**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université ABBAS LAGHROUR Khenchela**

**Faculté des Sciences et Technologies**

**Département mathématique et Informatique**

**Mémoire de fin d'étude pour l'obtention de diplôme**

**De master informatique**

***Option***

**Sécurité et Technologie Web**

# Thème

# Méthode D'équilibrage De Charge Basée Sur Les Systèmes Multi Agent dans le Cloud computing

Présenter par :
Zalfani Zakaria

Dirigé par :
Dr. Ouided Hioual

Devant le jury composé de :

| | |
|---|---|
| Dr. Sofiane HEMAM | MCA Université Abbes Laghrour Khenchela Président |
| Dr.Ouided HIOUAL | MCB Université Abbes Laghrour Khenchela Directrice |
| Mme. Otaila Chergui | MAA Université Abbes Laghrour Khenchela Examinatrice |

**Universitaire 2019/2020**

# Remerciement

Avant toute chose, je remercie Dieu.

Je tiens à exprimer mes vifs remerciements à

MLLE OUIDED HIOUAL, ma directrice de recherche

A toute l'équipe pédagogique STW, pour tout effort fourni durant notre période d'étude

Mes remerciements vont également à tous les enseignants, spécialement :

Mme OUASSILA HIOUAL,

MR HEMAM MOUNIR,

MR HEMAM SOFIANE

À qui je dois le respect et la reconnaissance, les membres de jury

Pour avoir accepté de juger mon travail

Ce mémoire n'aurait jamais pu voir le jour sans le soutien actif des amis proches que je viens vivement à les remercier.

Merci

# Dédicace

Aujourd'hui et après toutes ces années, J'ai l'honneur, mais surtout le plaisir de dédier ce travail à toutes les personnes qui m'aiment, qui croient en moi

À Ma très chère maman

A mon père

Que dieu les protège

A Mlle OUIDED HIOUAL

A Mes Frères et Sœurs

A tous mes amis et à tous ceux que j'aime

## Résumé

Dans le Cloud computing L'équilibrage de charge est une étape importante conditionnant les performances des applications parallèles. Dans le cas où la charge varie au cours de la simulation, il est important de redistribuer régulièrement la charge entre les différents processeurs. Dans ce contexte, il peut s'avérer pertinent d'adapter le nombre de processeurs au cours d'une simulation afin d'obtenir une meilleure efficacité, ou de continuer l'exécution quand toute la mémoire des ressources courantes est utilisée. Contrairement au cas où le nombre de processeurs ne varie pas, le rééquilibrage dynamique avec un nombre variable de processeurs est un problème. Dans ce travail, nous présentons une méthode pour équilibrer la charge dynamiquement entre les machines virtuelles basées sur des systèmes multi-agents et les algorithmes d'ordonnancement, Le modèle proposé se compose de quatre essentiels composants qui sont : l'agent de classement qui, est chargé de constituer trois classes des machines virtuelles (MV) de la plus performante à la moins performante classe des machines virtuelles, en utilisant pour cela une des techniques de classification supervisée. L'agent principale qui, en premier lieu, calcule la distance entre la tâche reçue et les centres de gravité des trois classes, puis il envoi la tâche à la classe la plus proche par rapport aux distances précédemment calculées. Nous avons un agent d'équilibrage local qui met à disposition trois agents budgétaires locaux, de manière décentralisée, chaque agent utilise une technique d'ordonnancement spécifique de manière centralisée dans chaque classe (R_Robin, SJF, FIFO). En fin l'agent d'équilibrage générale qui équilibre la charge entre les trois classes en utilisant l'algorithme TOPSIS



**Mots clé :** Cloud computing, systèmes multi agents, algorithmes d'ordonnancement.

## Abstract

In Cloud Computing Load balancing is an important stage conditioning the performance of parallel applications. In the event that the load varies during the simulation, it is important to regularly redistribute the load between the different processors. In this context, it may prove relevant to adapt the number of processors during a simulation in order to obtain better efficiency, or to continue execution when all the memory of the current resources is used. Unlike the case where the number of processors does not vary, dynamic rebalancing with a varying number of processors is a problem. In this work, we present a method to balance the load dynamically between virtual machines based on multi-agent systems and scheduling algorithms, the proposed model consists of four essential components which are: the ranking agent which, is responsible for constituting three classes of virtual machines (VM) from the most efficient to the least efficient class of virtual machines, using for this one of the supervised classification techniques. The main agent who first calculates the distance between the received task and the centers of gravity of the three classes, then sends the task to the nearest class based on the previously calculated distances We have a local balancing agent who provides three local budget agents, in a decentralized manner, each agent uses a specific scheduling technique centrally in each class (R_Robin, SJF, FIFO). Finally, the general balancing agent which balances the load between the three classes using the TOPSIS algorithm

**Keywords:** Cloud computing, multi agent systems, scheduling algorithms

**ملخص**

في الحوسبة السحابية، تعتبر موازنة الحمل مرحلة مهمة لتكييف أداء التطبيقات المتوازية. في حالة اختلاف الحمل أثناء المحاكاة، من المهم إعادة توزيع الحمل بانتظام بين المعالجات المختلفة. في هذا السياق، قد يكون من المناسب تكييف عدد المعالجات أثناء المحاكاة من أجل الحصول على كفاءة أفضل، أو لمواصلة التنفيذ عند استخدام كل ذاكرة الموارد الحالية. على عكس الحالة التي لا يختلف فيها عدد المعالجات، فإن إعادة التوازن الديناميكي مع عدد متفاوت من المعالجات يمثل مشكلة. في هذا العمل، نقدم طريقة لتحقيق التوازن الديناميكي للحمل بين الأجهزة الافتراضية على أساس أنظمة متعددة الوكلاء وخوارزميات الجدولة، ويتكون النموذج المقترح من أربعة مكونات أساسية وهي: وكيل التصنيف، مسؤول عن تكوين ثلاث فئات من الأجهزة الافتراضية(MV) من فئة الأجهزة الافتراضية الأكثر كفاءة إلى أقلها كفاءة، وذلك باستخدام إحدى تقنيات التصنيف الخاضعة للإشراف. الوكيل الرئيسي، الذي يحسب أولاً المسافة بين المهمة المستلمة ومراكز الجاذبية للفئات الثلاث، ثم يرسل المهمة إلى أقرب فئة بناءً على المسافات المحسوبة مسبقًا. لدينا وكيل موازنة محلي يوفر ثلاثة وكلاء موازنة محليين، بطريقة لا مركزية، يستخدم كل وكيل تقنية جدولة محددة بطريقة مركزية في كل فئة (R_Robin، SJF، FIFO). أخيرًا، وكيل الموازنة العام الذي يوازن الحمل بين الفئات الثلاث باستخدام خوارزمية TOPSIS .

**كلمات مفتاحية:** الحوسبة السحابية، الأنظمة متعددة الوكلاء، خوارزميات الجدولة.

# TABLE DES MATIÈRES

## Table des matières :

# TABLE DES MATIÈRES

# TABLE DES MATIÈRES



**Chapitre 04 : Conception De La Méthode Proposée**

# TABLE DES MATIÈRES

# LISTE DES FIGURES

## Liste des figures :

# LISTE DES FIGURES

# LISTE DES TABLEAUX

## Liste des tableaux :

# INTRODUCTION GÉNÉRAL

## Introduction général :

Le Cloud computing est un modèle informatique basé sur l'internet. Selon NIST « National Institute of Standards and Technology », le Cloud computing est un modèle permettant d'établir un accès à la demande en réseau vers un bassin partagé de ressources informatiques configurable. Ces ressources peuvent être : des réseaux, des serveurs, de l'espace de stockage, des applications et des services. Elles peuvent être approvisionnées rapidement avec un minimum d'effort de gestion et d'interaction avec le fournisseur de services [39]. D'un autre côté le domaine des systèmes Multi-agents est un domaine relativement jeune. Il est apparu pendant les années 80 suites à l'évolution considérable des applications informatiques qui sont devenues de plus en plus distribuées et diffusées dans de multiples objets et fonctionnalités qui sont amenées à coopérer.

Un système multi-agents est un système composé d'entités informatiques, appelé des agents, qui évoluent et interagissent dans un environnement commun. La notion d'interaction entre agents est essentielle car chacun d'eux est impliqué dans une dynamique commune. Elle représente aussi ce qui permet de construire la réponse collective à partir des réponses individuelles. [1]

L'ordonnancement des tâches dans un environnement cloud computing est soumis à des contraintes particulières. La nature distribuée du cloud computing exclut toute approche centralisée. L'approche distribuée est confrontée aux limitations de la visibilité de l'état global du système dynamique. [2]

Dans ce mémoire, nous proposons un modèle qui permet l'équilibrage de charge dynamique entre les différentes machines virtuelles dans le Cloud computing en utilisant les systèmes multi agents et les algorithmes d'ordonnancements. Le modèle proposé se compose de quatre essentiels composants : L'agent de classement qui, est chargé de constituer trois classes de Machines Virtuelles (MV) ; de la plus performante à la moins performante classe des machines virtuelles, en utilisant pour cela une des techniques de classification supervisée(K-means). L'agent principal qui, en premier lieu, calcule les distances entre la tâche reçue et les centres de gravité des trois classes, puis il envoie la tâche à la classe la plus proche par rapport aux distances précédemment calculées. L'agent d'équilibrage local dont le rôle est d'équilibrer la charge entre les différentes machines virtuelles, chaque agent implémente l'un des trois programmes d'ordonnancement (SJF, FCFS,

RR), appartenant à la même classe d'un manier décentraliser, tout en prenant en considération le pire temps d'exécution (PTE) d'une tâche et sa taille. Enfin, le composant agent d'équilibrage général qui, est chargé d'équilibrer la charge entre les classes en migrant les tâches d'une classe surchargée vers une classe là moins chargée.

Notre mémoire est structuré comme suit : après l'introduction générale, le premier chapitre sera consacré aux Cloud computing, puis dans le deuxième chapitre nous présentons les systèmes multi agents ces définitions et son comportement général, le troisième est consacré pour donner un état de l'art sur l'équilibrage de charge dynamique dans le Cloud computing, puis l'architecture générale du modèle proposé, le fonctionnement et le détaille de chaque composant sera proposé et détaillé dans le quatrième chapitre, le cinquième chapitre est consacré à l'implémentation et étude de cas. Et nous terminerons ce mémoire par une conclusion générale.

# CHAPITRE 01 : CLOUD COMPUTING

## 1. Introduction :

Le Cloud computing, où informatique en nuages, a émergé comme un nouveau paradigme pour offrir des ressources informatiques à la demande et pour externaliser des infrastructures logicielles et matérielles. Dans ce chapitre nous allons présenter brièvement les concepts du Cloud computing. Nous ne nous contentons pas de définir seulement, mais de présenter aussi ses niveaux, ses modèles de déploiement, ainsi que ses caractéristiques. Ses aspects, ses améliorations et son utilité ainsi que la technique qui la constitue et les différents acteurs du secteur. [3]

## 2. Historique :

C'est la cinquième génération de l'informatique après les mainframes, les PCs, les Clients/Serveurs et le Web. L'origine du concept du Cloud provient du fait que les informaticiens symbolisent Internet sous la forme d'un nuage dans leurs schémas.


***Figure 1.1 Schéma symbolique d'internet***

Traditionnellement, l'entreprise achetait des propres serveurs, et assurait le développement et la maintenance des systèmes nécessaires à son fonctionnement. Par opposition, le Cloud computing se repose sur une architecture distante, gérée par une tierce partie. Le fournisseur assure donc la continuité du service et la maintenance. Les services de Cloud computing sont accessibles via un navigateur web. [5]

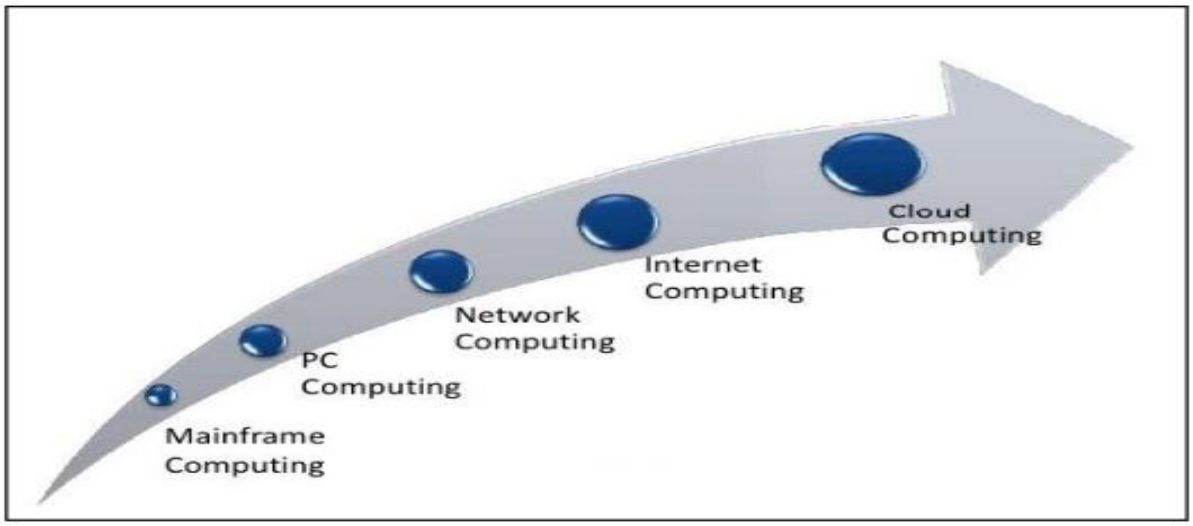


*Figure 1.2 Evolution de l'informatique*

## 3. Définition :

Le Cloud computing est un modèle pragmatisme et universel, à la demande, permettant d'établir un accès par le réseau à un réservoir partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement mobilisées et mises à disposition en minimisant les efforts de gestion et les contacts avec le fournisseur de services.


