

La fiabilité représente la capacité d'assurer le fonctionnement du système de manière constante sans interruption. Grâce à l'utilisation des sites redondants, la possibilité de perdre les données et le code est diminuée de façon spectaculaire. Ainsi l'infonuagique est adapté pour la continuité des activités et de la restauration après sinistre.

- **L'agilité** est une exigence de base pour l'infonuagique. Les fournisseurs du cloud sont capables de réagir en ligne à l'évolution de la demande des ressources et des conditions environnementales.

3.3. L'aspect économique :

Parmi les caractéristiques de l''infonuagique, on retrouve le modèle économique « pay-as-you-go ».

> Le modèle **Pay-as-you-go** suit le modèle économique de la facturation des services publics comme l'électricité et le gaz. Vous payez seulement la quantité que vous avez utilisée ainsi que sa durée correspondante. Avant la venue de ce modèle économique, les entreprises devaient d'abord estimer la charge de travail maximale qui doit être satisfaite avant de lancer leurs projets. Cela n'est plus requis par le modèle « pay-as-you-go » puisque le consommateur va payer seulement les ressources utilisées et les ressources peuvent s'agrandir grâce à l'élasticité que le cloud possède comme caractéristique.

4. Les technologies connexes liées au L'infonuagique:

L'infonuagique a évolué sur des décennies de recherche dans différentes technologies, dont il a hérité des caractéristiques et des fonctionnalités telles que les environnements virtualisés, l'infonuagique autonome, la grille informatique, et le calcul distribué. La Figure1.3 illustre l'évolution vers l'infonuagique dans l'hébergement des applications logicielles [11]. En fait, l'infonuagique est souvent comparé aux technologies connexes, dont chacun partage certains aspects avec l'infonuagique.

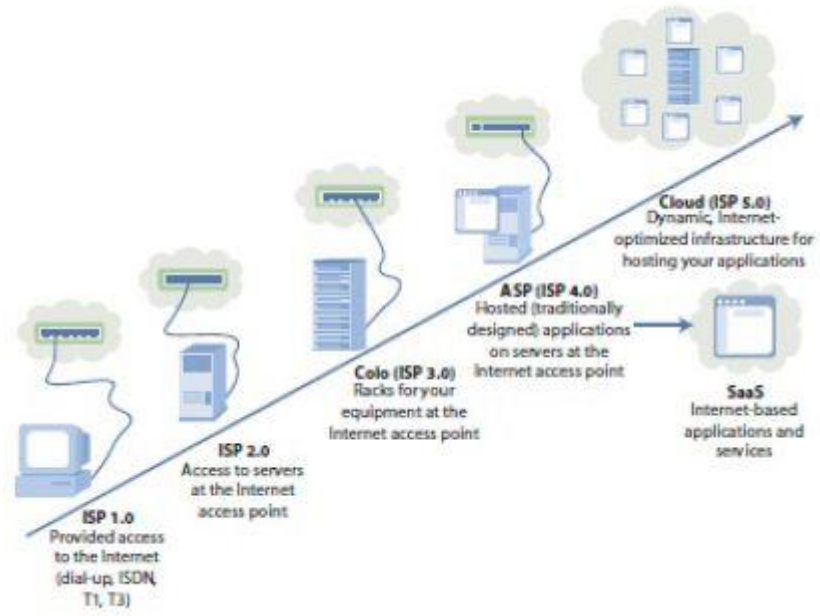

Figure 1.3 L'évolution vers l'infonuagique dans l'hébergement d'applications logicielles.

5. Modèle de déploiement du Cloud

Il existe trois modèles de déploiement du Cloud communément utilisé : privé, public, et hybride. Un modèle additionnel est le Cloud de communauté, qui est moins répandu par rapport aux trois modèles précédemment cités (voir La Figure 1.4) :

- **Cloud privé** a une portée délimitée par l'organisation où il est implémenté.
- **Cloud public** est un ensemble de ressources fournit par des organisations tierces. Parmi les plus populaires on peut citer : Amazon Web Services, Google AppEngine, et Microsoft Azure.
- **Cloud hybride** est un ensemble de ressources computationnels fournit par des cloud publics et privés.
- **Cloud de communauté** partage les ressources computationnelles à travers plusieurs organisations appartenant à une communauté spécifique.

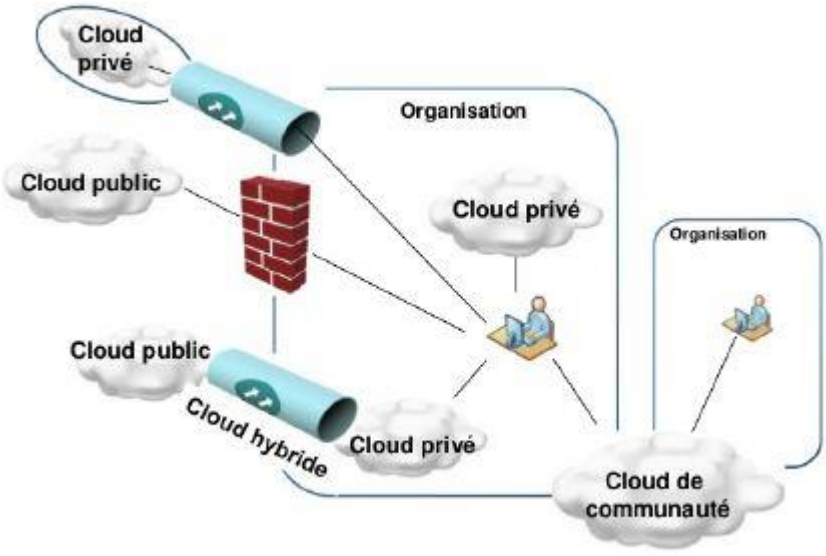

Figure 1.4 Les modèles de déploiement dans L'infonuagique

6. Service du Cloud

Il existe trois types de services dans l'infonuagique : IaaS (Infrastructure as a Service), Paas (Platform as a service) et SaaS (Software as a Service), comme il est montré dans la Figure 1.5.

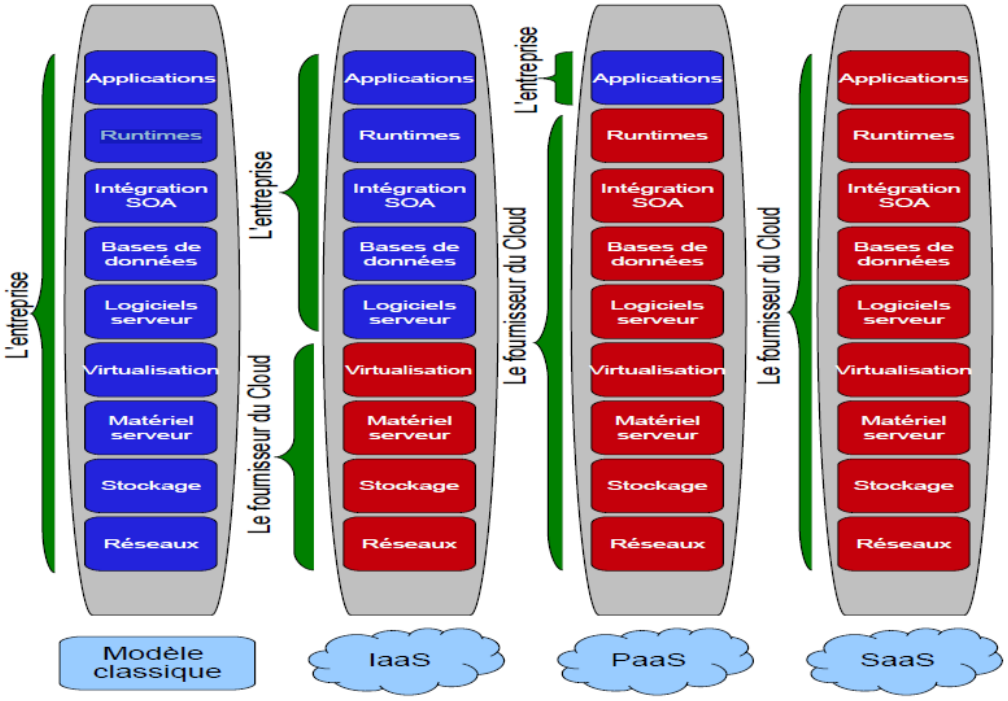

Figure 1.5 Les modèles de service dans l'infonuagique.

6.1. Software as a Service (SaaS)

Est un modèle d'approvisionnement d'application où les applications sont hébergées au sein du vendeur ou d'un fournisseur de service. L'accès à une telle application se fait à travers un réseau généralement l'internet. Grâce à ce modèle, l'utilisateur n'a pas à se préoccuper de l'ensemble des éléments essentiels pour l'exécution de l'application demandée : Infrastructure sous-jacente, système d'exploitation, etc.

6.2. Platform as a Service (PaaS)

Dans le modèle de PaaS, le fournisseur délivre tous ce qui est nécessaire afin que l'utilisateur puisse développer l'application désirée. Dans ce modèle, l'utilisateur a le contrôle sur l'application ainsi que l'environnement de développement de l'application, mais tout ce qui est attrait à l'infrastructure sous-jacente est géré par le fournisseur.

6.3. Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) fournit des ressources virtualisées, accessible à travers l'internet. Les ressources fournies sont virtualisées dans le sens où ces ressources sont une agrégation de plusieurs ressources fournit par plusieurs centres de donnés qui peuvent être géographiquement distribués. Grâce donc à la technologie de virtualisation, l'utilisateur aura l'impression qu'il a en possession un bloc de ressources, hors en réalité ces ressources peuvent être une agrégation de ressources distribuées géographiquement.

7. Comparaison générale des simulateurs de Cloud computing

Dans le tableau 1, un état comparatif de trois simulateurs est représenté : MDCSim [14], CloudSim [15] et GreenCloud [16]. Il montre les différents paramètres et sur la base de ces paramètres se caractérisent les différents simulateurs [16]. Cette section s'appuie sur l'article [17].

Paramètres	MDCSim [14]	CloudSim [15]	GreenCloud [14]
Communication Réseau	Limité	Limité	Complet
Support graphique	Aucun	Limité (Cloud Analyst)	Limité (NewtworkAnimator)
Disponibilité	Commerciale	OpenSource	OpenSource
Plateforme	CSIM	SimJava	NS2
Modèle d'application	Calcul	Calcul et transfert de donné	Calcul, transfert de donné et date limite d'exécution
Temps de Simulation	Secondes	Secondes	Dizaine de minutes
Langage/Script	C++/Java	Java	C++/OTcl
Modèle physique	Aucun	Aucun	Disponible en utilisant plug in

Tableau 1.1 Comparaison des outils de Simulation du Cloud.

1. **Plate-forme** - Les simulateurs MDCSim et CloudSim sont des simulateurs basé évènements. Le simulateur Green Cloud est en fait conçu comme une extension du simulateur NS2 qui est codé en C++ avec une couche supplémentaire OTcl implémenté au-dessus.
2. **Modèles d'application** - Les trois simulateurs implémentent des modèles d'applications utilisateur comme des objets simples décrivant les exigences de calcul pour l'application. En outre, GreenCloud et CloudSim précisent les exigences de communication des applications en termes de la quantité de données à transférer, avant et après l'achèvement de la tâche. Le modèle d'application implémenté dans le CloudSim s'accorde bien avec le calcul haute performance (HPC). Les tâches HPC, étant à calcul intensif et n'ayant pas de délai de réalisation spécifique sont des applications typiques pour les réseaux Grid. Dans l'infonuagique, les exigences de qualité de service pour l'exécution des requêtes utilisateur sont définies dans la SLA. Par conséquent, GreenCloud étend le modèle d'application utilisateur par l'ajout d'un délai d'exécution prédéfinie.
3. **Temps de Simulation** - Le temps nécessaire à la simulation dépend de nombreux facteurs comme le scénario simulé ou le matériel utilisé pour l'exécution du logiciel de simulation. En général, CloudSim et MDCSim, étant des simulateurs à base d'événements, sont plus rapides et s'étendent à un plus grand nombre de nœuds de centres de données. Néanmoins, le simulateur GreenCloud réalise encore des temps de simulation raisonnables. Il est de l'ordre de plusieurs dizaines de minutes pour une heure de temps de simulation lorsqu'il simule un centre de données typique avec quelques milliers de nœuds. En dehors du nombre de nœuds, la durée de la simulation est fortement influencée par le nombre de paquets de communications produits, ainsi que le nombre de fois où ils sont traités par des routeurs de réseau pendant leurs transferts. En conséquence, un Datacenter typique simulé dans GreenCloud peut être composé de milliers de nœuds tandis que le CloudSim et MDCSim Java peuvent simuler des millions d'ordinateurs.
4. **Langage/Script** - Le langage/script des simulateurs qui signifie la plateforme dans laquelle les simulateurs vont effectuer leurs implémentations.

8. Conclusion

Dans le présent chapitre, nous avons présenté les Paradigmes Infonuagiques qui représentent le cadre générale de notre Mémoire, les notions, les définitions, les avantages et les inconvénients, ainsi que quelques plateformes pour la modélisation de ce paradigme

Nous présentons dans le chapitre suivant un état de l'art sur les solutions disponible pour la planification des taches et l'allocation de ressources dans ce contexte.

Chapitre II : État de l’art sur l’allocation des ressources.

1. Introduction

Les systèmes informatique "Infonuagique" sont de plus en plus utilisés grâce à leur capacité de stockage et de calcul. L'un des problèmes importants de ces systèmes est la planification la gestion de ressources.

Récemment, la communauté scientifique a publié plusieurs approches et méthodes d'allocation de ressources, en s'efforçant de tenir compte des différentes spécificités de systèmes de grille de données : l'hétérogénéité, l'instabilité du système et la grande échelle. La structure de gestion centralisée prédomine dans les méthodes proposées, malgré les risques encourus par cette solution dans les systèmes à grande échelle.

Dans ce chapitre on s'intéresse à la description d'un état de l'art non exhaustive sur le sujet d'étude de notre mémoire qu'il s'agit d'allocation des ressources et de la planification des taches.

2. Définition

Allocation : est le fait d'associer un espace de ressources (Ex. mémoire) à certaines données ou un programme pour son exécution.

Ressources : La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tache, disponible en quantité limitée et sa capacité. Plusieurs types de ressources sont `à distinguer. Une ressource est renouvelable si après avoir ´et´e allouée à une ou plusieurs taches, elle est `à nouveau disponible en même quantité (les hommes, les machines, l'´equipement en général) ; la quantité de ressource utilisable `a chaque instant est limitée. Dans le cas contraire, elle est consommable (matières premières, budget) ; la consommation globale (ou cumul) au cours du temps est limitée. Une ressource est doublement contrainte lorsque son utilisation instantanée et sa consommation globale sont toutes deux limitées (l'argent en est un bon exemple). Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Sa courbe de disponibilité est en général connue a priori, sauf dans les cas ou` elle dépend du placement de certaines tâches génératrices.

Type de ressources : Les ressources incluent les ressources de CPU, de mémoire, d'alimentation, de stockage et de réseau.

CPU (**C**entral **P**rocessing **U**nit) (**U**nité **C**entrale de **T**raitement) : est le cerveau de l'ordinateur. Il permet de manipuler des informations numériques, c'est-à-dire des informations codées sous forme binaire, et d'exécuter les instructions stockées en mémoire.

RAM (Random **A**ccess **M**emory) : est un type de mémoire qui équipe tout ordinateur et mobile qui permet de stocker des informations provisoire. Son but n'étant pas de ranger de l'information mais d'y accéder rapidement et provisoirement.

Stockage : est un moyen de garder et préserver des données en sécurité. Il existe plusieurs sortes d'architectures de stockage, la plus courante est de disposer des serveurs en local, ils sont alors raccordés directement aux serveurs de production.

Réseau : est un ensemble des moyens matériels et logiciels mis en œuvre pour assurer les communications entre ordinateurs, stations de travail et terminaux informatiques. Tout ou partie de ces matériels peuvent être considérés comme faisant partie du réseau.

- **Tache :** est une entité élémentaire localisée dans le temps, par une date de début et une date de fin, et dont la réalisation nécessite une durée préalablement définie. Elle est constituée d'un ensemble d'opérations qui requiert, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif.

2.1. Allocation des Ressources

L'allocation de ressources est le processus de division et de répartition d'une quantité limitée des ressources disponibles à des usages alternatifs concurrents, satisfaisants des besoins illimités. Etant donné que la pénurie est endémique dans le monde (désirs et besoins illimités, mais des ressources limitées), tous les besoins ne peuvent être satisfaits par les ressources disponibles. Des choix doivent être faits. Ces choix et ces décisions sont le processus d'allocation des ressources.

Dans l'infonuagique, l'allocation de ressources est le processus d'attribution des ressources disponibles pour les applications infonuagiques sur Internet. L'allocation des ressources qui n'est pas gérée avec précision empêche le bon fonctionnement des services. L'approvisionnement des ressources résout ce problème en permettent aux fournisseurs de services de gérer les ressources pour chaque application.

2.2. Planification des Taches

La Planification dans l'infonuagique est classée au niveau de l'utilisateur et au niveau du système [59]. Au niveau de l'utilisateur, la planification traite les problèmes soulevés par la prestation de services entre les fournisseurs et les clients. La programmation au niveau de système gère la gestion des ressources dans les centres de données. Le Datacenter se compose de plusieurs machines physiques. Des millions de taches des utilisateurs sont reçues : l'attribution de ces taches aux machines physiques se fait au niveau des centres de données. Cette affectation de planification joue un rôle significatif sur les performances du Datacenter. En plus de l'utilisation du système, d'autres exigences comme la qualité de service, le SLA (Service Level Agreement), le partage des ressources, la tolérance aux pannes, la fiabilité, la satisfaction en temps réel, etc. Devraient être pris en considération.

Les ordonnanceurs basés sur le modèle du marché et sur les enchères sont appropriés pour réguler l'offre e la demande des ressources sur le nuage. L'allocation des ressources en fonction du modèle économique de la marché est efficace dans un environnement

Infonuagique ou les ressources sont virtualisées et livrés à l'utilisateur en tant que service. Une suite d'algorithmes de planification de taches axées sur la base du modèle de marché pour les environnements distribués hétérogènes est proposée dans le travail [20].

Le développement d'un modèle de tarification en utilisant le partage du processeur dans les Clouds, l'application de ce modèle de tarification aux services composites avec dépendance et le développement de deux ensembles de planification et de profit conduit aux algorithmes proposés dans [21].

3. Ressources ou Taches

À un niveau suffisamment élevé d'abstraction, un problème d'allocation de tâches peut être réduit à un problème d'allocation de ressources. Toutefois, il faut prendre en compte que les tâches sont comme des ressources ayant une utilité négative vis-à-vis de l'agent. De plus, les tâches se distinguent des ressources par le fait que ces dernières sont couplées à des contraintes définissant les combinaisons cohérentes qui peuvent leur être autorisées. Comme par exemple l'exécution d'une tâche comme pré condition pour l'exécution d'une autre tâche.

4. Etat de l'art sur l'allocation et la planification

Dans ce sujet d'allocation des ressources et planification des taches, plusieurs solutions sont développées dans le domaine. Parmi ces solutions

4.1. Les solutions heuristiques de planification des tâches

4.1.1. Première solution

FCFS ou **FIFO** (**F**irst **I**n, **F**irst **O**ut): est une structure de données basée sur le principe du premier entré, premier sorti ce qui veut dire que les premiers éléments ajoutés à la file seront les premiers à en être retirés. C'est l'un des algorithmes de planification les plus simples que nous avons alloué le CPU dans l'ordre dans lequel la tache arrive. Ce qui signifie que le premier travail sera traité en premier, sans autres préférences.[22]

Algorithme FCFS :

Initialiser les tâches.

Première tâche assignée à la file d'attente et ajout tâches jusqu'à n nombres.

Ajouter la prochaine tâche '' I '' à la dernière position dans la main queue.

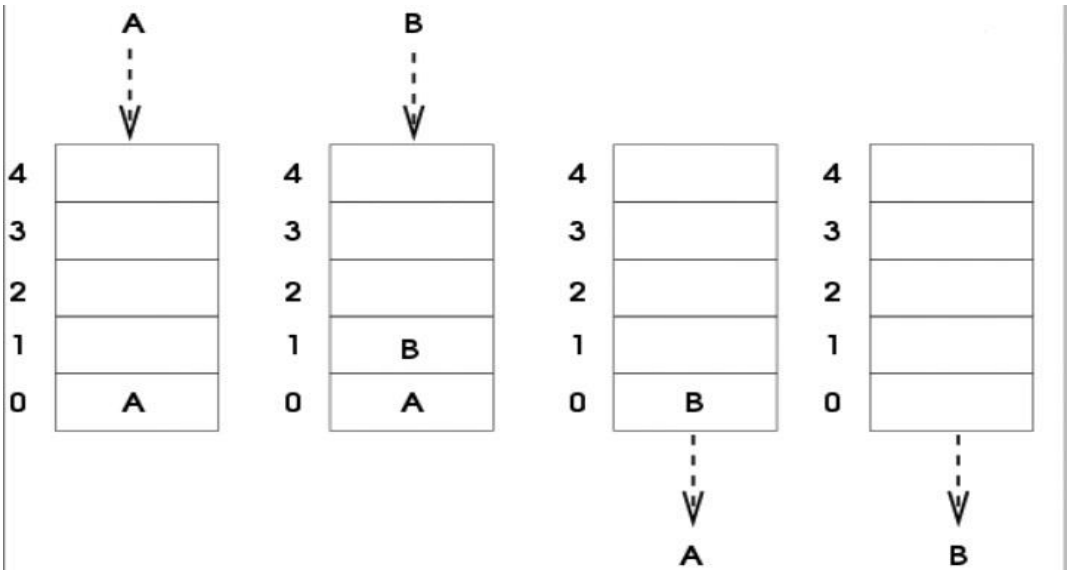

Figure 2.1 Algorithme de FCFS

4.1.2. Deuxième solution

SJF ou **SJN (S**hort **J**ob **N**ext): est un algorithme servant à choisir lequel de plusieurs taches sera traité en premier par le processeur. Le choix se fait en fonction du temps d'exécution estimé de la tâche. Ainsi, l'ordonnanceur va laisser passer d'abord le plus court des taches de la file d'attente.[22]

Il existe deux versions de cet algorithme : une version préemptive, et une version non préemptive. Dans cette dernière, une tache ayant pris le contrôle de l'UC ne le quitte qu'une fois la rafale terminée.

La version préemptive, aussi appelée **SRTF**, **Shortest Remaining Time First** (plus court temps restant en premier). Est plus souple. Si une tache dont le temps d'exécution est plus court que le reste du temps d'exécution de la tâche en cours de traitement entre dans la file d'attente, alors il prendra sa place. Il y a alors une commutation de contexte, et le traitement de la tache interrompue reprendra plus tard là où il avait été laissé.[22]

SJF s'avère un des algorithmes les plus rentables en ce qui concerne la réduction du temps passé dans la file d'attente des taches. Toutefois il est rarement utilisé en dehors

d'environnements spécialisés, car il nécessite une évaluation précise du temps d'exécution de toutes les taches en attente de traitement.

Algorithme de SJF :

```
For i = 0 to i < main queue-size
If task i+1 length < task i length then
        Add task i+1 in front of task i in the queue.
End if
If main queue-size = 0 then
        Task i last in the main queue
End if
End for
```

4.1.3. Troisième solution

Round-robin : est un algorithme de Planification courant dans les systèmes d'exploitation et il est adapté aux systèmes travaillant en temps partagés. Il alloue le processeur aux processus à tour de rôle, pendant une tranche de temps appelée quantum. Dans la pratique le quantum s'étale entre 10 et 100ms.[24]

La performance de l'algorithme de Round Robin dépend largement de la taille du quantum. Si le quantum est très grand, la politique Round Robin serait similaire à celle du FCFS. Si le quantum est très petit, la méthode Round Robin permettrait un partage du processeur : chacun des utilisateurs aurait l'impression de disposer de son propre processeur.[24]

Cependant le quantum doit être choisi de sorte à ne pas surcharger le système par des fréquentes commutations de contexte.

Algorithm RRS:

Conservez la file d'attente prête en tant que file d'attente FIFO des taches.

De nouvelles tâches sont ajoutées à la fin de la file d'attente prête.

Le planificateur de la CPU sélectionne la première tâche de la file d'attente prête, règle une minuterie sur interrompre après un intervalle de temps, et distribue la tâche.

La tache peut avoir une rafale de CPU inférieure à une fois le quantum.

Dans ce cas, la tache libère volontairement le CPU.

Le planificateur passera ensuite à la tache suivant dans la file d'attente prête.

Sinon, si la rafale du processeur de la tâche en cours est supérieure à une heure quantum,

La minuterie s'éteindra et provoquera une interruption du système d'exploitation.

Un changement de contexte sera exécuté, et la tâche sera placée à la fin de la file d'attente prête. ou Le planificateur CPU sélectionnera ensuite la tache suivante dans la file d'attente prête.

4.1.4. Quatrième solution

Cet algorithme associe à chaque tache une priorité. Et le planificateur sera affecté aux taches le plus haute priorité. Cette priorité varie selon les systèmes et peut aller de 0 à 127.