*Figure 1. 3 Cloud Computing*

# CHAPITRE 01 : CLOUD COMPUTING

"*Le Cloud Computing est un modèle qui permet d'accéder rapidement à un pool de ressources informatiques mutualisées, à la demande (serveurs, stockage, applications, bande passante, etc.), sans forte interaction avec le fournisseur de services* " [Peter.M and Timothy.G].[6]

## Principe du Cloud computing :

Le Cloud est un ensemble de matériels, de raccordements réseau et de logiciels qui fournissent des services sophistiqués que les utilisateurs peuvent exploiter à volonté depuis n'importe où dans le monde. Le Cloud computing est un basculement de tendance : au lieu d'obtenir de la puissance de calcul par acquisition de matériels et de logiciel, le consommateur se sert de puissance mise à sa disposition par un fournisseur via Internet. [7]

## 4. Caractéristique du Cloud computing [10] :

- **Elasticité :** les utilisateurs de Cloud computing à des capacités illimitées, qui peuvent être augmentées ou réduites selon l'usage.
- **Pay-ss-you-use** : les clients du Cloud vont donc recevoir des tarifications les plus précises possibles, pour les ressources qu'ils utilisent.
- **Self-service** : capacité à fournir une ressource informatique automatiquement, sans requérir d'interaction humaine côté fournisseur.
- **Mesure de la qualité de services** : évaluer et garantir un niveau de performances et de disponibilité adapté aux besoins spécifiques des clients.
- **Accès réseau universel :** l'accès très rapide des ressources à l'aide d'un réseau, par des protocoles standard en manière très élasticité.
- **Mise en commun de ressources** : Datacenter fournissant les ressources (machines, stockage, etc.) pour les différents clients en monde partagé.
- **Multi-tenancy :** la capacité de fournir un service simultanément à plusieurs clients, cela permet également d'augmenter l'utilisation des ressources déployées.
- **Disponibilité** : la haute disponibilité de la plate-forme est obtenue par redondance et par la capacité de se remettre rapidement en cas des problèmes.
- **Green IT :** ce concept qui désigne l'ensemble des nouveaux techniques à faible impact environnemental. Le Cloud computing, basé sur la virtualisation, qu'il réduise les risques des machines physiques sur l'environnement et la consommation énergétique. [8]

## 5. Acteurs :

L'écosystème du Cloud computing est composé principalement par cinq acteurs majeurs (Cloud Provider, Cloud Consumer, Cloud Carrier, Cloud Broker, Cloud Auditor) :

### 5.1. Cloud Provider :

Le fournisseur des ressources Cloud computing. Il est responsable de fournir un service Cloud computing qui satisfait les caractéristiques définies dans la précédente section, tout en respectant les Service Level Agreements (SLAs) établi avec les autres acteurs (en particulier le Cloud Consumer). Le Cloud provider a comme activité l'allocation, l'orchestration et la gestion des ressources qu'il offre tout en assurant le bon niveau de sécurité.

### 5.2. Cloud Consumer :

L'utilisateur des ressources Cloud computing. Cet utilisateur peut être un utilisateur final ou un développeur selon le type du service Cloud alloué. Cet utilisateur peut être une personne, un groupe de personnes, les petites et moyennes entreprises, les multinationales ou les gouvernements.

### 5.3. Cloud Carrier ou Network Provider :

Le fournisseur de réseau est l'intermédiaire qui assure principalement la connectivité entre les ressources Cloud computing et la liaison entre les acteurs de l'écosystème Cloud computing (en particulier entre le Cloud Provider et le Cloud Consumer). Cet utilisateur peut jouer un simple rôle d'acheminements des paquets, comme il peut jouer un rôle plus important en offrant des fonctionnalités avancées dans le réseau. Ces fonctionnalités sont basées sur des SLAs établies avec les autres acteurs de l'écosystème.

### 5.4. Cloud Broker :

Le courtier Cloud est un intermédiaire qui négocie la relation entre les Cloud Providers et les Cloud consumers. Il peut offrir de nouveaux services qui simplifient les tâches de gestion du Cloud Consumer. Ce dernier peut demander les ressources Cloud computing auprès du Cloud broker au lieu du Cloud provider directement.

### 5.5. Cloud Auditor :

L'auditeur Cloud s'occupe de la vérification et l'audition des services Cloud computing. Il évalue les services offerts par les Cloud Providers, Cloud Carriers et Cloud Brokers du point de vue

performances et sécuritaires. Le but principal est de vérifier que les fournisseurs respectent bien les SLAs qu'ils proposent. [9]

## 6. Niveaux de services Cloud computing :

Le Cloud computing peut être décomposé en trois couches :

- Applicative (SaaS, Software as a Service)
- Plate form (PaaS, Platform as a Service)
- Infrastructure (IaaS, Infrastructure as a Service)

La Figure 1.4 ci-dessous représente les différentes couches du cloud computing de la couche là moins visible pour les utilisateurs finaux à la plus visible. L'infrastructure as a Service (Iaas) est plutôt gérée par les architectes réseaux, la couche PaaS est destinée aux développeurs d'applications et finalement le logicielle comme un service (SaaS) est le « produit final » pour les utilisateurs. [Akbi.K Zehri.M,2013]

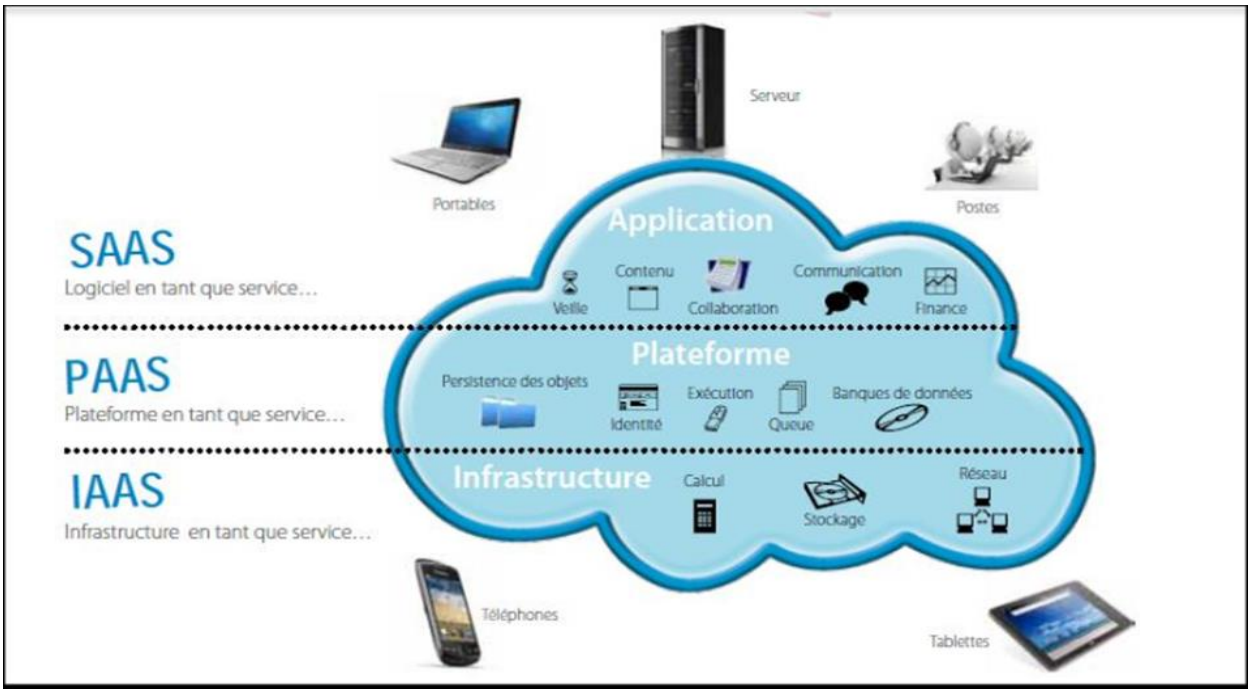


***Figure 1. 4 Les différents types de services dans le Cloud***

### 6.1. SaaS (Software as a Service) :

Ce service offre un environnement de logiciel complet à distance aux clients, par exemple, des applications de messagerie électronique, de traitement de texte, de gestion des relations clients et bien d'autres types d'applications. Examples: *Google Docs, Calendar et Gmail, Zimbra, Spotify, Salesforce.com, Microsoft Office 365 et SAP Business by Design*.

### 6.2. PaaS (Platform as a Service) :

Ce service permet aux développeurs de logiciels de créer des applications sur mesure dans les nuages, en tirant profit de la capacité des nuages à fournir automatiquement des ressources informatiques et de stockage supplémentaires lorsque cela est nécessaire. Examples: *IBM Web sphere, Force.com, spring source, Morph labs, Google App Engine, Microsoft Windows Azure ET Amazon Elastic Beanstalk.*

### 6.3. IaaS (Infrastructure as a Service) :

Ce service permet aux développeurs de logiciels de contrôler directement les ressources informatiques et de stockages fournis par un nuage. Ce service offre davantage de flexibilité au prix d'une plus grande complexité pour tirer profit de l'ensemble des services basés sur le nuage. Exemples : *Elastique Comput Cloud d'Amazon, Zimory, Elastichosts et vCloud Express de VMWare*.[12]

## 7. Les modèles de déploiement du Cloud computing :

### 7.1. Cloud public :

Les utilisateurs ont accès à des services Cloud via l'Internet public sans savoir précisément où sont hébergées leurs données ni où sont exécutés leurs traitements. Et sont fondés sur des réseaux mondiaux de centres de données (Datacenter) offrants des services basés sur un modèle de paiement à l'usage et destinés au grand public ou à un grand groupe industriel. Grâce à l'accumulation de la demande, à l'achat groupé de matériel informatique et d'énergie et à la réduction des coûts unitaires de la main-d'œuvre, les économies de coûts les plus élevées sont concentrées à ce stade. [13]

## 7.2. Cloud privé :

À partir de leur matériel informatique. Il s'agit souvent de la première étape de transition des systèmes informatiques existant vers des services d'informatique en Cloud public. Même si les Clouds privés n'offrent en général pas autant de gains de coûts que le Cloud public, ils peuvent être adaptés aux besoins de certaines organisations qui possèdent des données sensibles qu'elles ne veulent pas transférer hors de leurs systèmes.

Cette différence de coûts notable entre les différents types de Cloud est illustrée par le modèle de calcul des coûts du Cloud de Microsoft, qui révèle que le Cloud privés sont extrêmement coûteux lorsqu'ils comptent moins de 100 serveurs puisqu'ils n'offrent aucun des avantages liés aux économies d'échelle pour l'offre et la demande de Cloud public. Lorsqu'il abrite environ 1 000 serveurs, un Cloud privé offre des avantages plus importants même si l'on rapporte que ceux-ci restent toujours dix fois inférieurs par rapport aux services d'informatique en Cloud public. Les entreprises doivent également investir préalablement dans les équipements matériels et logiciels nécessaires. [13]

## 7.3. Cloud hybride :

Le Cloud hybride est une solution intermédiaire entre le Cloud public et le Cloud privé. Les ressources de calcul et de stockage sont partagées entre le Cloud public et les systèmes privés. Cela permet de traiter les données plus sensibles en interne et les autres données sur des serveurs de Cloud publics moins coûteux. [13]

## 7.4. Cloud communautaires :

Le Cloud communautaires est un autre type de Cloud créé sur mesure et exploité par un groupe d'organisations étant convenu de règles communes de sécurité et de respect de la vie privée et d'autres règles relatives à ce Cloud. La demande et les économies de coûts potentiels dépendent, entre autres facteurs, de la taille et du nombre d'organisations concernées.

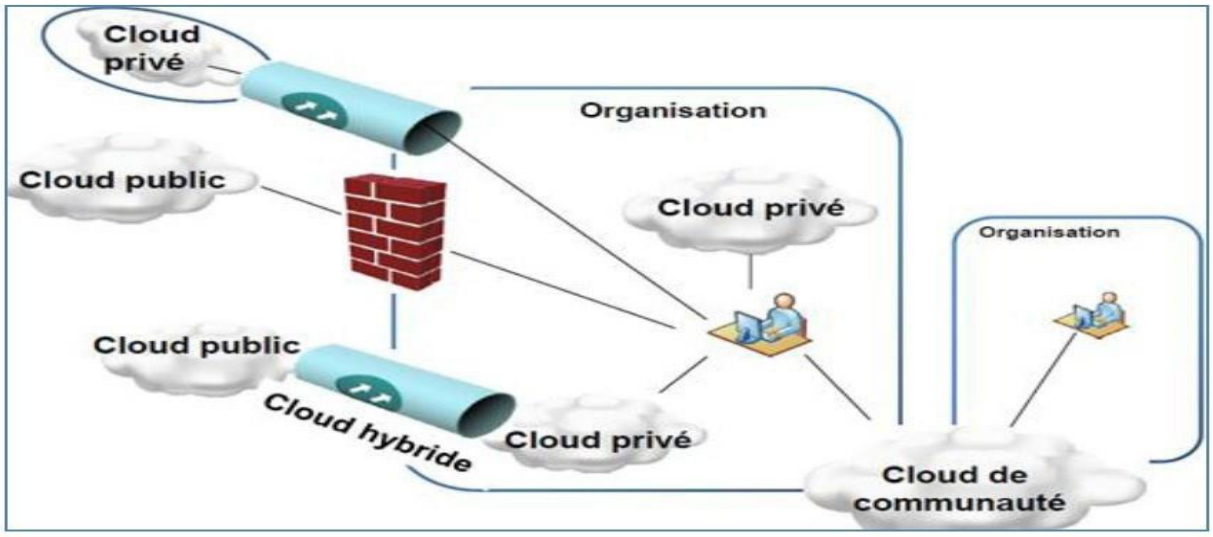


*Figure 1. 5 Modèles de déploiement du Cloud*

## 8. Les avantages et les inconvénients du Cloud Computing :

### 8.1. Les avantage :

Le modèle de Cloud computing est attractif car il libère le propriétaire de l'entreprise de la nécessité d'investir dans l'infrastructure, louant des ressources en fonction des besoins et ne payant que pour l'usage. De plus, cela permet de réduire les coûts d'exploitation, car les fournisseurs de services n'ont pas à provisionner les capacités en fonction de la charge maximale (en fait, les ressources sont libérées lorsque la demande de service est faible). Enfin, l'externalisation de l'infrastructure de service vers les Cloud déplace le risque métier vers le fournisseur d'infrastructure, généralement mieux équipé pour le gérer

En plus de ces avantages, le Cloud computing garantit un certain nombre d'avantages, notamment : l'efficacité énergétique, l'optimisation de l'utilisation des ressources matérielles et logicielles, l'élasticité, l'isolation des performances et la flexibilité. [14]

### 8.2. Les Inconvénients :

Le problème fondamental reste la sécurisation de l'accès à l'application entre le client et le serveur distant. D'autre part les entreprises perdent la maîtrise de l'implantation de leurs données ainsi que du cycle de vie des applications, et il n'y aura par ailleurs, plus la notion de confidentialité des données (financières, inventions, plans de prospection...). [14]

## 9. Conclusion :

Dans ce chapitre nous avons abordé le paradigme Cloud computing qui permet aux entreprises de disposer d'infrastructures et de logiciels directement en ligne sur Internet. On a vu les trois services du Cloud l'IaaS, PaaS et le SaaS. Ces trois services peuvent se déployer sous quatre formes de topologies différentes : le Cloud public, le Cloud privé, le Cloud hybride et enfin le Cloud communautaire. Le Cloud computing véhicule un concept de simplicité mais implique dans la réalité une vraie démarche de gestion de la complexité. Gouvernance, standardisation, processus, outils, technologies, tous les acteurs impactés sont entraînés dans l'inexorable spirale de l'évolution vers le Cloud, qu'il soit public, privé ou hybride.

Dans le chapitre suivant, nous aborderons les systèmes multi-agents et leur rôle dans le Cloud computing.

# Chapitre 02 : Systèmes Multi-Agents

## 1. Introduction :

Un système multi-agents est composé de plusieurs sous-systèmes autonomes appelés agents dont chacun a une activité et des informations propres. Le comportement général d'un SMA est lié à l'activité combinée de l'ensemble de ses agents et la réalisation d'une tâche peut alors impliquer plusieurs entités. Dans ce chapitre on va exposer les définitions et les concepts clés des systèmes multi-agents.

## 2. Agent :

### 2.1. Définitions :

Un agent est une entité (physique ou abstraite) caractérisée par le fait qu'elle est autonome dans la prise de décision, par ses connaissances sur elle-même et sur les autres, et par sa capacité d'agir.

Ce peut être un processus (en gestion des processus dans les systèmes d'exploitation), un robot (dans un environnement industriel), etc.

Pour Weiss (1999), un agent est une "entité computationnelle", comme un programme informatique ou un robot, qui peut être vue comme percevant et agissant de façon autonome sur son environnement. [11]


***Figure 2. 1 Architecture d'agent.***

## 2.2. Typologie des agents :

Il existe deux grandes familles d'agents qui sont : les agents réactifs et les agents cognitifs. Ces deux familles reposent essentiellement sur le processus décisionnel de l'agent et la représentation de l'environnement dont il dispose. Si l'agent est doté d'une représentation symbolique de l'environnement à partir duquel il est capable de formuler des raisonnements, nous disons qu'il est cognitif tandis que s'il ne dispose que d'une représentation limitée à ses perceptions, alors il est réactif.

***a-Agent réactif*** **[12]**

Un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent ne fait ni délibération ni planification, il se contente simplement d'acquérir des perceptions et de réagir à celles-ci (figure 2.2).

Ferber définit ce type d'agent comme suit : « Les agents réactifs, définis par le fait même qu'ils n'ont pas de représentation de leur environnement et des autres agents, sont incapables de prévoir ce qui va se passer et donc d'anticiper en planifiant les actions à accomplir [Ferber, 1995] ».

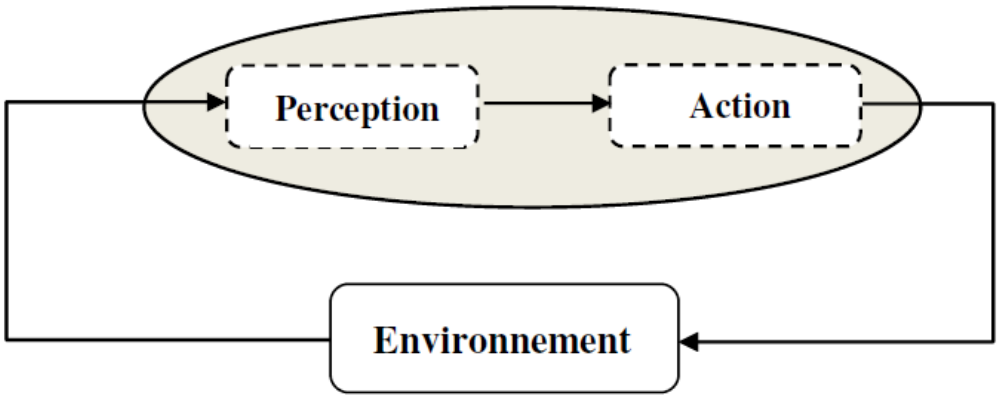


***Figure 2. 2 Agent réactif.***

Les caractéristiques de l'agent réactif sont :

- Pas de mémoire.
- Pas de représentation explicite.
- Prise de décision se basant sur le fait du Stimulus/Réponse.
- Simple à mettre en œuvre.

***b-Agent cognitif*** **[12]**

L'agent cognitif est un agent qui dispose d'une base de connaissances comprenant l'ensemble des informations et de savoir-faire nécessaires à la réalisation de sa tâche et la gestion des interactions avec les autres agents et avec son environnement [Ferber ,1995].

L'une des architectures cognitives les plus connues est sans doute l'architecture BDI : Belief (Croyance), Desire (Désir), Intention (Intention) qui comme son nom l'indique, est basée sur les notions d'attitudes mentales que sont la croyance, le désir et l'intention :

- Les croyances correspondent aux informations dont dispose l'agent sur son environnement.
- Les désirs correspondent aux états de l'environnement que l'agent souhaitera voir réalisés.
- Les intentions correspondent aux projets de l'agent pour satisfaire ses désirs.

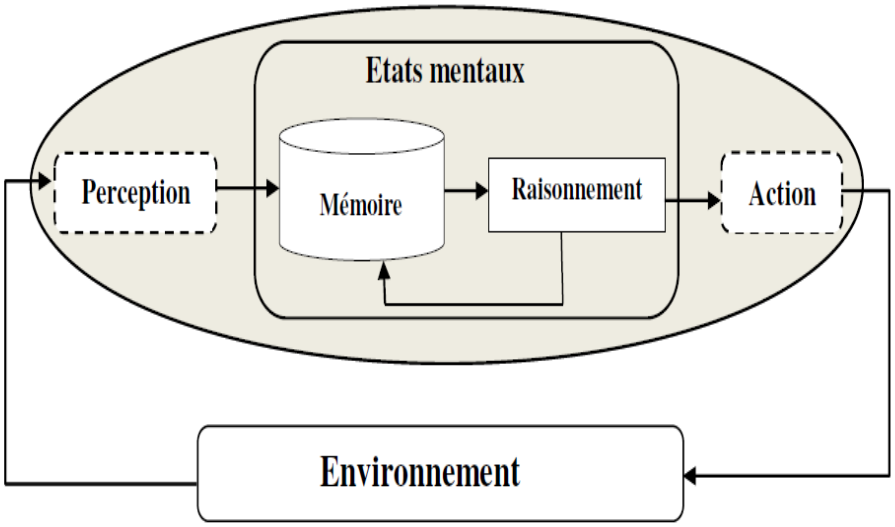


***Figure 2. 3 Agent cognitif.***

Ce type d'agent se caractérise par :

- Une représentation explicite de l'environnement et du monde auquel ils appartiennent.
- Une réaction planifiée.
- Une base de connaissances comprenant des informations et du savoir-faire.
- Une mémoire pour mémoriser les anciens états.

***c-Agent hybride***

L'architecture hybride est une structure constituée d'un ensemble d'unités organisées dans une hiérarchie, chaque unité étant soit un composant cognitif, soit un composant réactif. En conséquence, il s'agit d'une combinaison de comportement proactif de comportement ciblé à l'égard des travailleurs qui réagit aux changements de l'environnement. Dans les structures hybrides, il existe une couche interactive qui prend des décisions en fonction de données primaires. La classe moyenne extrait des données primaires et travaille au niveau de la connaissance environnementale. Enfin, la classe supérieure traite des aspects sociaux de l'environnement, c'est-à-dire de la logique qui prend en compte d'autres facteurs.

## 3. Système multi-agents :

### 3.1. Définitions :

Un système multi-agents est un système composé d'entités informatiques, appelées des agents, qui évoluent et interagissent dans un environnement commun. La notion d'interaction entre agents est essentielle car chacun d'eux est impliqué dans une dynamique commune. Elle représente aussi ce qui permet de construire la réponse collective à partir des réponses individuelles.

Un système multi-agents (ou SMA) est un système composé des éléments suivants :

*1. Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*

*2. Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*

*3. Un ensemble A d'agents, qui sont des objets particuliers (A inclut O), lesquels représentent les entités actives du système.*

*4. Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*

*5. Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.*

*6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

Les systèmes multi-agents mettent en œuvre des agents homogènes et hétérogènes ayant débuts communs ou distincts. Ils sont dynamiques.

Un système multi-agent est un système distribué composé d'un ensemble d'agents qui interagissent le plus souvent, selon des modes de coopération, de concurrence ou de coexistence. [13]

## 3.2. Typologie des systèmes multi-agents :

On distingue deux typologies des SMA selon le type des agents qui les composent, à savoir les systèmes multi-agents cognitifs et les systèmes multi-agents réactifs.

***a-SMA réactif***

Ces systèmes sont composés d'agents réactifs souvent en grand nombre. De tels systèmes se basent sur l'hypothèse qu'il est possible de produire des comportements collectifs *intelligents complexes* malgré la simplicité des comportements individuels. Dans un système multi-agent, un comportement global complexe peut apparaître comme le résultat des interactions entre des composants simples en grande quantité.

De nombreux phénomènes naturels peuvent s'expliquer de cette manière (par exemple systèmes inspirés des sociétés d'insectes comme les colonies de fourmis, les réseaux neuronaux, etc.). Les systèmes multi-agents réactifs présentent habituellement des intérêts en termes d'émergence de comportements intelligents, qui lui-même présentent des difficultés de prédiction du comportement global non représenté explicitement dans le système et de contrôle des comportements individuels par rapport à un objectif non représenté explicitement. **[14]**

***b-SMA cognitif***

Un système multi-agents cognitif est composé d'agents cognitifs. L'étude de ce système cherche à améliorer les comportements individuels des agents en s'intéressant à leur intelligence individuelle, leur modèle cognitif, et leurs communications. Ce type de système met l'accent sur l'agent et ses capacités.

Ce système est facile à concevoir pour les applications dont l'évolution est prévisible, aussi il tire profit des mécanismes de représentation complexes, et permet l'échange d'information. Cependant, il présente quelques inconvénients tels que : la difficulté de représenter les connaissances dans les problèmes complexes, la complexité de communications entre agents, la faible performance pour

des actions en temps réel, temps important de réalisation des taches et l'impossibilité d'adaptation pour un environnement dynamique. [15]

## 3.3. Communication dans les systèmes multi-agents :

**Définition**

La communication est l'échange volontaire d'informations résultant de la production et de la perception des signes d'un système commun de marques traditionnelles.
Les communications dans les SMA peuvent être classées en deux catégories distinctes, directes et indirectes.[16]

**a. Communications indirectes (ou implicites) :**

Il est basé sur l'utilisation des signaux qui prennent leurs sources en biologie et de l'éthique. En fait, des facteurs dans de tels systèmes peuvent laisser des traces de leur existence ou de leurs actions reconnues et interprétées par d'autres facteurs (par exemple, les phéromones chez les fourmis). Dans d'autres systèmes, les travailleurs environnementaux émettent des signaux ou des zones de potentiel qui guident d'autres agents dans la compétition ou la collaboration (tels que des robots pouvant signaler d'attirer ou de repousser d'autres). Agents de leur environnement).

**b. Communications directes (ou explicites) :**

Ce modèle de communication entre agents repose sur la communication par message. L'échange direct est réalisé volontairement en direction d'un individu ou d'un groupe d'individus. Dans ce type de communication les agents sont en liaison directe, les messages sont envoyés directement et explicitement de l'émetteur au destinataire et ce via un canal de communication.

## 3.4. Interactions dans les systèmes multi-agents :

**Définition**

L'avantage principal d'un système multi-agents est le concept d'interaction, qui peut être défini comme un lien dynamique entre deux opérateurs ou plus via un ensemble d'interactions.

Les interactions conduisent à une série d'actions dont les conséquences affectent le comportement futur des clients. De plus, la motivation du client dépend de l'interaction entre ses objectifs et de sa capacité à effectuer certaines tâches et ressources. L'interaction est répartie par nature et se situe généralement entre un agent informatique indépendant et son environnement. Peut être rempli, que ce soit d'autres facteurs ou des facteurs humains ou non vivants. Il est important dans la mesure où il peut combiner les efforts et s'entraider ou rivaliser et négocie [17]

## Coopération

La coopération est une caractéristique très importante des assistants des députés. La solution à ce problème est le résultat de l'interaction de coopération entre les différents facteurs. L'agent doit pouvoir mettre à jour le modèle Ocean World et intégrer les informations d'autres mandataires et districts pour aider d'autres agents et attribuer une tâche qui ne sait pas comment la résoudre à un avocat connaissant ses compétences. Ces caractéristiques sont les qualités fondamentales de l'agent de coopération. Le représentant doit coopérer dans les conditions suivantes :

- Le mode commande :

  Dans ce mode, un agent superviseur décompose un problème en sous-problèmes qu'il répartit entre les agents. Ces derniers les résolvent et envoient les résultats à l'agent superviseur.

- Le mode appel d'offre **:**

  Dans ce mode, un agent superviseur décompose un problème en sous-problèmes dont il diffuse la liste aux agents. Chaque agent envoie une offre. L'agent superviseur choisit les meilleures offres et distribue les sous-problèmes, le système fonctionne ensuite en mode commande.

- Le mode compétition :

  Dans ce mode, un agent superviseur décompose un problème en sous-problèmes, dont il diffuse la liste aux agents, Chaque agent résout un ou plusieurs sous problèmes et envoie les résultats l'agent superviseur qui à son tour fait tri.