Les priorités peuvent être définies en fonction de plusieurs paramètres : le type de tache, les limites de temps, les limites mémoires, …etc.

Critique de la méthode :

Une situation de blocage peut survenir si les taches de basse priorité attendent indéfiniment le planificateur, alors que des taches de haute priorité continuent à affluer.

Pour éviter une telle situation, on peut utiliser la technique dite du vieillissement. Elle consiste à incrémenter graduellement la priorité des taches attendant dans le système pendant longtemps. Par exemple, nous pourrions incrémenter de 1 la priorité d'une tache en attente toutes les 15minutes. En fin de compte, même une tache ayant une priorité initiale égale à 0 aurait la plus haute priorité dans le système et serait exécuté.[23]

Algorithme de PSA

- For i = 0 to i < main queue-size
    - If priority (task i+1) > priority (task i) then
      Add task i+1 in front of task i in the queue
    - End if
- End for

4.1.5. Cinquième solution

L'algorithme commence par calculer le temps d'exécution minimale pour toutes les taches puis la valeur minimale entre ces temps minimum est choisie ; qui représente le temps minimum d'exécution parmi toutes les taches sur les ressources. Ensuite, en fonction de ce temps minimum, la tâche est ordonnée sur la machine correspondante. Puis le temps d'exécution pour toutes les autres tâches sont mises à jour sur cette machine en ajoutant le temps d'exécution de la tache assignée à des temps d'exécution des autres taches sur cette machine/ressource et la tache assignée est supprimée de la liste des taches. Ensuite, la même procédure est répétée jusqu'à ce que toutes les taches soient assignées sur les ressources [22].

Un exemple d'application de l'algorithme pour six taches et quatre machines virtuelles, les temps d'exécution (en milliseconde) de toutes les taches sur toutes les machines sont présentés sur le tableau suivant :

	M_0	M_1	M_2	M_3
T_0	160	400	80	200
T_1	40	100	20	50
T_2	100	250	50	125
T_3	20	50	10	25
T_4	140	350	70	175
T_5	80	200	40	100

Tableau 2.1 Le temps d'exécution des taches (Algorithme Min-Min)

Le résultat d'exécution des taches selon l'algorithme Min-Min est donné dans la figure suivante :

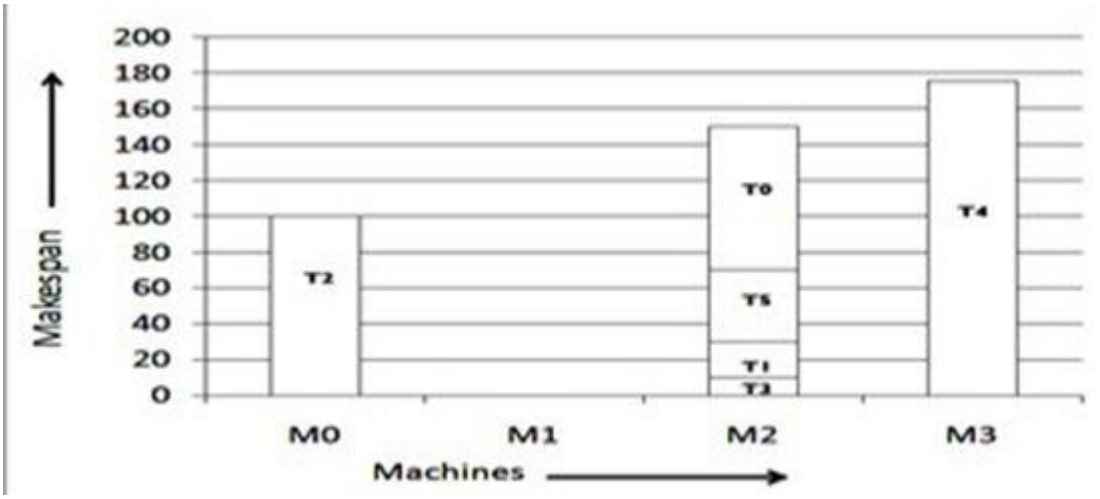

Figure 2.2 résultat d'exécution des taches selon Min-min

$Ct_{ij} = Et_{ij} + rt_j$

rt_j Représente le temps de disponibilité de la ressources Et_{ij} représente le temps d'exécution de la tache i. Pseudo Code de l'algorithme Min-Min est représenté ci-dessous :

Algorithme de Min-Min

- For all submitted tasks in the set Ti
- For all resources Rj
- $Ct_{ij} = Et_{ij} + rt_j$; end for ; end for;
- Do while tasks set is not empty
- Find task T_k that cost minimum execution time
- Assign T_k to the resources R_j while gives minimum expected complete time
- Remove T_k from the task set
- Update ready time rt_j for select R_j
- Update Ct_{ij} for all T_i
- End Do

4.1.6. Sixième solution

L'algorithme Max-min suit le même principe que l'algorithme Min-min à l'exception des propriétés suivantes : Après avoir calculer les temps d'exécution minimum, la valeur maximale est sélectionnée, qui est la durée maximale parmi toutes les taches sur les ressources. Ensuite, en fonction de ce temps maximum, la tâche est ordonnancée sur la machine correspondante. Puis le temps d'exécution pour toutes les

autres tâches sont mises à jour sur cette machine en ajoutant le temps d'exécution de la tache assignée à des temps d'exécution des autres taches sur la machine qui acquise la tache sélectionnée et la tache assignée est supprimée de la liste des taches. La même procédure est répétée jusqu'à ce que toutes les taches soient assignées sur les ressources [23].

Le résultat d'exécution des taches selon l'algorithme Max-min est donné dans la figure suivante en utilisant les mêmes paramètres du tableau 2.1

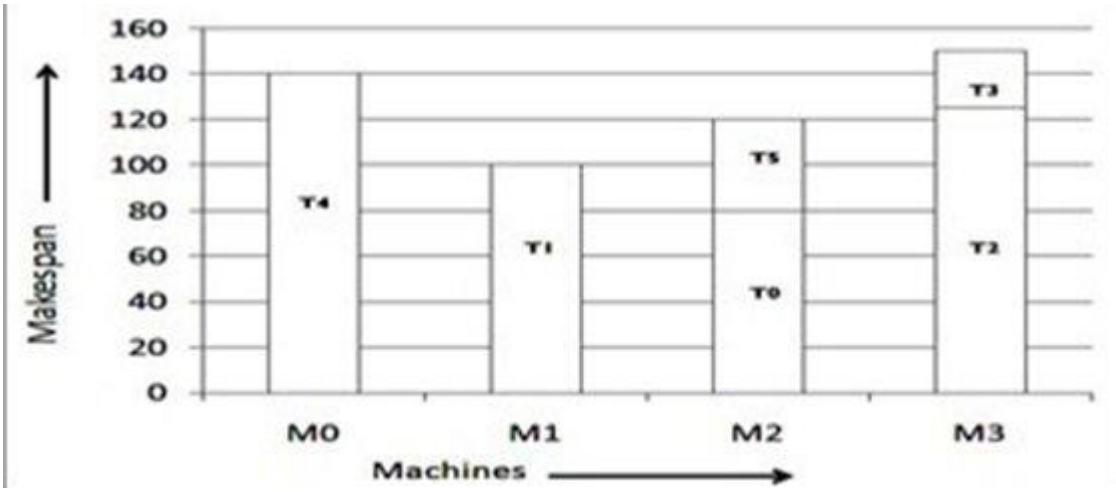

Figure 2.3 Résultat d'exécution des taches selon Max-min

Algorithme de Max-Min

- For all submitted tasks in meta-task T_i
- For all resource R_j
- Compute $C_{ij} = E_{ij} + r_j$
- While meta-task is not empty
- Find the task Tm consumes maximum completion time
- Assign task Tm to the resource R_j with minimum execution time
- Remove the tasks Tm from meta-tasks set
- Update r_j for selected R_j
- Update C_{ij} for all T_i

4.1.7. Septième solution

L'algorithme construit une matrice C représente le temps d'achèvement de la tâche Ti sur la ressource Rj. Si le nombre de ressources disponibles est impair, la stratégie Min-min est appliquée pour affecter la première tâche, sinon la stratégie Max-Min est appliquée. Les tâches restantes sont affectées à leurs ressources appropriées par l'une des deux stratégies,

alternativement. Par exemple, si la première tâche est affectée à une ressource par la stratégie Min-min, la tâche suivante sera affectée par la stratégie Max-Min. Au tour suivant, la tâche commence avec une stratégie différente de la dernière ronde. Par exemple, si le premier tour commence avec la stratégie Max-Min, le second tour commencera avec la stratégie Min-min. [24]

Algorithme de RASA

- For all tasks T_i in meta-task M_v
- For all resources R_j
- $C_{ij} = E_j + r_j$
- Do until all tasks in M_v are mapped
- If the number of resources is even then
- For each task in M_v find the earliest completion time and the resource that obtained it
- Find the task T_k with the maximum earliest completion time
- Assign task T_k to the resource R_j that gives the earliest completion time
- Delete task T_k from M_v
- Update r_i
- Update C_{ij} for all i
- Else
- For each task in M_v find the earliest completion time and the resource that obtained it
- Find the task T_k with the minimum earliest completion time
- Assign task T_k to the resource R_j that gives the earliest completion time
- Delete task T_k from M_v
- Update r_i
- Update C_{ij} for all i
- End if
- End do

4.2. Les Solution méta-heuristique de planification des tâches

4.2.1. Première solution

Pour la planification des tâches de workflow sur les ressources cloud, une nouvelle méthode méta-heuristique appelée l'algorithme BAT est appliqué. Cela s'appelle parce que son fonctionnement est basé sur le comportement d'écholocation des virtuels bats.

Ce comportement d'écholocation peut être formulé de telle sorte qu'il s'associe à la fonction objective qui doit être optimisé. L'algorithme a été développé par Xin-She Zang en l'an 2010 et est basé sur les règles suivantes : [25]

Pour détecter la distance, les chauves-souris (bats) utilisent leur comportement d'écholocation.

Les chauves-souris (bats) volent avec une certaine vitesse v_i à la position x_i. Ils émettent des impulsions dont la fréquence est ajustée automatiquement et le taux de l'impulsion émise $r \in [0\ ,\ 1]$ est ajusté en fonction de la distance de la chauve-souris par rapport à sa cible.

Sonie des chauves-souris varie d'une grande valeur L_0 à une valeur minimale L_{min}

Algorithme de BAT

- Initialize the bat population x_i and v_i (i = 1,2,…., n)
- Initialize frequencies f_i pulse rates r_i and the loudness L_i
- While (i < Max number of iterations)
- Generate new solution by adjusting frequency
- Update velocities and locations/solutions using following equations
- $f_i = f_{min} + (f_{max} - f_{min})$ rand
- $v_i^{s+1} = v_i^s + (x_i^s - x_0)\ f_i$
- $x_i^{s+1} = x_i^s + v_i^{s+1}$
- If (rand > r_i)
- Select a solution among the best solution
- Generate local solution among the selected best solution
- End if
- Generate a new solution by flying randomly
- If (rand < L_i & $f(x_i) < f(x_0)$)
- Accept the new solution

- Increase r_i and reduce L_i using constants β and μ according to following equations
- $L_i^{s+1} = \beta\ L_i^s$
- $r_i^{s+1} = r_i^0$[1- exp(-μs)]
- End if
- Rank the bats and find the current best x
- End while

4.2.2. Deuxième solution

Méthode de résolution de problèmes qui utilise la génétique comme modèle de résolution de problèmes. C'est une technique de recherche pour trouver une solution optimisée. GA gère une population de solution possible. Chaque solution est représentée à travers un chromosome. L'algorithme génétique est une méthode d'ordonnancement dans laquelle les tâches sont affectées à des ressources en fonction de solutions individuelles (appelées planifications dans le contexte de l'ordonnancement), indique quelle ressource doit être affectée à quelle tâche.