- La coopération par partage des tâches :

  Dans ce mode, le problème est distribué entre les différents agents. Les agents travaillent indépendamment les uns des autres. Chaque agent dispose de ressources et de compétences nécessaires pour accomplir la tâche qui lui a été assignée.

- La coopération par partage des résultats :

  Dans ce modèles agents ne peuvent accomplir leurs tâches de manière indépendante, ils sont appelés à se transmettre des résultats partiels. [18]

## 3.5. Domaines d'application des SMA :

Les systèmes multi-agents ont déjà prouvé particulièrement prometteur dans de nombreux systèmes artistiques et de gestion sociale, tels que la circulation aérienne, la gestion des ressources et des applications du marché boursier, la télémédecine, les robots cognitifs, les systèmes de contrôle et de contrôle. Temps réel, etc.

La technique proxy a également trouvé sa place dans les systèmes dynamiques, où travail autonome, flexibilité et intelligence sont nécessaires lorsque la quantité de données à traiter dépasse la plage de traitement sur un seul périphérique. Ceci explique l'utilisation actuelle des systèmes multi-agents dans divers domaines, ce que nous avons brièvement mentionné [19]

- **L'Énergie** La gestion des réseaux, le support d'un centre de crise.
- **L'Industrie** L'automatisation des processus et de la production, la logistique, les robots coopératifs, la maison intelligente.
- **La Communication (y compris télécommunication)** La gestion de réseaux, le commerce électronique, la maison intelligente, les services du réseau personnel, le calcul mobile.
- **L'Information** L'assistance personnelle, la recherche d'Information, la gestion du Workflow.
- **La Santé** La supervision des malades, les systèmes de support.
- **La transportation** La logistique, le support de la mobilité, l'information du voyage.
- **Les Composants** L'automatisation de la production, les smart cartes intelligentes.

Les systèmes multi agent permet de fournie des agents intelligences, ces agents permettent d'équilibrer les taches dans les systèmes distribués (Cloud Computing) de manier décentralisée et centralisée, telle qu'un agent capable fournie plusieurs agents de même type qu'exécute multiple programmes avec multiple instructions.

## 4. Conclusion :

Les SMA proposent aujourd'hui une nouvelle technologie effective de mise en œuvre de systèmes complexes dès lors que ceux-ci requièrent distribution, ouverture, coopération et autonomie ajustable. Dans ce chapitre, nous avons quelque définition sur les agents et les systèmes multi-agents. Nous avons commencé par la définition de concept agent, puis nous avons présenté les types des agents existants qui sont classés en général en deux groupes : les agents cognitifs et les agents réactifs, à partir de deux types d'agent nous avons distingué un autre type des agents qui combine entre les deux appelés les agents hybrides. Nous avons indiqué qu'un système où évoluent plusieurs agents est appelé un système multi agents qui est présenté à la fin de ce chapitre. Dans le chapitre suivant en vas présenter les différents techniques d'équilibrage de charge.

# CHAPITRE 03: EQUILIBRAGE DE CHARGE DYNAMIQUE

## 1. Introduction :

L'équilibrage de charge est un élément important lors de la mise en place de services amenés à croître. Il faut s'assurer que la capacité à monter en charge soit la plus optimale possible afin d'éviter toute dégradation que ce soit en matière de performances ou de fiabilité lors d'affluences importantes.

Dans ce chapitre, nous examinerons les techniques les plus importantes d'équilibrage de charge.

## 2. Définition :

### 2.1. La charge :

En anglais " workload ", la charge se dénit en fonction de la nature de l'application. Elle pourrait être l'ensemble de messages servis par un dispositif du réseau, un nombre de tâches à exécuter par processeur ou encore un stock de pièces de rechange d'un système de production. Donc elle peut être de nature physique ou virtuel. [18 ]

### 2.2. Equilibrage de charge :

L'équilibrage de charge est la régulation de charge, le partage de charge ou encore l'ordonnancement distribué des charges, désigne toutes les techniques utilisées pour transférer la charge de travail des nœuds (ou sites) surchargés vers des nœuds moins chargés en respectant la capacité de chaque nœud, c'est-à-dire déterminer celui des nœuds le moins chargé et le plus apte à faire tourner l'application. Même si jusqu'à maintenant la charge d'un nœud est spécifiée surtout par l'ensemble des tâches (processus et threads) tournant sur le nœud. La vision nouvelle qui s'impose actuellement est la gestion et l'équilibrage des ressources, surtout en matière de logistique, dans une entreprise distribuée, dont les ressources sont distribuées et dont l'équilibrage constitue la condition de base pour la bonne gestion de l'ensemble. [19]

## 3. Problème de l'équilibrage de charge :

Le problème de l'équilibrage étant un problème relativement ancien, beaucoup d'approches ont été proposées pour le résoudre. Casavant et Kuhlont défini une taxonomie largement adoptée par la communauté scientifique dont les principales classes sont [20]

# CHAPITRE 03: EQUILIBRAGE DE CHARGE DYNAMIQUE

## 3.1. Approche statique :

L'équilibrage de charge statique, quant à lui est basé sur la distribution de données parmi les processeurs participants, avant l'exécution de l'algorithme.

Les stratégies statiques sont plus spécifiques que les stratégies dynamiques mais elles se révèlent utiles si suffisamment de connaissance peut être injectée dans le processus décisionnel d'ordonnancement et d'application. En d'autres termes, si les caractéristiques de la plate-forme cible (vitesses des processeurs et capacités des liens) et de l'application cible (coût des calculs et des communications associés à chaque partition de données) sont connues avec suffisamment de précision, alors un excellent niveau de performance peut être atteint par le biais de stratégies statiques. Toutefois, des schémas sophistiqués de distribution de données sont indispensables pour atteindre ce ni- veau de performance. De nombreux auteurs se sont intéressés à l'équilibrage de charge statique dans le cadre de noyaux d'algèbre linéaire sur plates-formes hétérogènes. Ces algorithmes ont, entre autres, été étudiés dans. Les principales conclusions de ces articles s'appliquent pour trois types de problèmes :

La distribution de tâches indépendantes sur un réseau unidimensionnel (linéaire) de processeurs hétérogènes est facile.

La distribution de tâches indépendantes sur une grille bidimensionnelle de processeurs est difficile. Nous devons rechercher la meilleure distribution du travail sur ladite grille pour chaque arrangement de processeurs, le nombre de tels arrangements étant exponentiel en la taille de la grille. [20]

## 3.2. Approche dynamique :

La littérature étant vaste en ce qui concerne l'équilibrage de charge dynamique, nous allons voir des approches différentes pour traiter ce problème.

Il faut, toutefois, commencer par préciser le sens de l'équilibrage de charge dynamique dans la littérature, puisque de nombreuses interprétations sont possibles. Par exemple, nous pouvons parler d'équilibrage de charge dynamique lorsque la charge varie pendant le processus d'équilibrage de charge [19]. Nous parlerons aussi d'équilibrage de charge dynamique lorsque la topologie de la plate-forme varie [25].

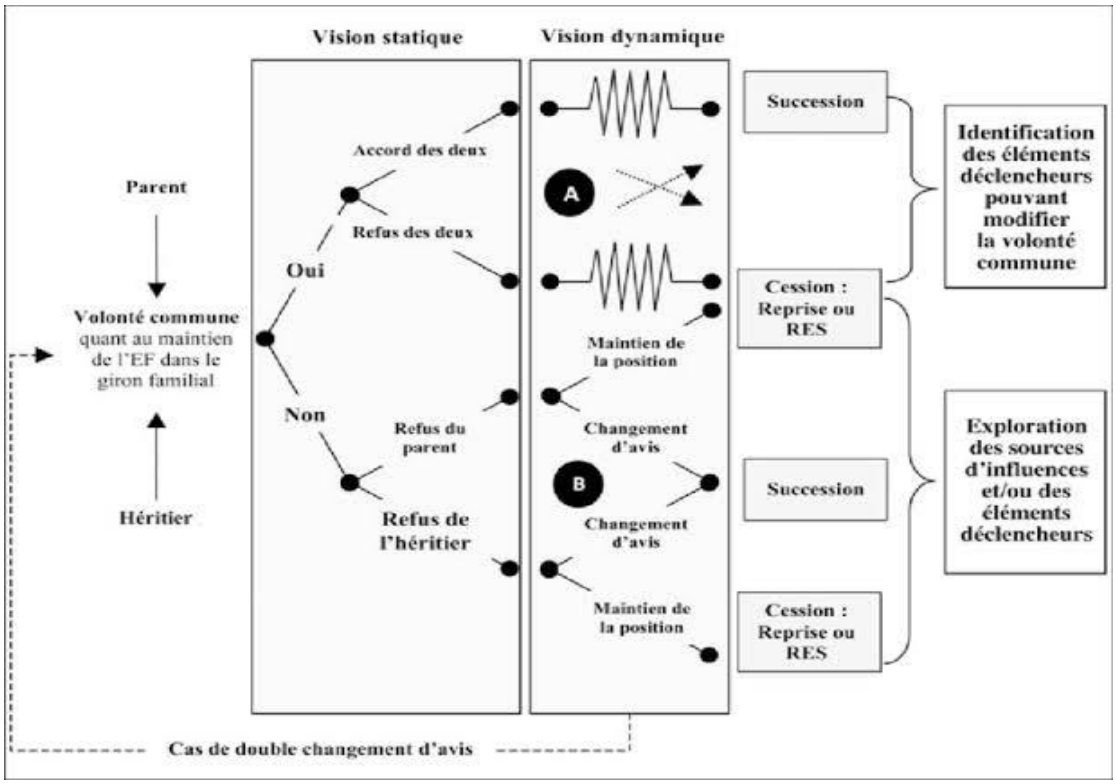


***Figure 3. 1 Approche statique VS approche dynamique [18]***

Dans ce dernier exemple, J. Bahi, R. Couturier et F. Vernier voient le réseau dynamique comme un réseau ayant des liens dynamiques. Ils supposent qu'aucun ordinateur ne peut être ajouté ou définitivement retiré de ce réseau et que chaque nœud connaît l'ensemble des arêtes viables (des arêtes peuvent être perdues lors de l'exécution de l'algorithme suite à des communications fautives ou bien encore des timeout). Avec ces considérations, les trois algorithmes qu'ils proposent sont synchrones et distribués. Leur premier algorithme est un algorithme d'équilibrage de charge par diffusion classique, nécessitant quelques adaptations du fait de la dynamique du réseau (la matrice de diffusion intègre l'information lorsqu'un lien est manquant). Les deux autres (GAE et l'algorithme de diffusion relâché accélérant la convergence de la diffusion classique) sont des adaptations de cet algorithme. Le modèle de GAE peut être vu comme un modèle de diffusion dans lequel, au temps t et pour chaque processeur, toutes les arêtes exceptée une sont inutilisables (le choix de l'unique voisin de chaque processeur s'effectue à l'aide de stratégies arbitraires, aléatoires ou plus sophistiquées). Quant au dernier algorithme, il est basé sur le schéma par diffusion, en introduisant un paramètre de relaxation dans le but d'accélérer la convergence. Les auteurs

simulent ensuite différents réseaux avec un certain pourcentage d'arêtes inutilisables (de 0% à 50%) pour illustrer le comportement de ces algorithmes et pour accentuer leur convergence vers la distribution uniforme de la charge de travail. Les algorithmes de J. Bahi, R. Couturier et F. Vernier permettent donc d'équilibrer la charge sur un réseau dynamique dans lequel les liens de communication ne sont pas fiables à 100%. [25]

## 3. Principe de l'équilibrage de charge :

Le but de toute technique d'équilibrage de charge (load-balancing) est d'optimiser l'utilisation des processeurs, chargés de prendre en charge, l'exécution des tâches dédiées à ces charges tout en spécifiant ses localités. Autrement dit, l'équilibrage de charge doit minimiser le temps d'exécution d'un ensemble donnée de tâches, ce qui revient souvent à maintenir une charge équivalente sur l'ensemble des processeurs. En général ce mécanisme est aussi utilisé pour gérer l'équité de temps d'exécution entre les tâches. [30]

### 3.1. Elaboration d'une fonction d'équilibrage de charge :

L'élaboration d'une fonction d'équilibrage de charge pour un système requiert, de manière générale, la mise en œuvre des fonctionnalités suivantes :

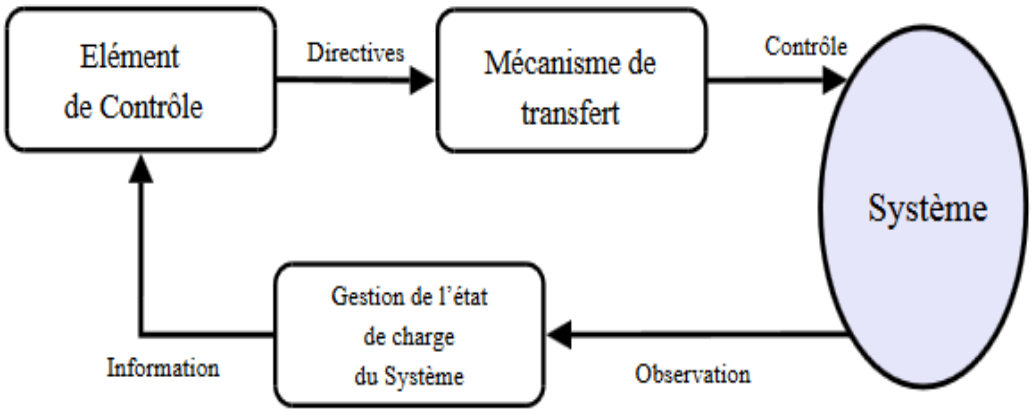


***Figure 3. 2 Eléments d'une fonction d'équilibrage [18]***

# CHAPITRE 03: EQUILIBRAGE DE CHARGE DYNAMIQUE

Un gestionnaire de l'état de charge du système (composant d'information) ; Il a pour objectif, le maintien de l'information concernant l'état de charge de chaque site pour établir la charge globale du système, à partir duquel l'élément de contrôle prendra la décision d'équilibrage de charge. Ces informations permettant de déterminer le meilleur nœud récepteur (le moins chargé) ou de déterminer le nœud émetteur (le plus chargé). Il s'appuie pour cela sur :

- L'estimation de la charge de chaque nœud.
- L'estimation de l'état global du système.