L'algorithme génétique est basé sur le concept biologique de génération de population. Dans l'algorithme génétique, la population initiale est générée de manière aléatoire. L'algorithme génétique est une recherche aléatoire méthode. [26]

.**Algorithme de GA**

- Randomly create the initial population s
- While the termination criterion is not met
- f = FitnessFunction(s) E
- v = Selection(s, f) D
- v' = Crossover(v) T
- v" = Mutation(v0) T
- s = Reproduction(v00)
- End

4.2.3. Troisième solution

League Championship Algorithm (LCA) : C'est une nouvelle technique d'optimisation conçu sur la base de l'inspiration des compétitions de football une ligue de championnat. Le nouvel algorithme est une population schéma algorithmique basé pour l'optimisation globale

sur un espace de recherche continue. LCA a été conçue pour être un algorithme basé sur la population stochastique pour continue optimisation globale qui essaie d'imiter un championnat où les clubs de football synthétiques participent à ligue artificielle pour une période de temps. L'algorithme LCA a été utilisé dans de nombreux domaines et réalisé de manière crédible bien par rapport à d'autres schémas d'optimisation connus algorithmes heuristiques.[24]

4.2.4. Quatrième solution

Particle Swarm Optimisation (PSO) est un algorithme intelligent basé sur la température, influencé par le comportement social des animaux, comme un troupeau d'oiseaux trouvant une source de nourriture ou une école se protégeant contre les prédateurs. Une particule dans PSO est analogue à un oiseau ou un poisson volant à travers un espace de recherche (problème). Le mouvement de chaque particule est coordonné par une vitesse qui a à la fois la grandeur et la direction. Chaque position de particule à chaque instant est influencée par sa meilleure position et la position de la meilleure particule dans un espace problématique. La performance d'une particule est mesurée par une valeur d'aptitude, qui est spécifique au problème. L'algorithme PSO est similaire à d'autres algorithmes à évolution. Dans PSO, la population est le nombre de particules dans un espace de problèmes. Les particules sont initialisées de manière aléatoire. Chaque particule a une valeur de comportement qui sera évaluée par une fonction d'aptitude à optimiser à chaque génération. Chaque particule connaît sa meilleure position et la meilleure position jusqu'ici parmi le groupe entier de particules. Le meilleur résultat d'une particule est le meilleur résultat (valeur d'aptitude) atteint jusqu'à présent par la particule.[23]

Algorithme de PSO

1. Set particle dimension as equal to the size of ready tasks in $\{t_i\} \epsilon$ T
2. Initialize particles position randomly from PC = 1, …, j and velocity v_i randomly
3. For each particle, calculate its fitness value as in Equation 4
4. If the fitness value is better than the previous best pbest, set the current fitness value as the new pbest
5. After steps 3 and 4 for all particles, select the best particle as gbest
6. For all particles, calculate velocity using Equation 6 and update their positions using Equation 7
7. If the stopping criteria or maximum iteration is not satisfied, repeat from Step 3

4.2.5. Cinquième solution

Ant Colony Optimization (ACO) est une méta-heuristique inspirée par l'observation de vraies colonies de fourmis et basée sur leur comportement de recherche de nourriture collective. Les fourmis sont des insectes sociaux et vivent en colonies. Leur comportement est régi par l'objectif de la survie des colonies. Lors de la recherche de nourriture, les fourmis voyagent fréquemment entre leur nid et les sources de nourriture. Au début, les fourmis explorent la zone entourant leur nid de manière aléatoire. Tout en se déplaçant, les fourmis déposent des substances spéciales appelées phéromones le long de leurs chemins. Les fourmis peuvent sentir les phéromones. En choisissant leur chemin, ils ont tendance à choisir, en probabilité, chemins marqués par de fortes concentrations de phéromones. Dès qu'une fourmi trouve une source de nourriture, elle évalue la quantité et la qualité de la nourriture et en ramène une partie au nid. Pendant le voyage de retour, la quantité de phéromones qu'une fourmi les feuilles sur le sol peuvent dépendre de la quantité et de la qualité de la nourriture. Les sentiers de phéromones guideront d'autres fourmis vers la source de nourriture. La communication indirecte entre les fourmis via les traînées de phéromones leur permet de trouver les chemins les plus courts entre leur nid et les sources de nourriture.[24]

Les algorithmes de ACO sont diffèrent principalement par rapport aux trois points suivants.

- Mise à jour de phéromone.
- Définition de phéromone et d'informations heuristiques.

Procedure ACO

Begin
Initialize the pheromone
While (stopping criterion ***not*** satisfied) ***do***
Position each ant ***in*** a starting node
Repeat
For each ant ***do***
Chose next node by applying the state transition rate
End for
Until every ant has built a solution
Update the pheromone
End while
End

5. Les Styles d'allocation

Il existe deux types de styles d'allocation de ressources, statique et dynamique. Dans notre sujet on focalise toute notre attention sur l'allocation statique comme un commencement de programmation dans le domaine :

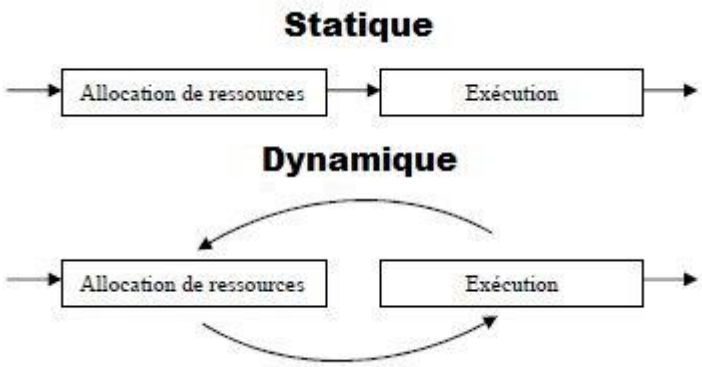

Figure 2.4 la stratégie d'allocation de ressources statique et dynamique itérative

5.1. Allocation statique

Le type statique consiste à effectuer l'allocation de ressources entièrement pour une requête avant la phase d'exécution en vertu des informations sur les ressources qui sont disponibles au moment de la planification. Cela assure une utilisation maximale du parallélisme. Cependant, dans un environnement dynamique comme la grille de données, il y a une évolution rapide de tous les paramètres, les métadonnées peuvent rapidement devenir obsolètes et le plan d'allocation de ressources choisi peut devenir sous-optimal [27].

5.2. Allocation dynamique

Le type dynamique consiste à effectuer l'allocation de ressources par parties, en alternance avec l'exécution de ces parties du plan d'exécution. Ce type de stratégie peut être considéré comme raisonnable dans des environnements avec une dynamicité élevée des nœuds et en absence de métadonnées précises. Toutefois, en l'absence d'un plan d'exécution complètement placé et le principe d'exécution successive pour des opérations dépendantes, il

n'est pas possible d'utiliser le parallélisme pipeline, ce qui peut être considéré comme un désavantage important.

6. Les ressources partagées dans le temps et dans l'espace

6.1. Les ressources à espace partagé (Space-Shared)

Dans la politique de planification Space Shared, le planificateur (Broker) planifie une tache sur la machine virtuelle concernée à un instant donné et après son achèvement, il lance une autre tâche sur la machine virtuelle. Cette même politique est utilisée pour programmer les machines virtuelles sur l'hôte.

6.2. Les ressources à temps partagé (Time-Shared)

Dans la politique de planification en temps partagé, le planificateur planifie toutes les taches sur la machine virtuelle en même temps. Il partage le temps entre toutes les taches et les planifie simultanément sur la machine virtuelle. Cette politique est également utilisée pour planifier la machine virtuelle sur l'hôte.

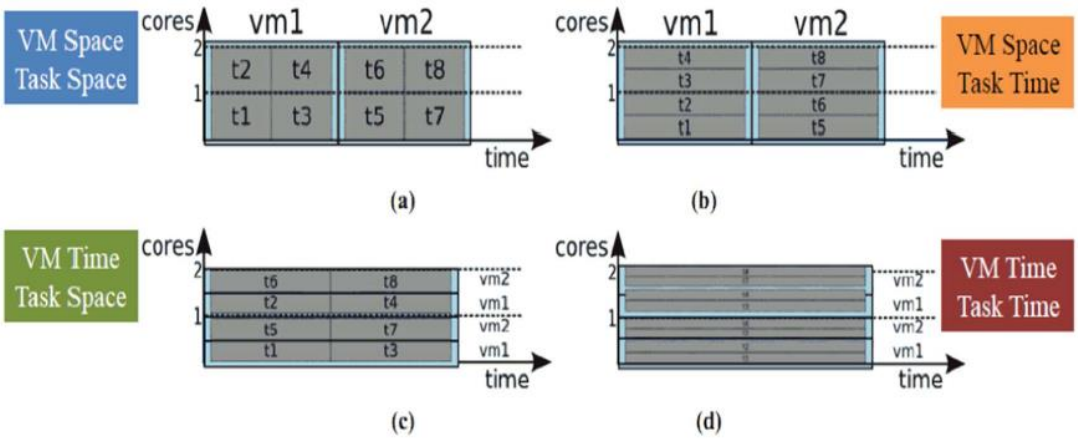

Figure 2.5 Les ressources partagées dans le temps et dans l'espace

7. Conclusion

Dans ce chapitre nous avons détaillé le sujet de ce mémoire qu'il s'agit d'allocation des ressources et de planification des taches. Nous avons montré le lien assez fort entre une ressource et une tache en déduisant qu'on ne peut jamais parler d'une planification sans exprimer la ressource. Pour cette dernière, nous avons développés un ensemble des solutions, heuristiques et méta-heuristique

Après avoir présenté notre sujet d'étude, nous allons détailler la phase d'implémentation et d'expérimentation dans le chapitre suivant.

Chapitre III : Réalisation& Expérimentation

1. Introduction

Ce chapitre est consacré à la phase pratique; reflète la réalisation de l'ensemble des solutions adoptée. En effet, nous avons incorporées quatre solutions de planification (FCFS, PSO, RR et SJF) en considèrent la ressource de calcule (CPU) pour estimer la quantité d'énergie consommée par cette ressource. L'objective de nos prépositions est de connaitre le meilleur algorithme parmi ces quatre pour réduire la consommation d'énergie

2. les plateformes et les outils de développement :

2.1. Plateforme de Simulation : CloudSim

CloudSim est un Framework de simulation généralisé et extensible qui permet la modélisation, la simulation et l'expérimentation des nouvelles infrastructures d'infonuagiques et des services d'application associés. Nous avons utilisé pour la réalisation de nos travaux de thèse la version du simulateur CloudSim 3.0.3.

Le simulateur CloudSim est composé de plusieurs classes. Parmi les classes fondamentales qui forment les blocs constitutifs du simulateur CloudSim. Nous pouvons citer :

Datacenter : La classe Datacenter permet de modéliser le cœur de l'infrastructure du Cloud. elle encapsule un ensemble de machines physiques appelées Hosts qui se caractérisent par leurs configurations (mémoire, CPU, stockage et nombre de cœurs). Chaque Datacenter implémente un ensemble d'algorithmes pour l'allocation de la bande passante, la mémoire et le stockage aux différents hosts et machines virtuelles du Cloud.

Cloudlet : La classe Cloudlet modélise les applications. Elle a un nombre d'instructions et de données connu à exécuter et à transférer. A noter que cette classe est étendue par WorkflowSim en Task puis en Job pour modéliser les relations de dépendances entre les tâches.

Datacenter Broker : Cette classe modélise le courtier (Broker), qui est responsable de la médiation entre les utilisateurs et les prestataires de service selon les conditions de QoS des utilisateurs

et elle permet de déployer les tâches de service à travers les Clouds. Le Broker agit au nom des utilisateurs, il identifie les prestataires de service appropriés du Cloud par le service d'information du Cloud CIS (Cloud Information Services) et négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS1 des utilisateurs.

Machine virtuelle : Cette classe modélise une instance de la machine virtuelle (VM), dont la gestion pendant son cycle de vie, est une responsabilité de la machine (Host). Un Host peut simultanément instancier de multiples VMS et assigner des cœurs à base des politiques prédéfinies de partage de processeur espace partagé ou temps partagé.

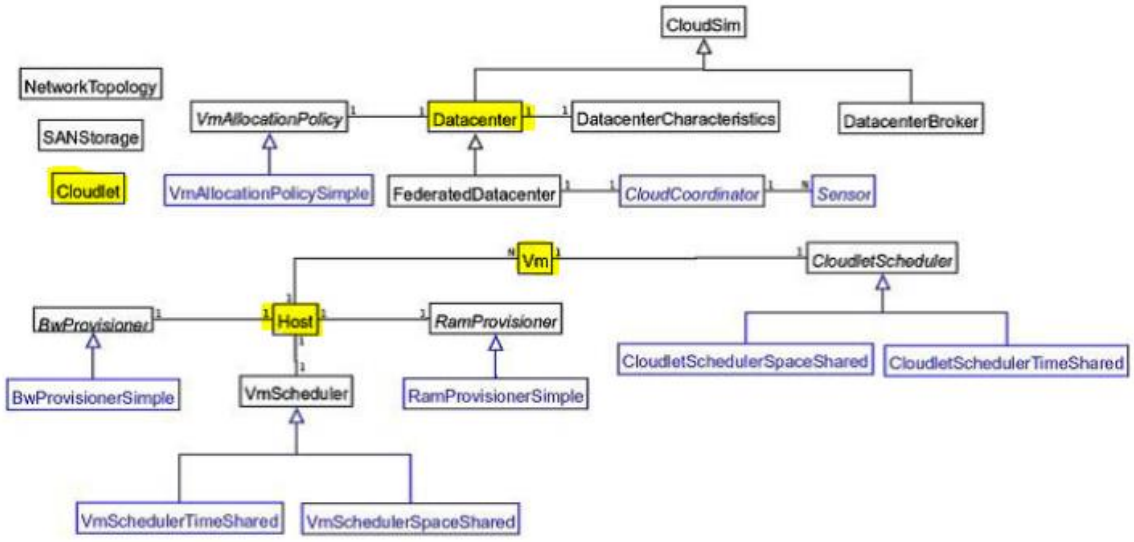

Figure 0.1 Les Principales Classes de CloudSim.