❖ Un élément de contrôle et prise de décision (composant de contrôle) ; C'est l'élément décisionnel d'une fonction d'équilibrage de charge. Il établit des directives à partir de l'état de charge des nœuds maintenu par le gestionnaire et de la stratégie de répartition qu'il met en œuvre. Le choix d'une stratégie est déterminé à partir des caractéristiques de l'architecture cible et de critères d'efficacité pour suivie par le système. Cet élément décide : L'instant où il est nécessaire d'effectuer un équilibrage de charge (politique d'Activation)

- La quantité de charge (ressources) à transférer (politique de Détermination).
- L'appariement des nœuds source et cible du transfert (politique de Sélection).

❖ Un mécanisme de transfert de la charge ; La répartition de charge peut être dynamique ou statique :

Dans une stratégie statique (placement) : la décision de placement est uniquement basée sur les informations concernant le comportement moyen du système. Dans ce cas, les tâches sont allouées à leur création sur un site et y restent jusqu'à leur terminaison.

Dans les approches dynamiques (migration) : les décisions sont prises en fonction de l'état courant du système.

- Gestionnaire de l'état de charge de système
- 2.Élément de contrôle et prise de décision
- 3. Mécanisme de transfert de la charge

Sur la base de cette répartition des rôles (Gestionnaire de l'état de charge de système, Elément de contrôle et prise de décision et Mécanisme de transfert de la charge) nous attribuerons des rôles similaires à nos agents lors de la modélisation de notre problème dans le chapitre 4, afin qu'ils puissent assurer l'équilibrage de ressources [19].

## 4. Le système d'équilibrage de charge :

Un système d'équilibrage de charge est composé de deux éléments essentiels politiques et mécanismes [20]. Les politiques considèrent l'ensemble des choix à effectuer pour distribuer une charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et fournissent les informations exigées par les politiques.

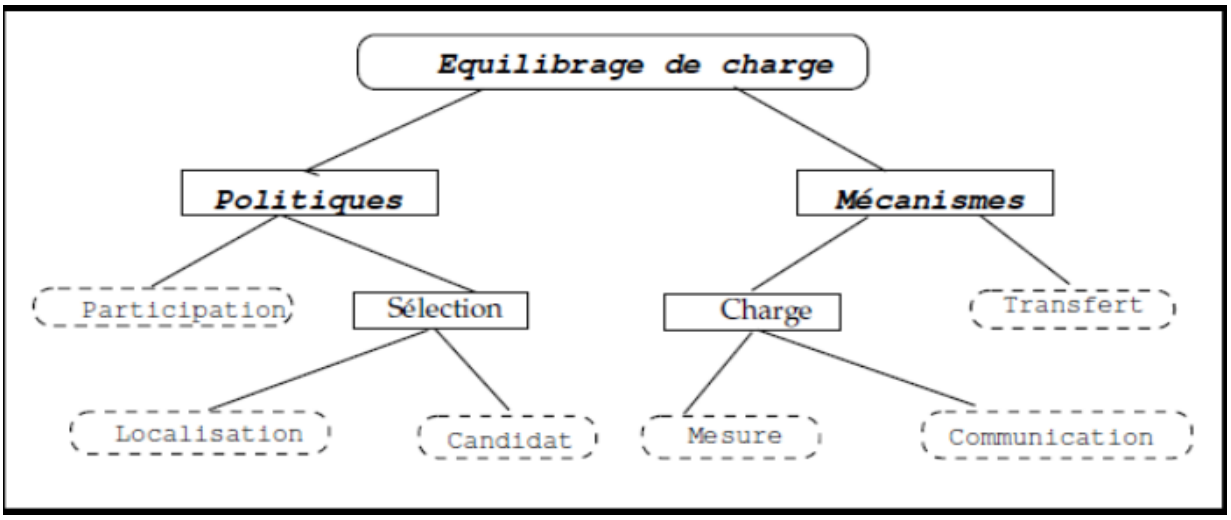


*Figure 3. 3 Composants d'un système d'équilibrage de charge [21]*

### 4.1. Les Politiques :

- **Politique de participation :** Le but de cette politique consiste à déterminer si un site est dans un état approprié pour participer à un transfert de tâches comme source (site surchargé) ou comme receveur (site sous-chargé).
- **Politique de sélection de la localisation :** Cette politique est responsable de trouver, pour un site donné, un partenaire (source ou receveur), une fois que la politique de participation a décidé que ce site était soit source, soit receveur.
- **Politique de sélection des tâches à transférer :** Une fois que les politiques de participation et de localisation ont décidé qu'un site Si est source et qu'un autre site Sj est receveur, cette politique est responsable du choix des tâches à transférer de Si vers Sj. [22]

## 4.2. Les Mécanismes :

- **Mécanisme de mesure de la charge :** Dans toute approche d'équilibrage de charge, une des difficultés majeures est celle qui consiste à évaluer la mesure de la charge d'unité. Dans la plupart des travaux existants, c'est la longueur de la file d'attente qui détermine la charge d'un site. Certains auteurs [15-16] préconisent comme indicateur de charge, une combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire. Dans le cas des grilles de calcul, il est nécessaire de tenir compte aussi de l'hétérogénéité des ressources et des réseaux de communication pour mesurer la charge d'un site.
- **Mécanisme de définition de la charge :** Ce mécanisme essaie de définir la charge globale d'un système en collectant les informations de charge sur l'ensemble ou une partie des sites du système. Il faudra alors définir les méthodes selon lesquelles l'information de charge est collectée puis diffusée aux sites. [23]

## 5. Algorithmes d'équilibrage de charge :

Les algorithmes d'équilibrage de charge sont caractérisés par leurs objectifs. Peut être utilisé dans le système pour répondre à différentes normes d'efficacité. Leurs utilisations les plus courantes consistent à optimiser l'utilisation des ressources informatiques (partage de charge) et / ou le temps de réponse des applications (équilibrage de charge). Ces cibles peuvent être complétées par des contraintes de temps plus strictes dans le contexte d'applications en temps réel ou des restrictions de tolérance aux pannes qui modifient le comportement du contrôleur de charge. [24]

Il existe plusieurs algorithmes avec caractéristique et stratégique, qui sont résumés dans le tableau suivant :

| Algo/Critères | Modèle d'information distance | Modèle d'information distance | Extensible (Scalable) | Nb de Communications locale |
|---|---|---|---|---|
| **Aléatoire** | Un seul de processus | - | Oui | 0 |
| **Paire** | Nb de processus | Un processeur au hasard | Oui indépendant | 2 messages en surcharge |
| **Vecteur** | Nb de processus | Nb de processeurs d'un sous ensemble de processeurs | Oui | 2 Messages périodiquement |
| **Drafting** | 2 seuils de processus | Ensemble des voisins directs | Oui, mais la charge se propage difficilement | Nb voisins périodiquement |
| **Enchères** | Nb de processus | Seuil de processus à l'ensemble des processus | Oui, mais c'est limité à K | 2(n−1) 2kauplus |
| **Gradient** | Nb de processus | Proximité de site sous-chargé | Oui, mais la charge se propage difficilement | Nb voisins lors de transmission de l'état |
| **Optimale** | Nb de processus | Nb de processus de tous les processeurs | Oui si le coût de Comm. Est nul | 2(n−1) |

***Table 3. 1 Comparaisons entre les différents algorithmes cités. [25]***

## 6. Equilibrage de charge dynamique dans le Cloud computing :

Avec l'apparition de l'informatique en nuage (ou Cloud Computing) plusieurs algorithmes d'ordonnancement pour équilibrer ont été adaptés pour ce type de plate-forme et étudient le problème d'ordonnancement des tâches dans les Cloud et proposent différentes techniques pratiques mises en œuvre pour résoudre ce problème.

La plupart des algorithmes d'ordonnancement dans les Cloud visent à atteindre un ou plusieurs objectifs. Certains sont liés à la rapidité, d'autres concernent la distribution équitable des ressources entre les tâches. Ect …

D'autres encore cherchent à respecter les restrictions et le placement, qui peut par exemple provenir d'affiliations ou de réductions entre les tâches. Enfin, quelques objectifs peut-être en respectant d'autres types de restrictions, En tant que contraintes frontales entre les tâches.

Il est important de noter qu'il reste beaucoup à faire pour améliorer certaines performances, surtout en matière d'équilibrage de charge et de qualité de service

Dans ce Mémoire, nous allons équilibrer les taches dans le Cloud en utilisant le système multi-agent.

## 7. Conclusion :

Dans ce chapitre, nous avons présenté les stratégies d'équilibrage de charge, nous avons d'abord définie la charge et l'équilibrage de charge, les problèmes d'équilibrage de charge (approche statique et l'approche dynamique), les principes d'équilibrage de charge, les éléments du système d'équilibrage de charge (les mécanismes et les politique). Puis nous avons abordé les défirent algorithmes d'équilibrage de charge.

Dans le quatrième chapitre, nous proposant une méthode d'équilibrage de charge dynamique basé sur des systèmes multi-agents utilisant des algorithmes d'ordonnancement dans une plate-forme (Cloud computing).

# CHAPITRE 04 : CONCEPTION DE LA METHODE PROPOSEE

## 1. Introduction :

L'équilibrage de charge est un élément important lors de la mise en place de services amenés à croître. Il faut s'assurer que la capacité à monter en charge soit la plus optimale possible afin d'éviter toute dégradation que ce soit en matière de performances ou de fiabilité lors d'affluences importantes. Le principe de base de l'équilibrage de charge consiste à effectuer une distribution des tâches à des machines de façon intelligente.

Dans ce chapitre, nous proposons un modèle qui permet de faire l'équilibrage de charge dynamique base sur les systèmes multi agent et les algorithmes d'ordonnancements entre les différentes machines virtuelles d'un Cloud. Il est structuré comme suit : après l'introduction, la section 3 sera consacrée aux travaux voisins. Dans la section 4 nous allons présenter d'abord l'architecture générale du modèle proposé, puis le fonctionnement et le détaille de chaque composant seront expliqués dans section 5. Et nous terminerons ce chapitre par une conclusion dans la section 6

## 2. Problématique :

Les systèmes parallèles et distribués apparaissent comme une solution compétitive pouvant résoudre les problèmes nécessitant de grandes capacités de calcul et la manipulation de grandes quantités de données. Plusieurs avantages caractérisent ces systèmes tels que leur grande disponibilité, leur meilleure tolérance aux pannes, leur grande flexibilité et surtout, ils permettent de partager de nombreuses ressources (processeur, mémoire, disque, etc.) à travers un réseau. L'utilisation optimale des ressources de calcul des systèmes distribués est un aspect très important qui mobilise beaucoup de chercheurs à travers le monde.

L'utilisation optimale des ressources de calcul des systèmes distribués est un aspect très important qui mobilise beaucoup de chercheurs à travers le monde. Cette optimalité nécessite une répartition de la charge de travail sur les différents nœuds de calcul. Il faut en effet éviter, dans la mesure du possible, les situations où certains nœuds sont surchargés alors que d'autres sont sous-chargés ou complètement libres., nous proposons dans ce chapitre un modèle d'équilibrage de charge dynamique basée sur les SMA et les algorithmes d'ordonnancements.

## 3. Travaux voisins :

Beaucoup de travaux ont été proposés dans le domaine de l'équilibrage de charge dynamique parmi les qu'elle on peut citer :

Les auteurs dans [29] expliquent que, les équilibreurs de charge peuvent être déployés en fonction de plusieurs architectures différentes. L'architecture d'équilibrage de charge centralisée, qui se base sur un seul équilibreur de charge central, permet de maintenir l'ensemble du système à un état équilibré, en sélectionnant la ressource de Cloud qui, devrait prendre en charge le travail reçu. Dans l'architecture d'équilibrage de charge hiérarchique, un équilibreur de charge principal (parent) reçoit toutes les demandes de travail, puis les répartit vers d'autres équilibreurs de charge connectés (enfants) où chaque équilibreur de charge de l'arborescence peut utiliser un algorithme différent.

Les auteurs dans [33] ont proposé une nouvelle approche basée sur l'idée de cloner un service Cloud sur un ou plusieurs nœuds, lorsque le nombre de requêtes utilisateur sera important à un instant donné. Ils ont proposé un algorithme qui calcul la moyenne de la charge des nœuds afin de sélectionner les nœuds les moins chargés.

Dans [28], l'algorithme d'équilibrage de charge est évalué pour les régions divisées convexes géométriques. Ces m régions fournissent m services situés à certains pm distances. L'idée est de diviser la région en plusieurs sous-régions par un facteur constant de la graisse du polygone d'origine.

Dans [38], l'algorithme d'équilibrage de charge est présenté dans l'utilisation de services redondants pour accélérer l'évaluation des demandes. Le modèle présenté s'applique à l'environnement Ethernet dans une communication asynchrone. Le service redondant est implémenté comme instanciation de la même classe, il reste sur un hôte et l'hôte détermine sa charge.

Dans [17], les auteurs proposent une nouvelle solution pour augmenter la bande passante, non seulement en tenant compte de la force du signal, mais aussi en charge du point d'accès. La solution concerne la nouvelle association de point d'accès appelée Association de points d'accès à charge équilibrée et pourrait être considérée comme une option résultat de la misassions. Les auteurs discutent l'approches, car la charge de collecte des informations pour la première option est trop élevée, impliquant la possibilité de rompre une connexion en cours sur une station à l'intérieur d'un sans fil réseau.

Dans [40], les auteurs définissent la charge comme la somme des tâches confiées à une certaine machine divisée par sa capacité. L'algorithme d'équilibrage de charge vise à minimiser la charge sur les machines du système et à offrir au planificateur des informations sous la forme d'un coût de réaffectation en fonction du travail. Un algorithme en ligne est décrit pour résoudre le problème d'attribution des tâches. L'algorithme en ligne doit être capable de satisfaire et de résoudre les demandes arrivant séquentiellement, sans connaître le final résultat.

Contrairement de ces travaux, notre travail consiste à équilibrer la charge au niveau de chaque classe de façon décentralisée, puisque les équilibreurs locaux sont indépendants entre eux, ce qui permet un gain en temps de service et de réponse d'un côté, et ils permettent de sélectionner la machine virtuelle qui correspond le mieux à la tâche reçu d'un autre côté. Quant à l'équilibreur central, il permet de mettre en contact les équilibreurs locaux des classes les surchargées et les moins-chargées afin que ces dernières collaborent entre eux pour migrer les tâches d'une classe à une autre.

## 4. Architecture du modèle proposé :

Cette section décrit l'architecture proposée (voir Figure 4.1). Cette dernière est constituée de deux types d'équilibreur de charge : les agents d'équilibrages locaux qui se trouvent au niveau de chaque classe de machines virtuelles, chaque agent implémente l'un des trois programmes d'ordonnancement (SJF, FCFS, RR), et un agent d'équilibre général qui fonctionne entre les classes de machines virtuelles.

En plus de ces deux composants, notre architecture contient le composant agent de classement, dont son rôle est de classer les machines virtuelles en utilisant la méthode K-means, et le composant agente principale qui, permet d'envoyer les tâches des clients vers les classes qui leur conviennent. Les classes d'appartenance des machines virtuelles sont : très rapide (TR), rapide(R) Moyenne (M). Notre architecture est composée, aussi, des services, des machines virtuelles et leurs files d'attente.

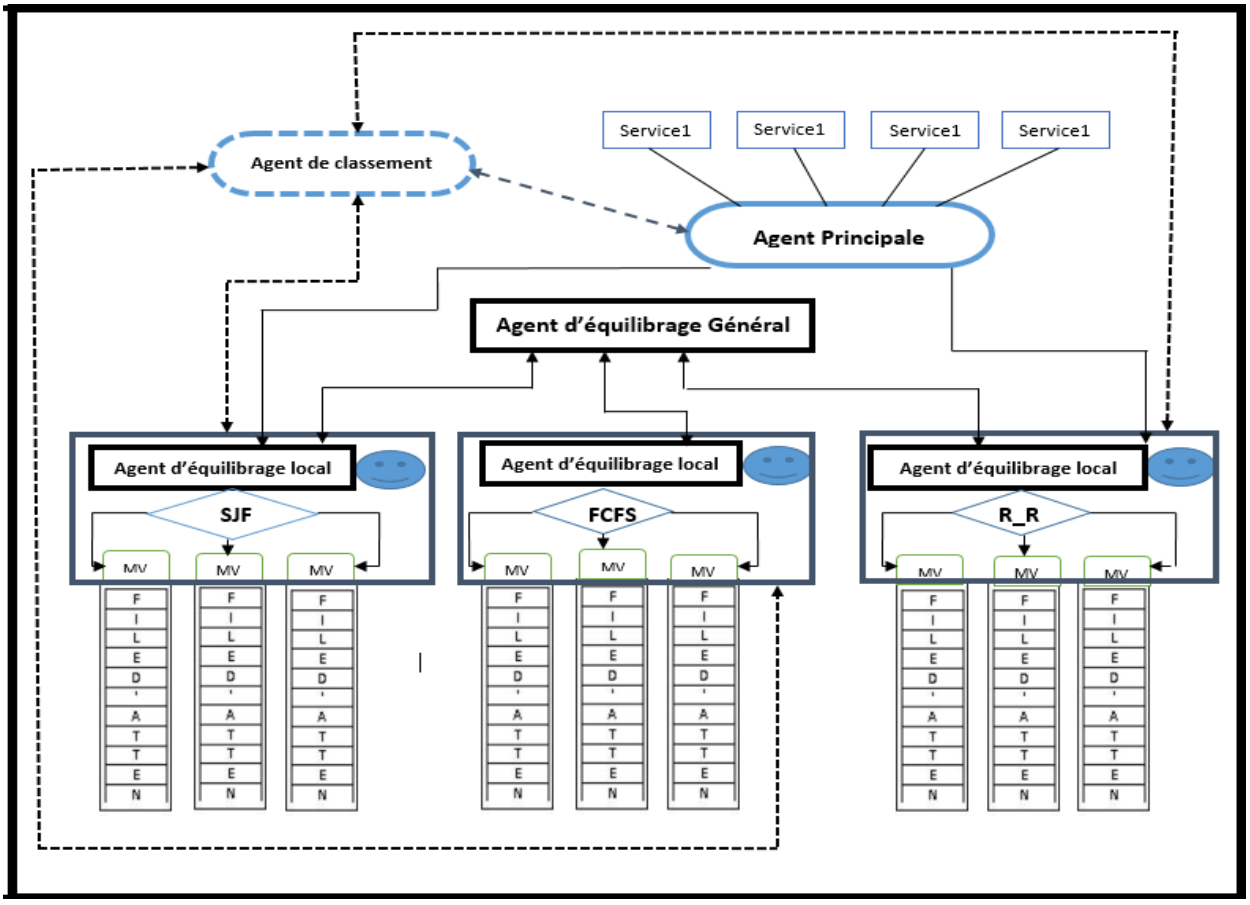


***Figure 4.1 Architecture du modèle proposé.***

## 4.1. Services :

Les services (les objets) envoient leurs tâches à l'agent principal. Les tâches se composent d'un ensemble d'instructions, et se caractérisent par leurs pires temps d'exécution qui est exprimé en Millions d'Instructions Par Seconde (MIPS) et leurs tailles.

## 4.2. Machine virtuelle :

Une machine virtuelle (MV) se compose d'un ensemble de logiciels et matériels géographiquement distants, elle permet d'exécuter les tâches émissent par les clients.

## 4.3. File d'attente :

La file d'attente est une structure qui permet d'ordonner les tâches, en attente d'exécution, selon leur ordre d'arrivée. Sa structure permet un accès direct aux tâches.

## 4.4. Agente de classement :

Cette composante permet de classer les MVs en trois classes en fonction de la vitesse du CPU et la taille de la RAM en utilisant la méthode K-means. Ce composant est déclenché à chaque fois que le nombre de machines virtuelles change.

## 4.5. Agent principal :

Cet agent est responsable d'envoyer les tâches des services vers les classes. Pour cela, en fonction des caractéristiques (pire temps d'exécution et taille) d'une tâche, et en fonction des caractéristiques des classes. L'agente principale sélectionne la classe adéquate pour l'exécution d'une tâche donnée.

## 4.6. Agent d'équilibrage local :

Chaque classe possède son propre agent d'équilibrage de charge local. Chaque agent implémente l'un des trois programmes d'ordonnancement (SJF, FCFS, RR), et il dépend des architectures centralisées SIMD (Single Program Multiple Data), où il utilise un algorithme avec des données différentes, et Chaque agent traite un algorithme dans une machine virtuelle déférente.

## 4.7. Agent d'équilibrage Général :

Ce composant assure l'équilibrage de charge entre les classes. Il est déclenché périodiquement, ou à la demande d'un agent d'équilibrage de charge local.

## 5. Fonctionnement du modèle proposé :

Les composants du modèle proposé interagissent entre eux afin de garantir une charge équitable entre les différentes machines virtuelles, ce qui permet de réduire le temps de réponse et de service.

Le fonctionnement de notre modèle est divisé en trois niveaux. Le premier niveau permet de classer les machines virtuelles en 3 classes par rapport à la vitesse des processeurs (très rapide, rapide, moyen) et la taille de la mémoire des MVs. Ce niveau est réalisé en appliquant la méthode de classement K-means, cet algorithme est exécuté la première fois, et à chaque fois que le nombre de machines virtuelles change. L'agent principal permet l'attribution d'une tâche aux MV de classe qui lui correspond.

Le deuxième niveau se situe à l'intérieur de la classe. Une fois que le pire temps d'exécution de la tâche est calculé par l'agent principal, Tous les agents d'équilibrage de charge locaux attribuent des

tâches aux machines virtuelles qui leur correspondent. Chaque agent utilise l'un des trois algorithmes d'ordonnancement. A un moment donné, il va y avoir un déséquilibre de charge à l'intérieur de la classe car le choix de MV est basé sur le pire temps d'exécution d'une tâche par rapport à la vitesse du processeur de la MV, c'est-à-dire que probablement certaines tâches vont terminer avant leur temps estimé, ce qui provoque un déséquilibre entre les MVs de la même classe.

Le troisième niveau est l'équilibrage de charge entre les classes. Ainsi, l'agent d'équilibrage général est responsable d'équilibrer de charge entre les classes. Pour cela, il sélectionne les classes surchargées, par calculant la charge pour chaque catégorie afin de migrer les tâches de ces derniers vers les classes les moins chargées.

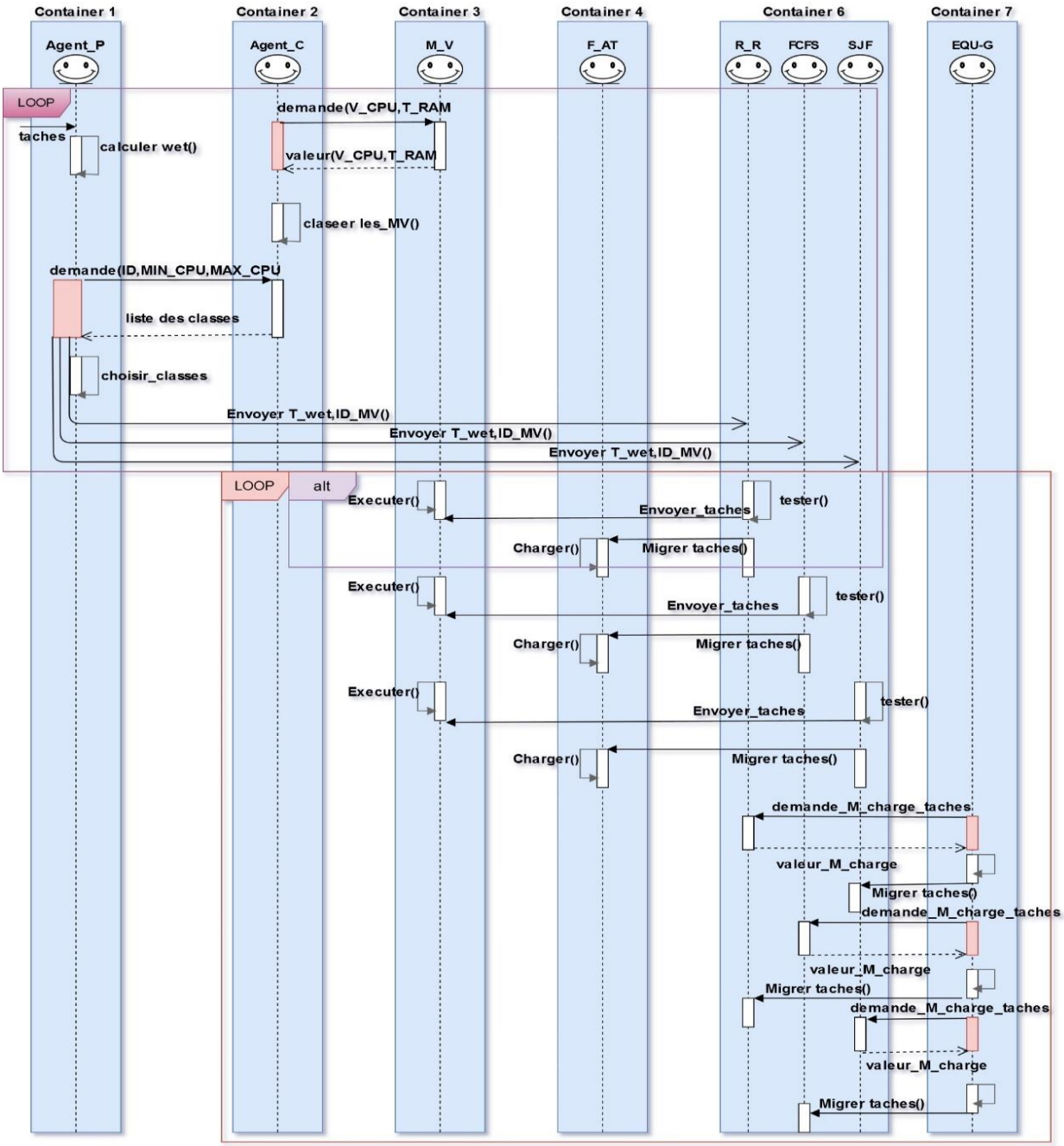


*Figure 4.2 Diagramme de Séquence du Modèle*

## 5.1. L'agent principal :

Le Routeur commence par calculer le pire temps d'exécution d'une tâche en se basant sur les travaux de [22] (voir l'algorithme ci-dessous), puis en fonction de ce temps calculé, de la taille de la tâche et en fonction du centre de gravité Ci de chaque classe, il sélectionne la classe la plus adéquate à la tâche.

Voici L'algorithme WCET pour calcule le pire temps d'exécutions des taches. [32]

**Algorithme WCET pour calculer le pire temps d'exécution**

**//La Tache est un ensemble d'action (boucle, affectation, test…)**

**//TTime : tableau dynamique**

**//On prend le simple algorithme du WCET**

1. Parcourir l'arbre des actions en boucle multicritère, si la fin de l'arbre aller vers 5
2. A chaque fin de feuille on calcule le nombre d'action
3. Si la fin de tache mettre le résultat dans le tableau TTime et Aller vers 1
4. Sinon aller vers 2
5. Parcourir le tableau TTime en cherchant le max
6. WCET : =max(w)

## 5.2. L'agente de classement :

L'agente de classement applique la méthode de K-means. Ainsi, il permet de classer toutes les MVs en trois classes (TR, R, M). Les MVs sont caractérisées par deux critères (Vitesse CPU et taille RAM). La méthode commence par affecter les trois premières MVs aux trois classes respectivement puis, à chaque itération elle calcule la distance entre la MV et le centre de gravité de la classe. [35]

$$\boldsymbol{Dxy}=\sqrt{(xcpu-ycpu)^2+(xRAM-yRAM)^2} \quad \textbf{(1)}$$

Où Dxy est la distance entre deux nœuds.

$$\boldsymbol{Ci}: cpui=\frac{\sum_{j=1}^{n} mvi(cpu)}{n} \quad \textbf{(2)}$$

$$\boldsymbol{Ci}: \ RAMi=\frac{\sum_{j=1}^{n} mvi(RAM)}{n}$$

Où Ci (CPU, RAM) est le centre.

L'agent de classement affecte la MV à la classe où la distance est la minimale, jusqu'à ce que la structure des classes soit la même. Les classes sont nommées R, M, L, et chacune d'elles est caractérisée par son centre de gravité Ci.