2.2. Environnement de développement :- 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. Sun Microsystems a fondée projet open source NetBeans en Juin 2000 et continue d'être le sponsor principal du projet [29].

2.3. Le langage de programmation Java

Le langage Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. Créé à ses débuts pour s'exécuter surtout sur environnements hétéroclites (embarqués principalement), son succès est surtout dû à son intégration aux navigateurs internet grand public [29].

Java à la particularité principale d'être portable, c'est-à-dire, que les logiciels écrits avec ce dernier sont très facilement réutilisables sur plusieurs systèmes d'exploitation tels qu'Unix, Microsoft Windows, Mac OS, ou linux avec un peu ou pas de modification. C'est la plateforme qui garantit la portabilité des applications développées en Java. [28].

NB: Nous avons réalisé ce travail sur une machine avec un processeur Intel® Core™ i3-3110 M CPU @ 2.40GHZ, doté d'une capacité mémoire 4GB, sous Windows 7 Professionnel de 64 bits. Nous avons utilisé le langage de programmation Java, les environnements de développement Netbeans version 8.0.1, et le simulateur CloudSim version 3.0.3.

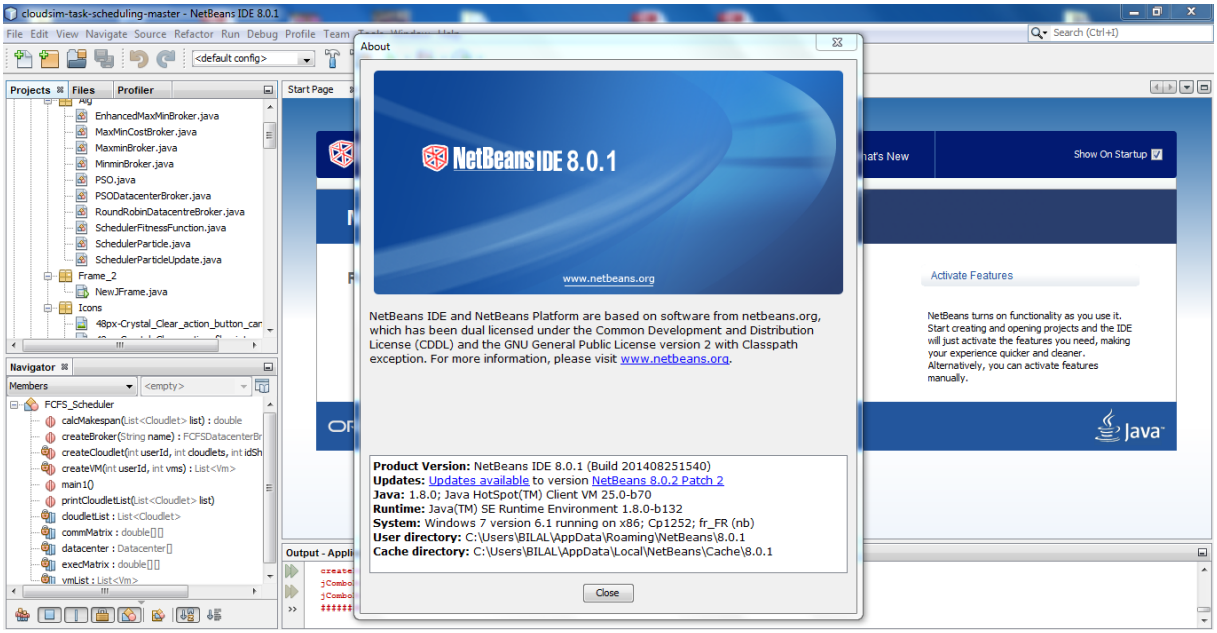

Figure 3.1 L'environnement de developpement Netbeans

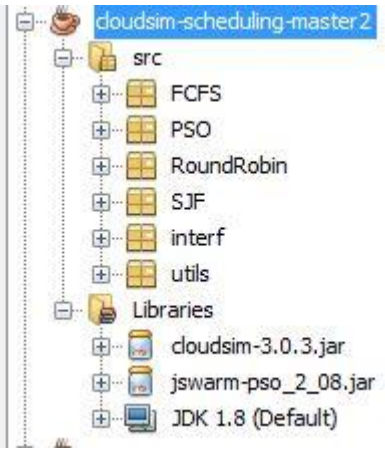

Figure 3.2 La structure de notre application

3. Implémentation

3.1. Description de l'application

La version CloudSim 3.0.3 n'a pas d'interface graphique, il utilise le mode console pendant l'exécution, qui rend difficile à l'utilisateur de profiter pleinement du CloudSim, comme la modification des paramètres de simulation pour l'affichage graphique des outputs (résultat de simulation)... Nous avons crée une interface graphique qui facilite l'accès et la manipulation du simulateur, nous allons dévoiler les différentes étapes procédées pour réaliser notre application.

3.1.1. Interface principale

Dès le lancement de notre application, la figure (3.3) apparait en premier aux utilisateurs, elle est constituée d'une barre de menu contenant les menus suivants:

Scheduler, Modifier les paramètres; et Aide, chaque menu contient un ensemble d'items, nous allons mettre l'accent dans cette partie à la démonstration de notre application via un exemple en faisant référence à quelques interfaces graphique.

Figure3.3 L'interface Principale

3.1.2. Configuration de Datacenter :

Les paramètres par défaut sont déjà définis. Cette étape consiste à modifier les paramètres nécessaires voulus comme le nombre de Datacenter, la taille de la mémoire, le stockage, la bande passant et le système d'exploitation.

Lors de la configuration des Datacenter ils doivent être respectés la valeur appropriée de chaque paramètre.

La taille de RAM, il faut prendre le nombre de 256 et de ses complications (host
Memory (MB)).
Pour le Storage représente la taille de stockage de l'host, de 10000 et plus (host
Storage (MB)).
La bande passante BW à offrir prend le nombre 1000 et plus (BW = 1000 et plus).

Figure 3.4 Configuration de Datacenter selon les besoins.

3.1.3. Configuration de Virtuelle Machine

Cette étape consiste à modifier les paramètres nécessaires voulus : la définition de nombre de machine virtuelle VM, et le CPU dans une VM, la spécification de la taille RAM de la VM, la taille de disque dure, le nombre de Core, la bande passante et le système d'exploitation.

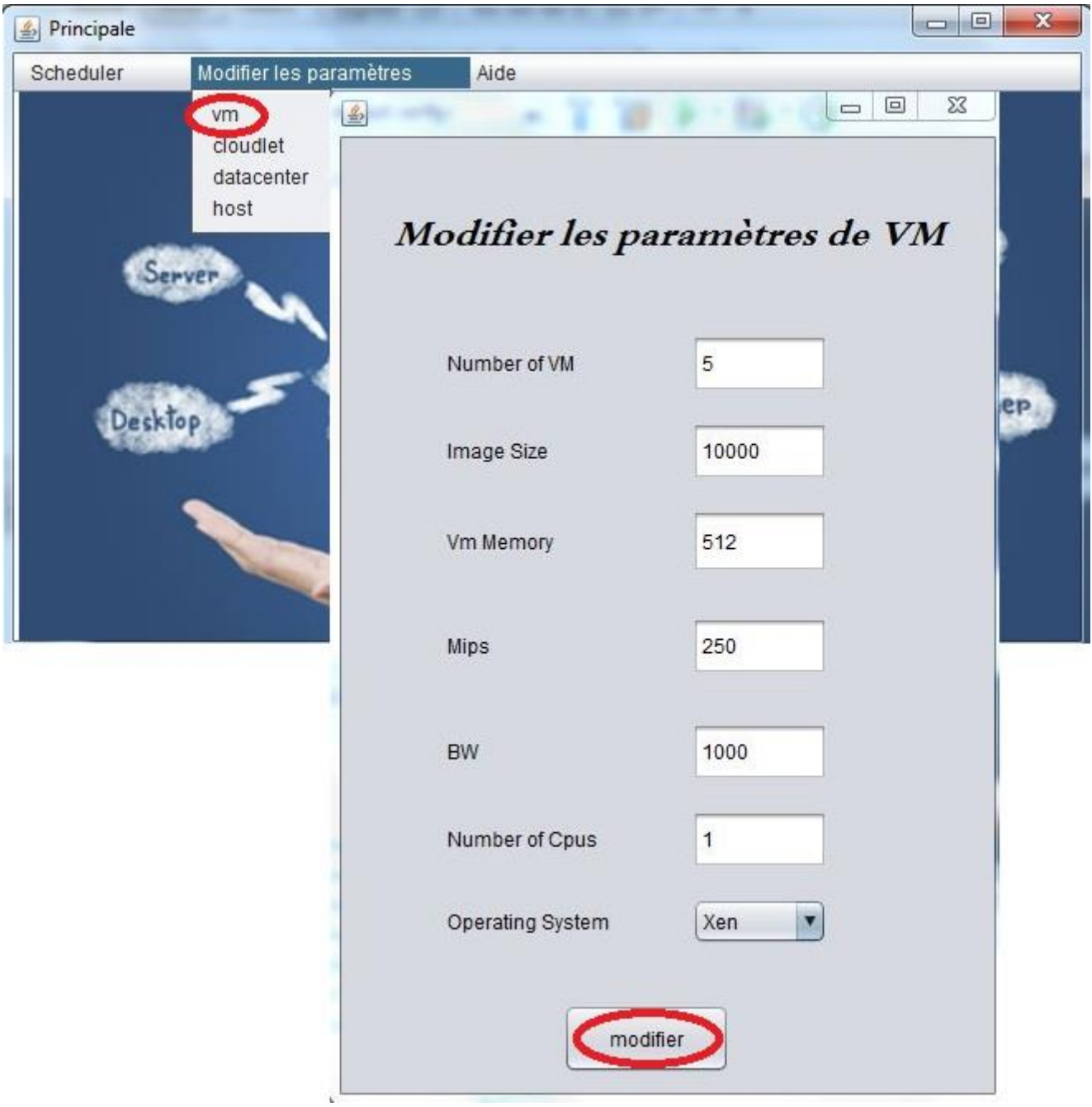

Figure 3.5 Configuration du Virtuelle Machine

La configuration des machines virtuelles représente l'affectation de chaque paramètre par des valeurs concernées.

Number of VM: représente le nombre des virtuelles machines dans un Datacenter.

Image Size : est la taille d'une image qui prend 1000 et plus (Image Size(MB)).

VM memory : représente la RAM des machines virtuelles, RAM = 512 et plus (MB)

Mips : représente le nombre d'instruction exécuté par seconde, c'est-à-dire combien de million d'instructions par seconde, MIPS prendre la valeur mips = 250 et plus selon l'exigence d'application.

BW : la bande passante BW demandé par la VM, elle prend le nombre 1000 et plus.