Voici L'algorithme K-means pour classifier les MVs en trois classes :

1. Créer la liste des classes (TR, R, M)
2. Récupérer les machines virtuelles (CPU, RAM, Etat de CPU)
3. Affecter les trois première MV aux trois classes respectivement
4. Calculer le centre de chaque classe
5. Calculer la distance entre chaque MV et les trois centres
6. Affecter chaque MV à la classe la plus proche.
7. Tant que l'état des trois classes est diffèrent allers vers 8sinon allé vers 12
8. Calculer le centre de chaque classe
9. Calculer la distance entre chaque MV et les trois centres (
10. Affecter la MV à la classe où la distance est la plus petite.
11. Aller vers 7
12. Fin

## 5.3. Les Agents d'équilibrage local :

Nous avons trois agents dans une architecture centralisée (SIMD), chaque agent implémentant l'un des trois algorithmes d'ordonnancement (SJF, FCFS, RR).

## 5.3.1. Agent d'équilibrage SJF (Préemptif) :

Voici L'algorithme de SJF :

1. Traversez jusqu'à ce que tout le processus soit complètement exécuté.
2. Trouvez le processus avec le temps restant minimum à chaque tour de temps.
3. Réduisez son temps de **1**.
4. Vérifiez si son temps restant devient **0**
5. Incrémentez le compteur de fin de processus.
6. Heure de fin de**processus actuel = ProssAt_time** +1 ;
7. Calculer le temps d'attente pour chaque processus terminé.
8. **wt[i]= Completion time - arrival_time-burst_time**
9. Incrémenter le tour de temps de un.
10. Trouver le temps de traitement

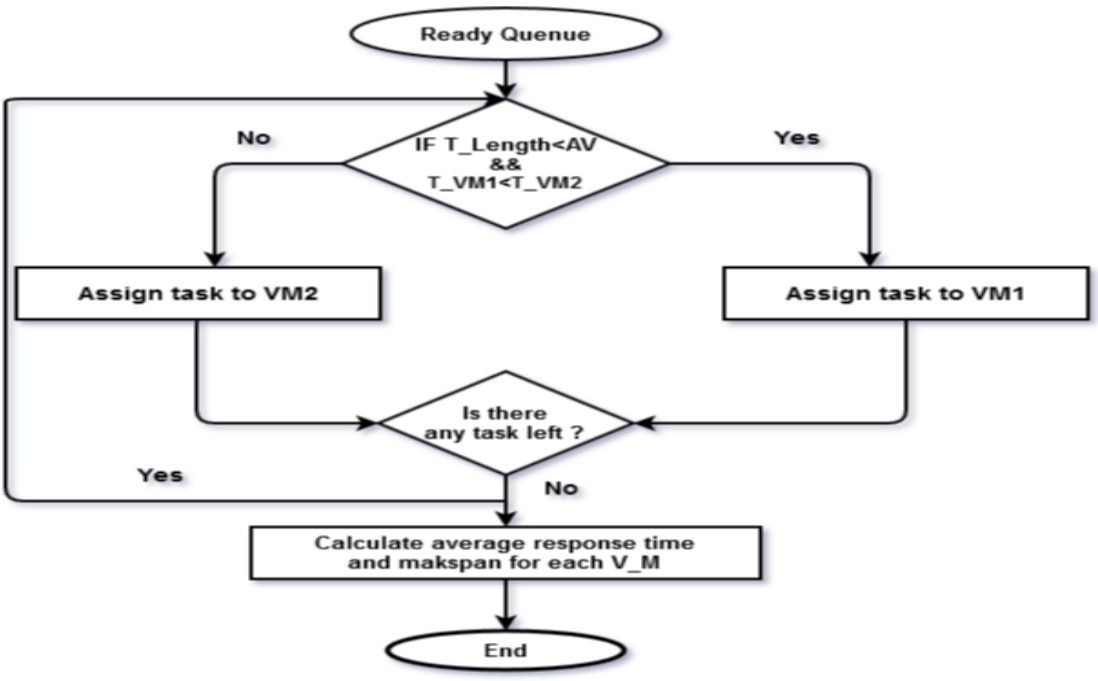


***Figure 4.3 Diagramme d'activité d'algorithme SjF.***

## 5.3.2. Agent d'équilibrage R_R :

Voici L'algorithme de R_R :

1. Créez un tableau **rem_bt []** pour garder une trace du temps de rafale restant des processus.Ce tableau est initialement une copie de **bt []** (burst times array)
2. Créez un autre tableau **wt []** pour stocker les temps d'attente des processus. Initialisez ce tableau à **0**.
3. Temps d'initialisation **: t = 0.**
4. Continuez à parcourir tous les processus pendant que tous les processus ne sont pas terminés. Suivez pour **i**the processus si ce n'est pas encore fait.
5. **Si rem_bt [i]> quantum**

   **(i) t = t + quantique**

   **(ii) bt_rem [i] - = quantum;**

6.**Sinon //** Dernier cycle de ce processus

**(i) t = t + bt_rem [i];**

**(ii) wt [i] = t - bt [i]**

**(ii) bt_rem [i] = 0 ; //** Ce processus est terminé

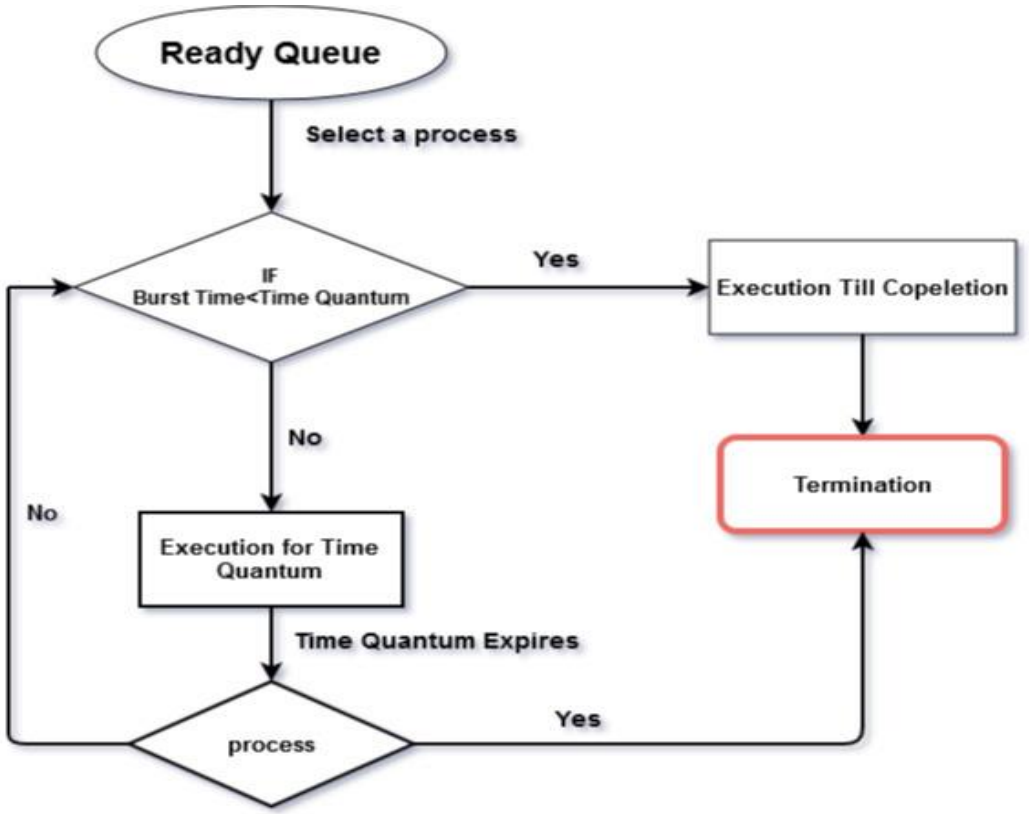


*Figure 4.4 Diagramme d'activité d'algorithme R_R.*

## 5.3.3. Agent d'équilibrage FCFS :

Voici L'algorithme de SJF :

1. Entrez les processus avec leur temps de rafale **(bt)**.
2. Trouvez le temps d'attente **(wt)** pour tous les processus.
3. Comme le premier processus qui vient ne doit pas attendre,
   Le temps d'attente pour le processus **1** sera de **0**, c'est-à-dire **wt [0] = 0**.
4. Trouvez le temps d'attente pour tous les autres processus,
   C'est-à-dire pourprocessus **i -> wt [i] = bt [i-1] + wt [i-1]**.
5. Trouver **le temps de traitement = temps_attente + temps_saute**
   Pour tous les processus.
6. Trouvez **le temps d'attente moyen = total_waiting_time / no_of_processes**.
7. De même, recherchez **le temps de traitement moyen= total_turn_around_time / no_of_processes.**

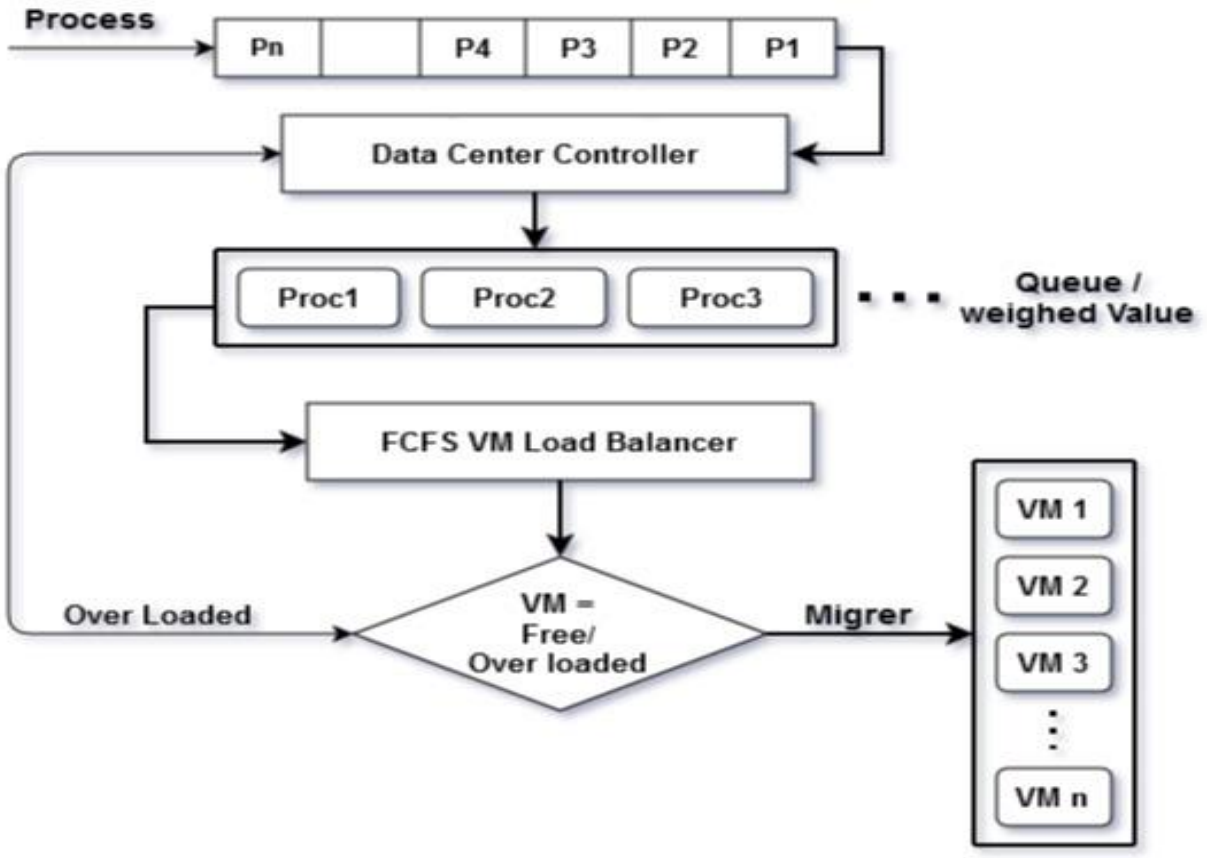


*Figure 4.5 Diagramme d'activité de l'algorithme FCFS. [31]*

## 5.4. Agent d'équilibrage général :

L'équilibreur général calcul la moyenne de charge des classes afin de migrer les tâches des classes les plus chargées vers les classes les moins chargées. Dans ce contexte, nous pouvons distinguer deux cas. Le premier cas est lorsque la classe réceptrice est inférieure à la classe émettrice, dans ce cas la tâche candidatée doit être choisie parmi celles ayant le pire temps d'exécution le plus petit. Dans le cas contraire, il choisira la tâche dont le pire temps d'exécution est le plus élevé.

Voici L'algorithme d'aide à la décision TOPSIS :

# CHAPITRE 04 : CONCEPTION DE LA METHODE PROPOSEE

1. Lecture des critères
2. Lecture des poids
3. For i : =1 to n do
4. For j : =1 to n do
5. Construire la matrice d'entrée // **ligne = Taches, colonne =temps et taille**
6. Normaliser la matrice en utilisant la distance euclidienne
7. Calcul du la meilleur et la pire solution (A+ et A-)
8. Calcul de la distance de chaque alternative de la pire et la meilleure solution (E+ et E-)
9. E+ vecteur exprimant la distance de chaque alternative de A+
10. E- vecteur exprimant la distance de chaque alternative d'A-
11. Calcul de la proximité de chaque alternative
12. Le coefficient de proximité de chaque alternative : S : =E-/ (E- +E+) ;
13. Afficher (Max(S[i])) ; // **la tâche à choisir**

Il calcul la charge de chaque MV (MV c) selon la formule (A).

$$\boldsymbol{MVc = Vp * \sum_{j=1}^{m} Tj} \qquad \textbf{(A)}$$

Où **Vp** est la vitesse du processeur, **Tj** est le pire temps d'exécution de la tâche et **m** est le nombre de tâches en attente d'exécution.

Puis il calcul la moyenne de charge de la classe selon la formule (B), où n est le nombre de **MVs**

$$\boldsymbol{Moy\ C = \sum MVc / n} \textbf{(B)}$$

Après il sélectionne les MVs surchargées, ç-à-d ceux dont leur charge est supérieure à la moyenne, afin de migrer certaines tâches vers les MVs les moins chargées. Le choix de la tâche à migrer se fait en se basant sur deux critères essentiels qui sont : la taille de la tâche et son pire temps d'exécution. Afin de minimiser le cout de migration, l'équilibreur local doit choisir la tâche dont la taille est le plus petit possible mais le pire temps d'exécution est le plus élevé.