3.1.4. Configuration de Cloudlets (Taches)

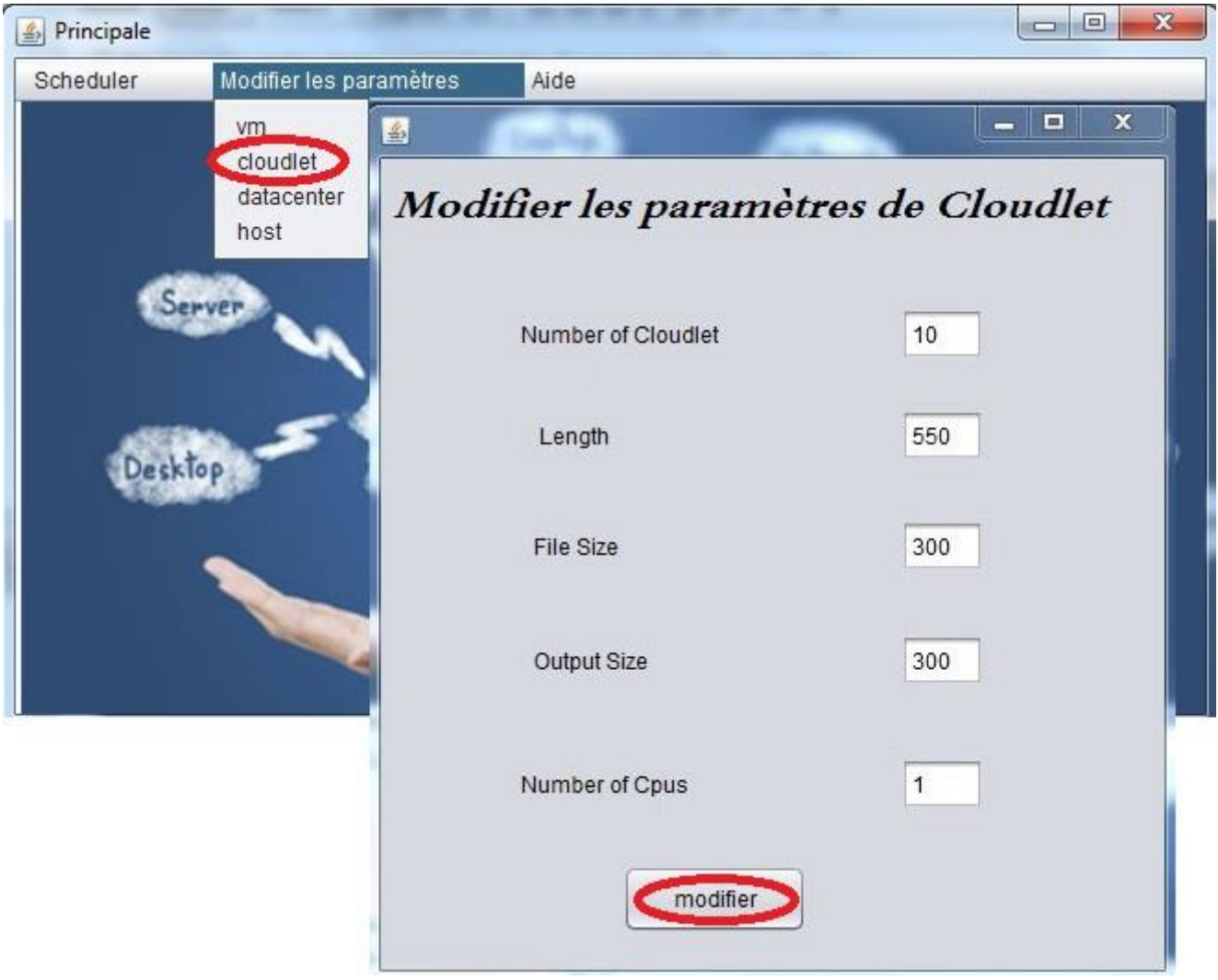

Figure 3.6 Configuration du Cloudlets

Dans cet te étape, on travaille avec les Cloudlets telle qu'on définit. Les caractéristiques des taches octroyées par Cloudlet. Cela signifie que vous pouvez constituer le comportement

du Cloudlet : Longueur Cloudlet (paramétré par MIPS- qui influent sur le temps d'exécution de la tâche dans une machine virtuelle), fichier d'entrée et de sortie (que d'impact sur des simulations avec le stockage), le modèle d'utilisation (qui définit quelle comportement que vous tâche fera lorsque son utilisation PSe (le nombre de CPU), mémoire et bande passante. (La figure 3.7 montre les paramètres de configuration de Cloudlet).

Number of Cloudlet: est un nombre entier positif.

Length: la taille de la Cloudlet à exécuté dans le CloudRessource (exemple long length = 500).

File size : la taille de fichier d'entrée du Cloudlet avant l'exécution (la taille du programme + les données en entrée), qui prend un nombre entier positive (exemple long file Size = 300).

Output size : la taille du fichier de sortie de la Cloudlet après l'exécution (exemple long file Size = 300).

3.1.5. Lancement de la simulation

Une fois les paramètres de simulation introduite, l'utilisateur peut lancer la simulation à partir du menu Sheduler en choisissant l'item qui correspond à la solution voulu (FCFS, PSO, RR ou bien SJF).

Quand on favorise la solution désirée (dans notre exemple PSO), elle va nous apparaitre une matrice vide, quand on clique sur valider elle va se remplir par les informations suivantes : Cloudlet ID, STATUS, Data Center ID, VM ID, Time, Start Time, Finish Time, et le plus important dans notre thèse est le ressource d'énergie, celui qui va nous aider à déterminer la meilleure solution (voir la figure 3.7).

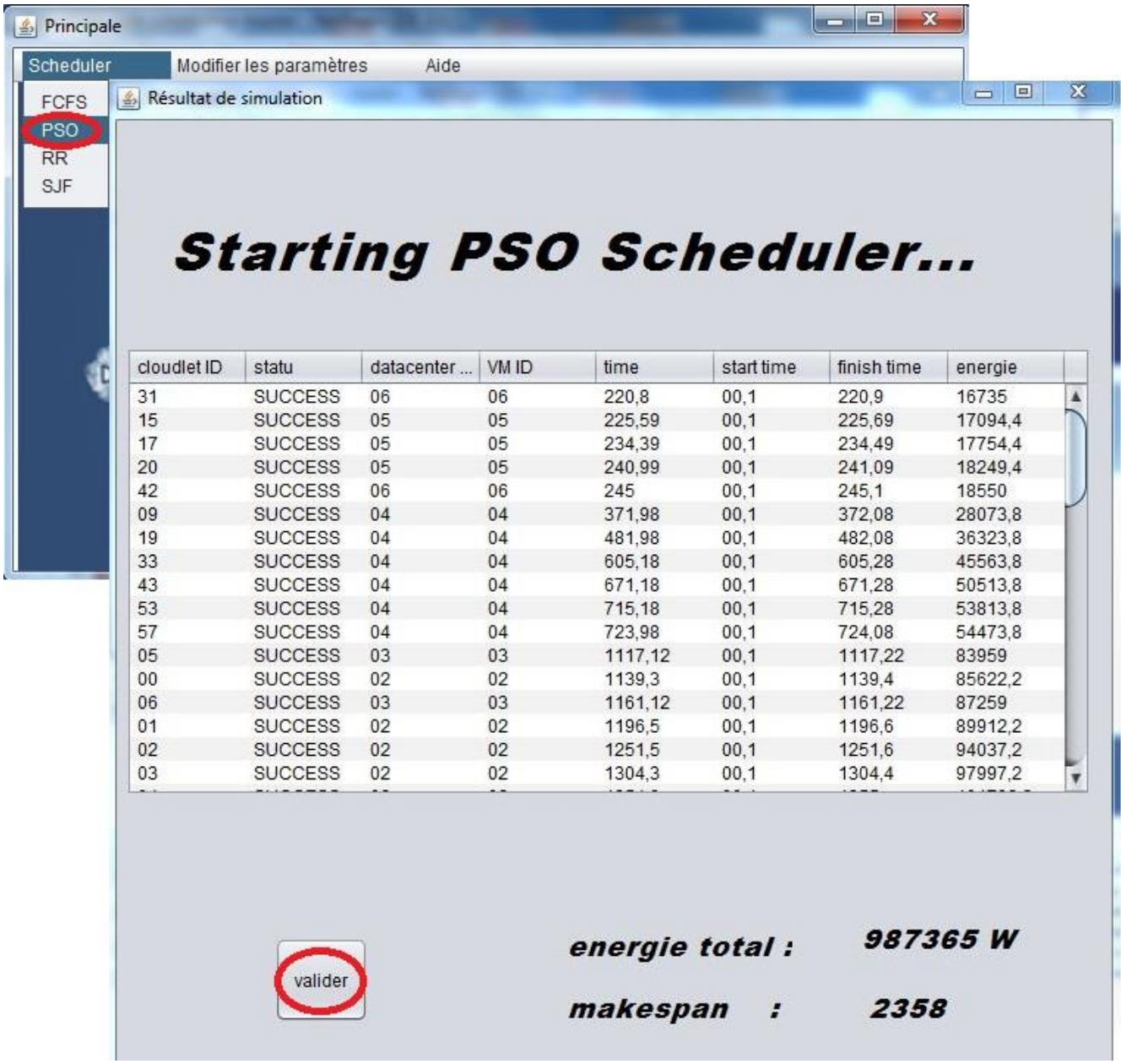

Figure 3.7 Lancement de la simulation

3.2. Analyse et Discussion

Nous allons présenter dans cette partie les résultats des différentes expérimentations que nous avons obtenus en appliquant le type de ressource partagée dans le temps pour les solutions que nous avons améliorés en ajoutant le facteur d'énergie et l'adapté dans le CloudSim. Pour effectuer les différentes séries d'expérimentations de notre application, nous avons fixé un certain nombre de paramètres de simulation et varié d'autres, dans le but d'étudier, comparer l'évolution de chaque solution et de déduire la meilleure entre eux.

Dans notre étude, nous avons fixé le nombre de Data Center à 5, le nombre de VM à 5, le nombre machine physique (host) à 1, puis nous avons varié le nombre des Cloudlets (30, 40, 50), et ses tailles (une fois le nombre de Cloudlet augmente (n), la taille de Cloudlet length(n) elle va s'augmenter selon la formule suivante : length(n) = 550*n, tel que n ∈ à l'ensemble des nombres entiers N), après on va calculer l'énergie consommée par chaque VM et pour chaque solution par la formule suivante: $E = \frac{\sum_{j=1}^{n} P(t)j}{m}$ telle que

E est l'énergie consommée

M est le nombre de host (machine physique dans un Data Center).

P(t) : la puissance par rapport à un utilisateur en fonction de temps, on calcule la puissance par : $P(t) = P_{max} * (0.7+0.3t)$ (P_{max}=250w pour des serveurs moderne)

T est le temps de traitement de chaque Cloudlet.

Tant que le nombre de host est 1 alors l'énergie E = $\sum_{j=1}^{m} p(t)j$

Dans notre travail on va faire trois scénarios en adoptant la formule d'énergie suivante :
E = $\sum_{j=1}^{m} p(t)j$

3.2.1. Influence de 30 taches "Cloudlets"

La première série a pour objectif d'étudier l'impact de 30 Cloudlets dans chaque solution et de calculer l'énergie consommée par chaque VM ainsi l'énergie totale. Le tableau suivant montre combien de Cloudlets exécutés par une VM et l'énergie consommée par elle sachant que le temps d'exécution de Cloudlet est différent à l'autre (c'est-à-dire si le temps

d'exécution de 4 Cloudlets pour la VM1 est X ***ms*** ça veut pas dire que le temps d'exécution de 4 Cloudlet pour la VM2 ou VM3 … VM5 est le même), et l'énergie consommée par toutes les VMs ce qui est consommé par des solutions.

Solutions	VMs	Cloudlet exécuté	énergie	énergie Totale
FCFS	1	3	5968,5	404668,2
	2	3	20489,1	
	3	8	131243,4	
	4	7	95100,7	
	5	9	151866,5	
PSO	1	3	23962,6	463434,3
	2	4	37648,2	
	3	11	261626,6	
	4	2	3485	
	5	10	136711,9	
RR	1	4	23137,6	418343,1
	2	5	37174,1	
	3	3	30015	
	4	9	220874	
	5	9	107142,4	
SJF	1	3	29840	423861,9
	2	10	229815	
	3	6	66777,4	
	4	7	68856,3	
	5	4	28573,2	

Tableau 3.1 Comparaison des solutions pour 30 Cloudlets.

On remarque dans le tableau 3.1 que le nombre maximum de Cloudlets exécutés par les VMs dans la 2ème solution du tableau (PSO) est le plus grand et consomme plus d'énergie, et le nombre maximum de Cloudlets traité par la 1er solution du tableau (FCFS) est le plus petit et consomme moins d'énergie, dans ce cas nous distinguons que la solution qui possède la machine virtuelle qu'elle traite plusieurs Cloudlets Consomme beaucoup d'énergie et par conséquent augmente l'énergie totale.

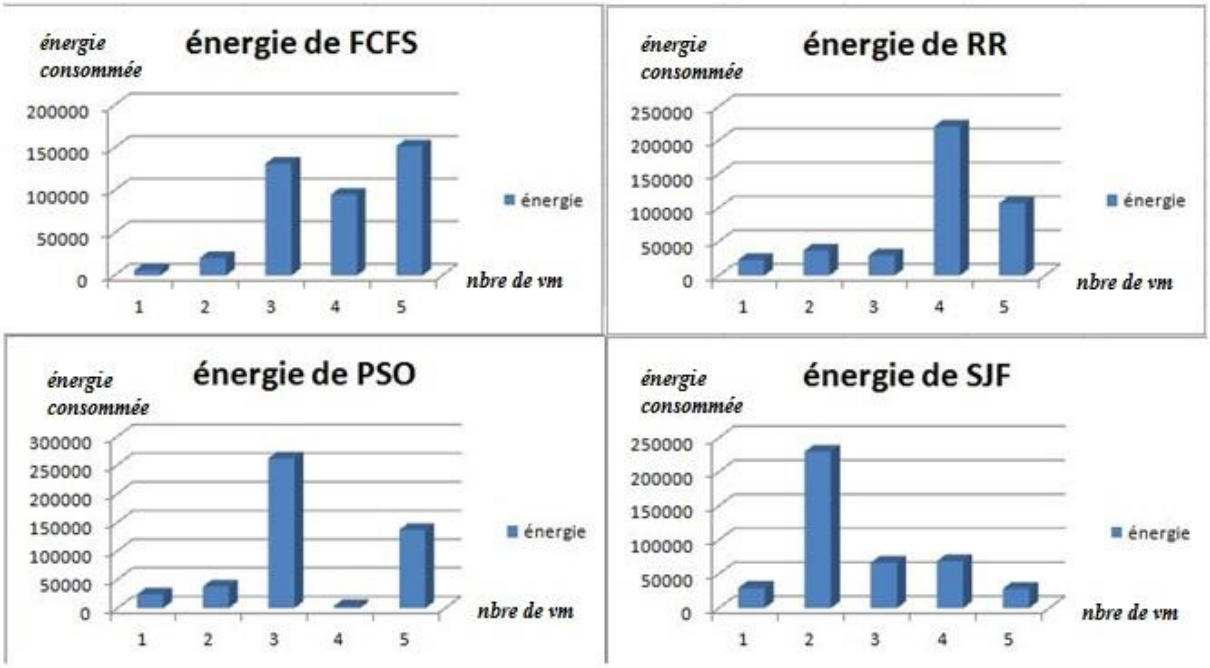

Figure 3.8 La consommation d'énergie par les 4 solutions (30 Cloudlets)

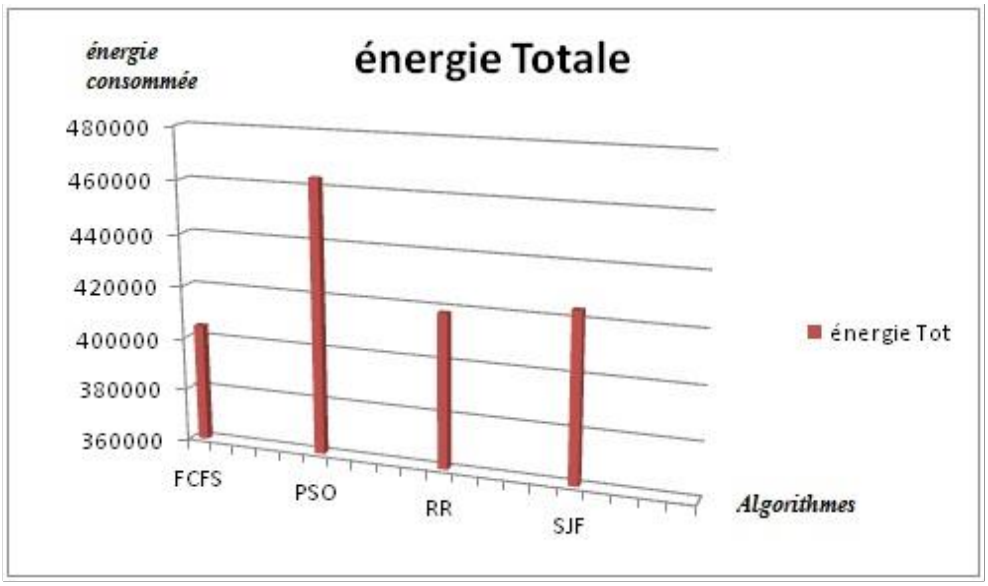

Figure 3.9 La Consommation d'énergie totale (30 Cloudlets)

3.2.2. Influence de 40 taches "Cloudlets"

Cette deuxième série à pour objectif d'étudier l'effet de 40 Cloudlets dans chaque solution et de calculer l'énergie consommée par chaque VM ainsi l'énergie totale. Le tableau suivant montre la même chose que le tableau précédant.

Solutions	VMs	Cloudlet exécuté	énergie	énergie Totale
FCFS	1	8	195284,35	799271,2
	2	6	99711,6	
	3	12	243135	
	4	10	202032,65	
	5	4	59107,6	
PSO	1	7	77277,4	925301
	2	9	110978,4	
	3	7	88827,4	
	4	3	15373,2	
	5	14	632844,6	
RR	1	11	280134,7	1167796
	2	2	17510	
	3	4	39564,1	
	4	8	279060,6	
	5	15	551526,6	
SJF	1	3	20158,2	945414,7
	2	7	126777,4	
	3	14	414231,3	
	4	12	335369,6	
	5	4	48878,2	

Tableau 3.2 Comparaison des solutions pour 40 Cloudlets

On remarque dans ce tableau que le nombre maximum de Cloudlets exécutés par les VMs dans la 3ème solution du tableau (RR) est le plus grand et consomme plus d'énergie, et le nombre maximum de Cloudlets traité par la 1er solution du tableau (FCFS) est le plus petit et consomme moins d'énergie, dans ce cas nous voyons que la solution qui a la machine virtuelle qu'elle traite plusieurs Cloudlets Consomme beaucoup d'énergie et augmente l'énergie totale.

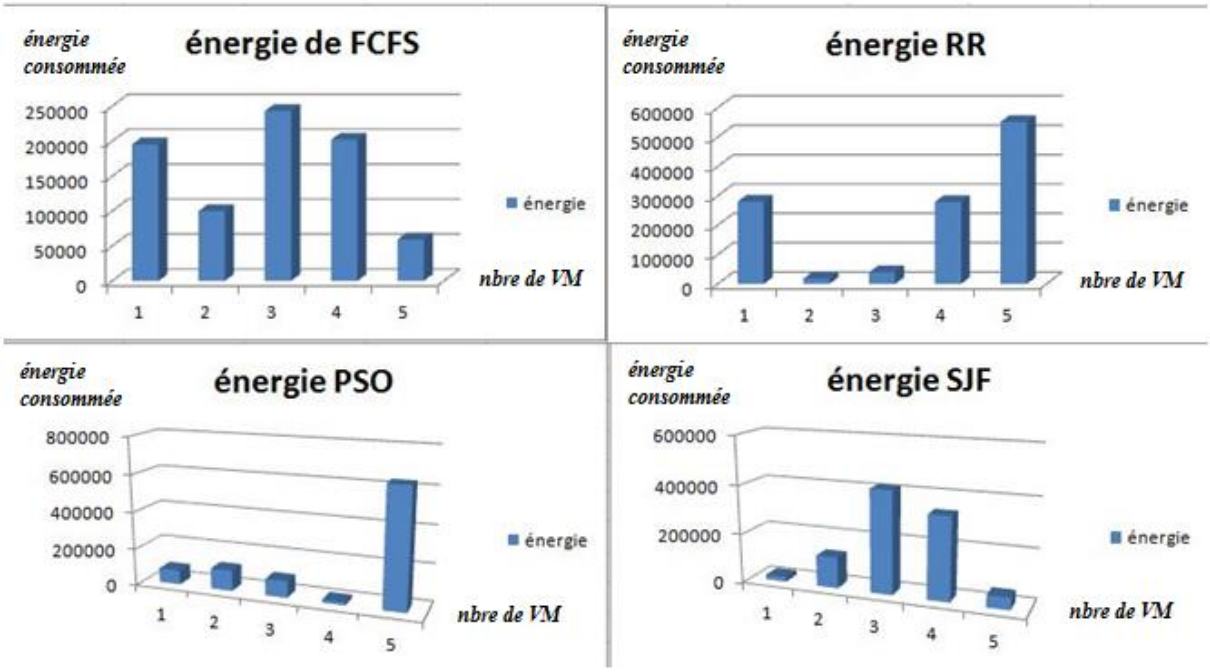

Figure 3.10 La consommation d'énergie par les 4 solutions (40 Cloudlets)

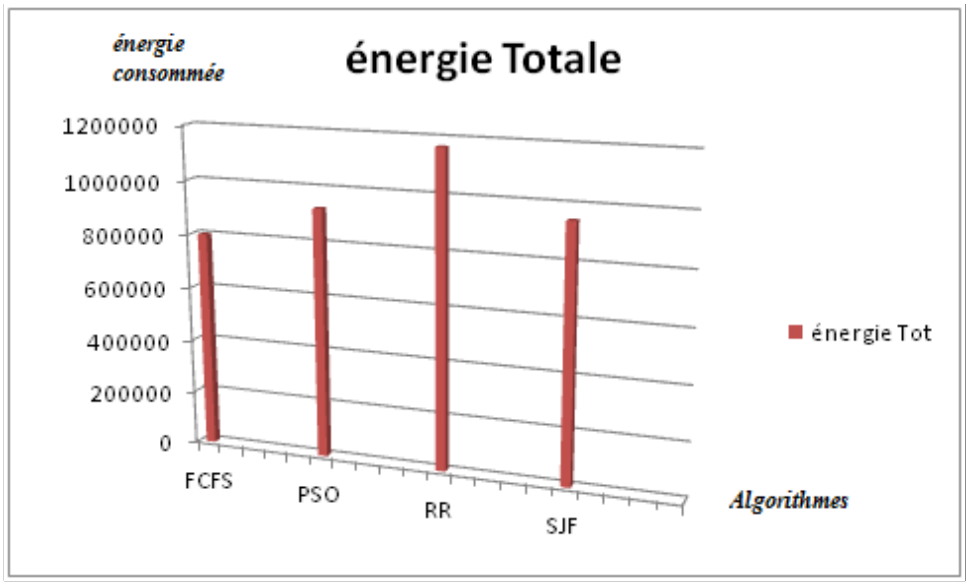

Figure 3.11 La Consommation d'énergie totale (40 Cloudlets)

3.2.3. Influence de 50 taches "Cloudlet".

La troisième série a pour objectif d'étudier l'impact de 50 Cloudlets pour chaque solution et de calculer l'énergie consommée par chaque VM ainsi l'énergie totale. Le tableau suivant vous montre la même chose que les tableaux précédents.

Solutions	VMs	Cloudlet exécuté	énergie	énergie Totale
FCFS	1	10	221032,45	1483640,9
	2	9	285210,3	
	3	8	194779,85	
	4	12	399397,2	
	5	11	383221,1	
PSO	1	7	120672,4	1533043,5
	2	9	217700,6	
	3	11	305018,9	
	4	8	136853,4	
	5	15	752798,2	
RR	1	9	263248,8	1661449,8
	2	10	308470,6	
	3	16	751192,4	
	4	8	130250,6	
	5	7	208287,4	
SJF	1	10	349719,7	1523529,8
	2	10	181403,4	
	3	11	410784,5	
	4	9	148898,4	
	5	12	432723,8	

Tableau 3.3 Comparaison des solutions pour 50 Cloudlets

On remarque dans ce tableau que le nombre maximum de Cloudlets exécutés par les VMs dans la 3ème solution du tableau (RR) est le plus grand et consomme plus d'énergie, et le nombre maximum de Cloudlets traité par la 1er solution du tableau (FCFS) est le plus petit et consomme moins d'énergie, dans ce cas nous voyons que la solution qui a la machine virtuelle qu'elle traite plusieurs Cloudlets Consomme beaucoup d'énergie et augmente l'énergie totale.

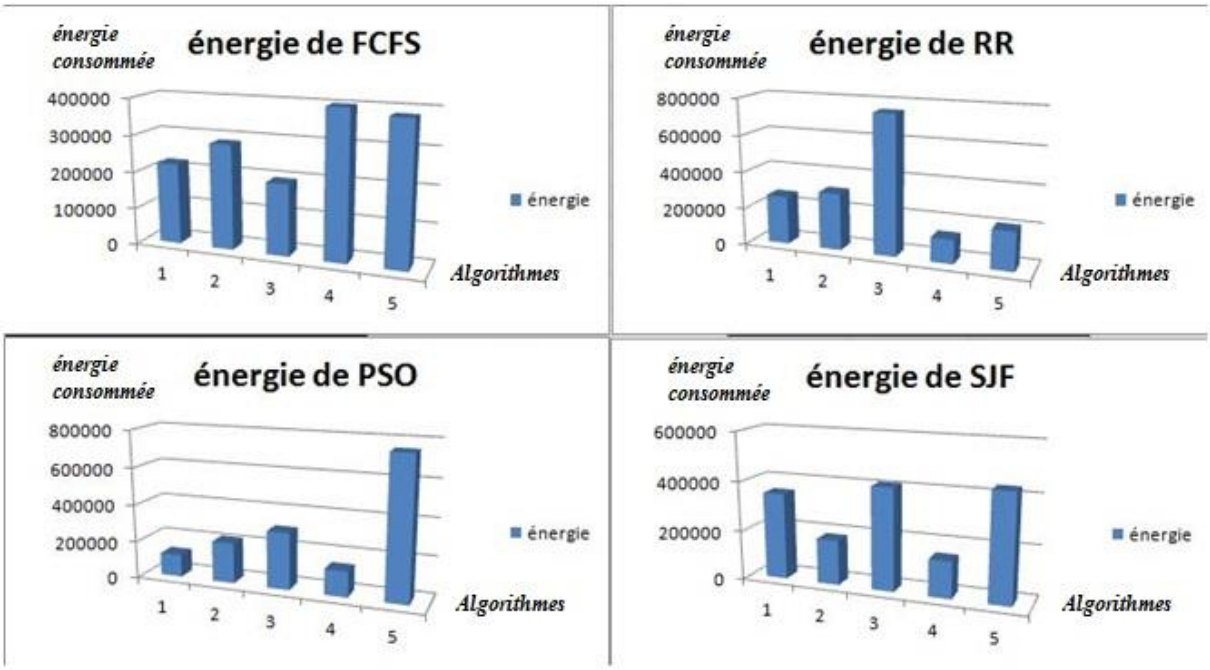

Figure 3.12 La consommation d'énergie par les 4 solutions (50 Cloudlets)

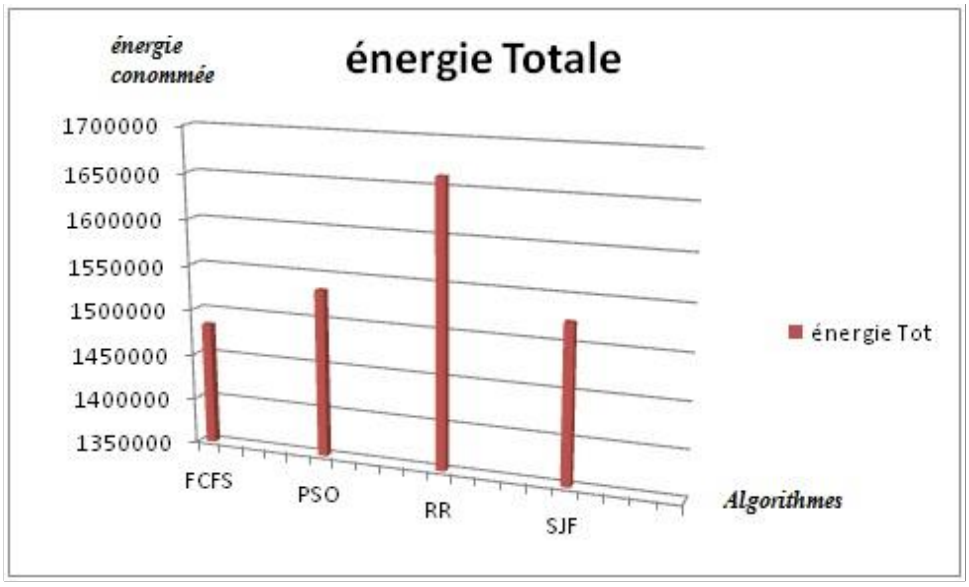

Figure 3.13 La Consommation d'énergie totale (50 Cloudlets)

Voici une comparaison finale des solutions implémentées dans ce travail.

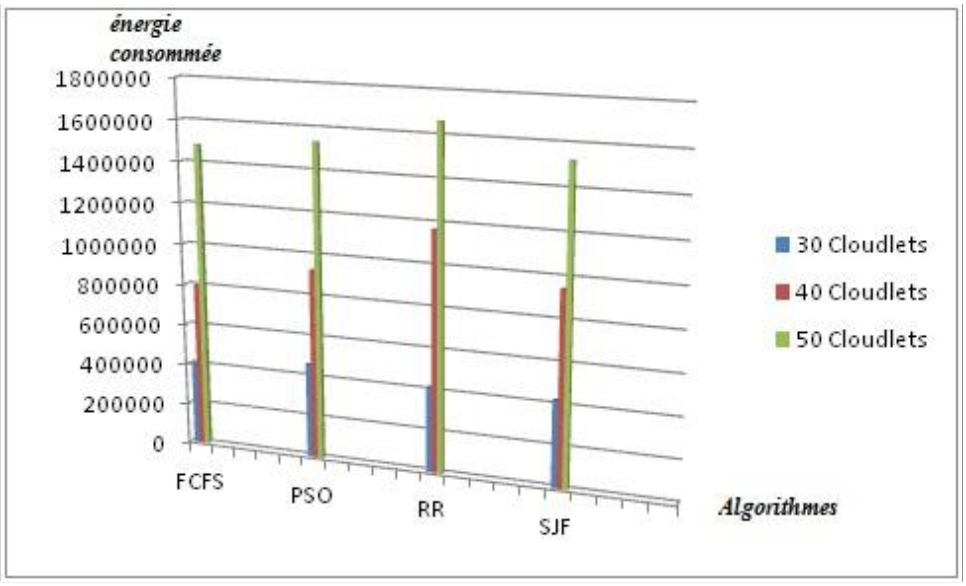

Figure 3.14 L'énergie consommée par les 4 solutions pour 30,40 et 50 Cloudlets

3.3. Synthèse

Dans cette section nous avons implémenté quatre solutions pour la planification des taches (FCFS, PSO, RR et SJF) en considérant la ressource énergétique dépensée par le CPU, dans le but d'avoir la meilleure entre eux. i.e qui consomme moins d'énergie.

Selon les scénarios précédents nous avons déduit que la meilleure solution qui consomme moins d'énergie est la 1er solution (FCFS (First Come First Serve)) en temps partagé en raison de sa simplicité et parce qu'il minimise le temps moyen que chaque Cloudlet doit attendre jusqu'à ce que son exécution soit terminé.

4. Conclusion

Nous avons présenté dans ce chapitre, les outils de développement utilisées, les algorithmes de planification et la ressources favorisée dans notre réalisation ainsi que l'application développés avec les différentes solutions distinguées et les divers scenarios d'expérimentation appliqués surtout sur les taches exécutées par les machines virtuelles.

Conclusion Général & Perspectives

En conclusion, le sujet abordé et le projet abordé dans ce manuscrit consiste à réaliser une application pour examiner et minimiser l'énergie associée à la gestion des ressources matérielles et logiciels. Rappelant que l'objectif de ce travail était de concevoir une application pour aider les entreprises à bien choisir entre les solutions reconnus et disponibles de la planification des taches. Pour cela, nous avons réalisé une application interactive permettant d'aider et de satisfaire les besoins des entreprises ainsi que les utilisateurs de domaine.

Notre travail est débuté par un chapitre introductifs sur les concepts d'infonuagique et ses caractéristiques aussi ses services.

Ultérieurement, nous avons réalisé dans le deuxième chapitre des définitions sur les mots clés qu'on a déjà utilisés concernant notre sujet d'étude, ensuite un état de l'art sur l'allocation et la planification ce qui est notre objectif, à ce stade on a fixé nos œil surtout sur les solutions de planification, puis on a cité un aperçu général sur les stratégies d'allocation et les types de ressources partagées.

Enfin, et à travers le dernier chapitre nous avons effectué la réalisation de cette application à travers la plateforme NetBeans et le Framework CloudSim.

Par ailleurs, la moindre des choses on doit dire est que ce projet de fin d'étude n'été qu'une source de bénéfices tant qu'au niveau technique qu'au niveau personnel et relationnel. En effet, un aspect important dans notre expérience était l'esprit d'équipe, ceci nous a pris qu'un problème ne peut être résolue sans la singerie des compétences. On a eux aussi l'occasion de côtoyer un domaine nouvel pour nous qu'il s'agit d'une plateforme de développement CloudSim dans l'environnement infonuagique et d'approfondir nos compétences en temps que des étudiants sur le système d'information des entreprises.

En perspective, notre application peut être améliorée en ajoutant d'autre fonctionnalité comme le cout d'énergie consommée.

References Bibliographique

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and emerging it platforms : Vision, hype, and realty for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp.599-616, 2009.

[2] R.Sinha, N. Purohit, and H.Diwanji, "Power aware live migration for data centers in cloud computing using dynamic threshold, "International Journal of Computer Technology and Applications, vol. 2, no. 6, pp. 2041-2046, 2011.

[3] Ian T .Foster, Yong Zhao, Ioan Raicu, and Shiyoung Lu. Cloud computing and grid computing 360-degree compared. *CoRR*, abs/0901.0131, 2009.

[4] Sheuti Chhabra and V. S. Dixit. Cloud computing : State of the art and security issues. *SIGSOFT Softw. Eng. Notes*, 40(2): 1-11, April 2015.

[5] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Linder. A break in the clouds : Towards a cloud definition. *SIGWOMM Comput. Commun. Rev*., 39(1) ;50-55, December 2008.

[6] A. Ohri. R for cloud computing : An Approach for data scientist. Springer, New York Heidelberg Dordrecht London, 2014.

[9] Teng F., "Resource Allocation and Scheduling Models for Cloud Computing", tel00659303, version 1- 12, 2012.

[11] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh, and Christopher Mcdermid. "availability and load balancing in cloud computing. In Internatio-Bibliographie nal conference on computer and Software Modeling, IPCSIT11, 2011.

[13] Romain Hennion, Hubert Tournier, and Eric Bourgeois. Cloud computing : Décider, Concevoir, Piloter, Améliorer. Groupe Eyrolles, 2012

[14] Lim S-H, Sharma B, Nam G, Kim EK, Das CR (2009) MDCSim: a multi-tier data center simulation, platform. In: IEEE international conference on cluster computing and workshops (CLUSTER).

[16] Kliazovich D., Bouvry P., and Khan S., "Greencloud: a packet-level simulator of energyaware cloud computing data centers," The Journal of Supercomputing, 2010.

[17] Rothkopf, M., Pekeˇc, A., and Harstad, R. (1998). Computationally manageable combinatorial auctions. Management Science, 44:1131–1147.

[18] Wickremasinghe B., Calheiros R. and Buyya R. CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications. Proceedings of the

24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia, 20–23 April 2010; 446–452.

[20] Han Zhao and Xiaolin Li. Auctionnet: Market oriented task scheduling in heterogeneous distributed environments. In the International Parallel and Dis- tributed Processing Symposium (IPDPS), pages 1–4. IEEE, 2010.

[21] Young Choon Lee, Chen Wang, Albert Y. Zomaya, and Bing Bing Zhou. Profit- driven service request scheduling in clouds. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10, pages 15–24, Washington, DC, USA, 2010.

[22] M. Hemamalini. Review on grid task scheduling in distributed heterogeneous environment. International Journal of Computer Applications, 40(2) :24–30, 2012. (Cit´e en page7.)

[23] M. Hemamalini. Review on grid task scheduling in distributed heterogeneous environment. International Journal of Computer Applications, 40(2) :24–30, 2012.

[24] "RASA: A New Task Scheduling Algorithm in Grid Environment" Saeed Parsa, Reza Entezari-Maleki.

[25] "A Budget-Constrained Time and Reliability Optimization BAT Algorithm for Scheduling Workflow Applications in Clouds" Navneet Kaur, Sarbjeet Singh,*.

[26] "Analysis of Job Scheduling Algorithms in Cloud Computing" Rajveer Kaur, Supriya Kinger

[27] Franck Morvan and Abdelkader Hameurlain. Dynamic query optimization: towards decentralised methods. In the International Journal of Intelligent Information and Database Systems, vol. 3, issue 4, pages 461–482, decembre 2009.

[28] The Cloud Computing and Distributed Systems (CLOUDS Laboratory) Uni- versity of Melbourne. Cloudsim. http ://www.cloudbus.org/cloudsim/, (Consult´e Mars 2015)

[29] L. Christophe, 'De la nécessité d'une vision holistique du code pour l'analyse statique et la correction automatique des Applications Web ', thèse de doctorat, UNIVERSITÉ DE RENNES 1, 2011.

[30] B. Samah, 'Gestion des ressources dans le Cloud Computing à base des modèles économique ', Mémoire de magister, Université d'Oran, Faculté des sciences, 2011.

Web graphique

[7] Global Digital Vision. Cloud computing. http://www.gdv.com.an/cloudcomputing.html, (Consulté Mars 2014).

[8] http://www.hebergeurcloud.com. Hébergeur cloud. http://www.hebergeurcloud.com/les-technologies-du-cloud-computing/, (Consulté Mars 2015).

[10] https://en.wikipedia.org/wiki/Cloud_computing_security

[12] Le Cloud Kesako. Cloud-serveur. http://www.cloud-serveur.fr/fr/le-cloud/cloud-kesako, (Consulté Mars 2016).

[15] http://www.cloudbus.org/cloudsim/

[19] http://www.hebergeurcloud.com.Hébérgeurcloud.

http://www.hebergeurcloud.com/les-technologies-du-cloud-computing.

(Consulté Mars 2015).