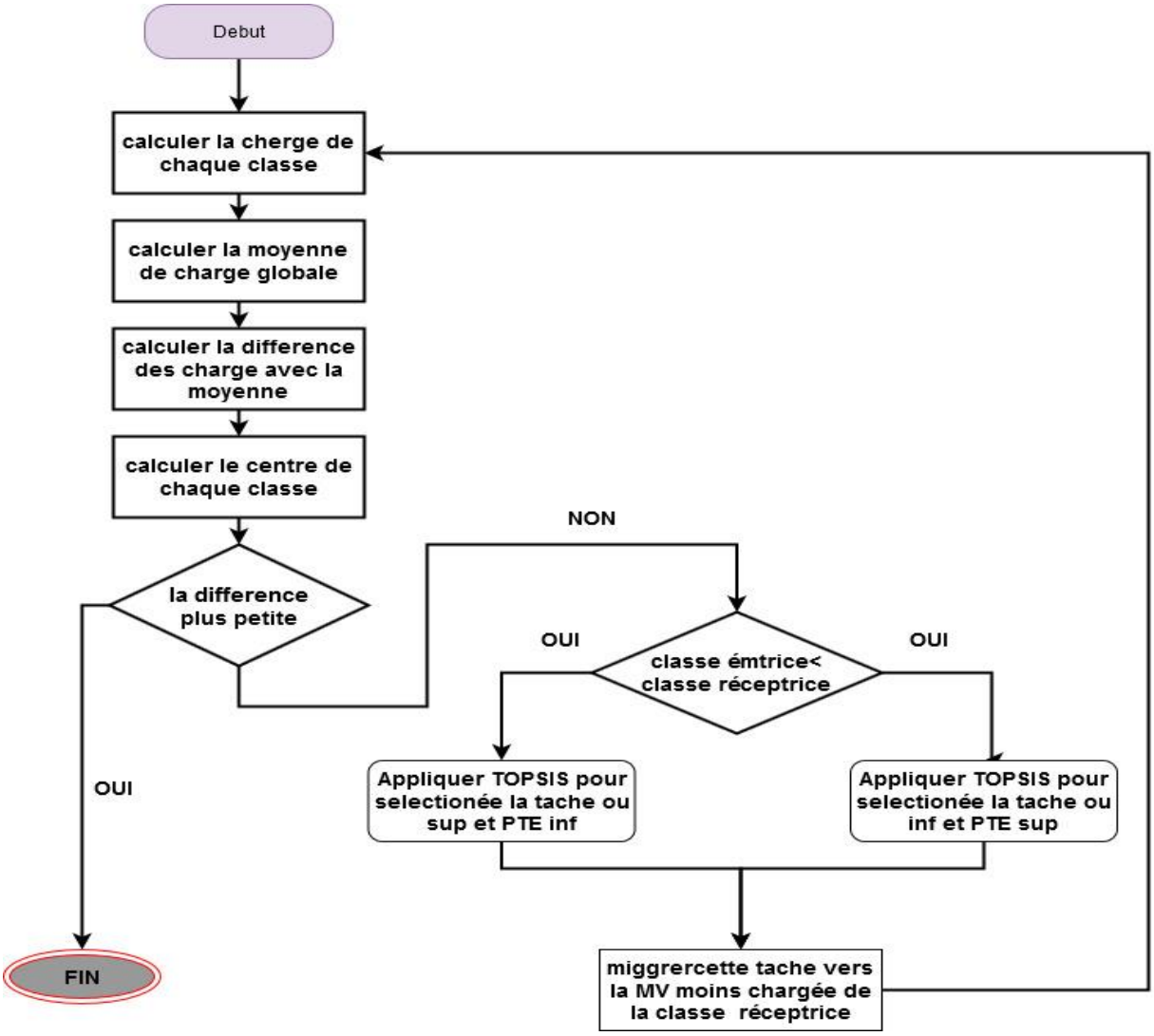


*Figure 4.6 Diagramme d'activité d'application TOPSIS. [37]*

## 6. Conclusion :

Dans ce chapitre, nous avons abordé le problème de déséquilibrage de charge entre les MVs dans le Cloud computing, nous avons proposé un modèle qui permet de faire l'équilibrage de charge dynamique base sur les systèmes multi agent entre les différentes machines virtuelles d'un Cloud. Le fonctionnement de notre modèle pour équilibrer la charge se déroule en trois phases principales : la première phase consiste à classifier les MVs en trois classes, la deuxième phase est l'affectation des tâches aux classes et ceci en fonction des caractéristiques des tâches (PTE et Taille) et le centre de gravité des classes, quant à la troisième phase, elle permet d'équilibrer la charge intra-classe et interclasse.

Dans le chapitre suivant, nous envisageons de valider notre modèle en effectuant des simulations dans l'environnement Cloud Sim, et Platform JADE afin de montrer la pertinence de notre modèle d'un côté, le temps nécessaire pour affecter les tâches aux classes, et bien d'autres paramètres qui nous permettrons de ressortir les points positifs et négatifs de notre proposition.

.

# CHAPITRE 05 : IMPLEMENTAION

## 1. Introduction :

Dans le quatrième chapitre, nous avons proposé une architecture de notre modèle basée sur le système multi agent et la classification des MVs, et afin de clarifier les différentes idées et concepts inclus dans l'architecture proposée, nous allons dérouler les principaux aspects de notre architecture sur un exemple concret afin de montrer la faisabilité et la mise en évidence de nos idées.

Dans ce chapitre, nous présenterons l'environnement de travail, c'est-à-dire les différents langages de programmation ainsi que l'environnement matériel et logiciel dont nous aurons besoin pour vérifier notre architecture par des simulations. Nous utilisant la plate-forme jade pour les system multi agent, CloudSim, et langage de programmation java.

## 2. Définition de la simulation informatique :

La simulation informatique est une série de calculs effectués sur un ordinateur et reproduisant un phénomène physique. Elle aboutit à la description du résultat de ce phénomène, comme s'il s'était réellement déroulé. Cette représentation peut être une série de données, une image ou même un film vidéo.

Un simulateur peut réagir à des modifications de paramètres et modifier ses résultats en conséquence. Un simulateur de vol, par exemple, modifie la trajectoire calculée de l'avion en fonction des commandes transmises par l'utilisateur.

## 3. Langage et environnement de développement :

### 3.1. Langage de programmation java :

Java est un langage de programmation informatique orientée objet et un environnement d'exécution informatique portable 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 Sun World.

Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java à la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs

systèmes d'exploitation tels que unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications... C'est la plate-forme qui garantit la portabilité des applications développées en Java.

Java permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur. Côté client, les applets sont à l'origine de la notoriété du langage. C'est surtout côté serveur que Java s'est imposé dans le milieu de l'entreprise grâce aux servlets, le pendant serveur des applets, et plus récemment les JSP (Java Server Pages) qui peuvent se substituer à PHP, ASP et ASP.NET.

## 3.2. JADE - Java Agent Développent Framework :

JADE est une plate-forme multi-agents créée par le laboratoire TILAB. Entièrement implémentés en JAVA et permettent le développement et l'exécution de systèmes multi-agents conformes aux normes de la FIPA.

Un agent selon JADE est conforme au standard FIPA, possède un cycle de vie, possède un ou plusieurs comportements (*Behaviours*), communique avec des messages du type agent communication language (*ACL*) et rend des services. Un agent est identifié de manière globale par un nom unique (l'*Agent Identifier* ou *AID*).

JADE contient ainsi :

### 3.2.1. Les agents de base de JADE : [38]

**a. Remote Management Agent « RMA » :**

Le RMA permet de contrôler le cycle de vie de la plat-forme et tous les agents la composant. Avec son interface, il est possible de créer, supprimer et de migrer des agents. L'agent RMA permet également la création et la suppression des conteneurs et la fermeture de la plateforme. Il peut être lancé à partir de la ligne de commande **Java jade. Boot –gui.**

**b. Agent Management System « AMS » :**

Le AMS est un agent chargé de la supervision et le contrôle de l'accès et l'usage de la plateforme. Chaque agent doit s'inscrire auprès de l'AMS pour obtenir un AID valide.

**c. Directory Facilitor « DF »** :

Enregistre les descriptions et les services des agents et maintiennent des pages jaunes de services. Grâce à ce service, un agent peut connaître les agents capables de lui fournir les services qu'il requiert pour atteindre son but. L'interface du DF peut être lancée à partir du menu du RMA.

**d. Dummy Agent « DA »** :

L'outil *DummyAgent* permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

**e. Sniffer Agent « SA »** :

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent Sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du RMA ou de la ligne de commande suivante Java **jade. Boot sniffer : jade. Tools. Sniffer. Sniffer.**

**f. Introspector Agent** :

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et de la file de ses messages envoyés et reçus.

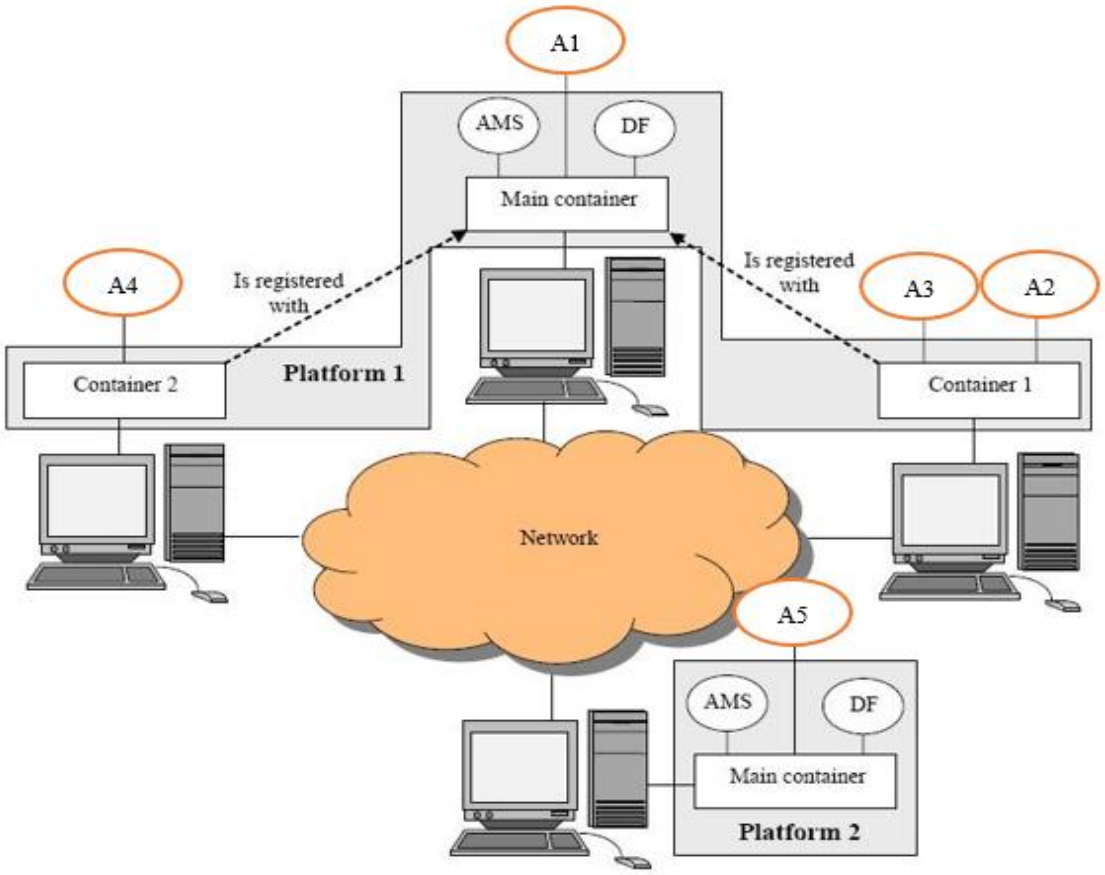


***Figure 5.1 La Plat-form JADE***

## 3.3. CloudSim :

Ce Framework modélise et simule l'environnement du Cloud computing et ses services, a été réalisé en Java. Afin que nous puissions simuler cet environnement, il faut connaitre l'architecture de ce Framework.

Cloud Sim est un simulateur qui permet la modélisation, la simulation et l'expérimentation de nouvelles Cloud Computing infrastructures et de services d'application. Dans le cas du Cloud Computing, les outils de Simulations comme CloudSim donnent d'importants avantages aux clients et fournisseurs. Pour les clients, il leurs permet de tester leurs services dans un environnement contrôlable avec exemple du coût et de vérifier la performance avant de les publier sur les Cloud. Pour les fournisseurs, il leurs permet de vérifier les types de location en fonction de divers prix et la charge. Sans ces outils, à la fois les clients et les fournisseurs doivent s'appuyer

sur des évaluations imprécises, ou sur des approches essai-erreur, ces approches peuvent conduire à l'inefficacité de la performance des services. En outre, CloudSim aide les chercheurs et développeurs à tester la performance d'un service d'application développée.

## 3.4. L’architecture du CloudSim :

La structure logicielle de CloudSim et ses composants est représentée par une architecture en couche comme le montre la Figure 5.2. Le niveau le plus bas est le moteur de simulation des événements discrets SimJava, qui implémente les fonctionnalités de base requises pour les cadres de simulation du niveau supérieur, telles que les files d'attente, le traitement des événements, création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation. CloudSim supporte la modélisation et la simulation de l'environnement de Datacenter basé sur le Cloud, tel que les interfaces de gestion dédiées aux VMs, la mémoire, le stockage et la bande passante. La couche CloudSim gère l'instanciation et l'exécution des entités de base (VM, hôtes, Datacenters, applications) au cours de la période de simulation. Dans la couche la plus haute de la pile de simulation, on trouve le code de l'utilisateur qui expose la configuration des fonctionnalités liées aux hôtes (ex : nombre de machines, leurs spécifications), les politiques d'ordonnancement de Broker, applications (ex : nombre de tâches et leurs besoins), VM, nombre d'utilisateurs.

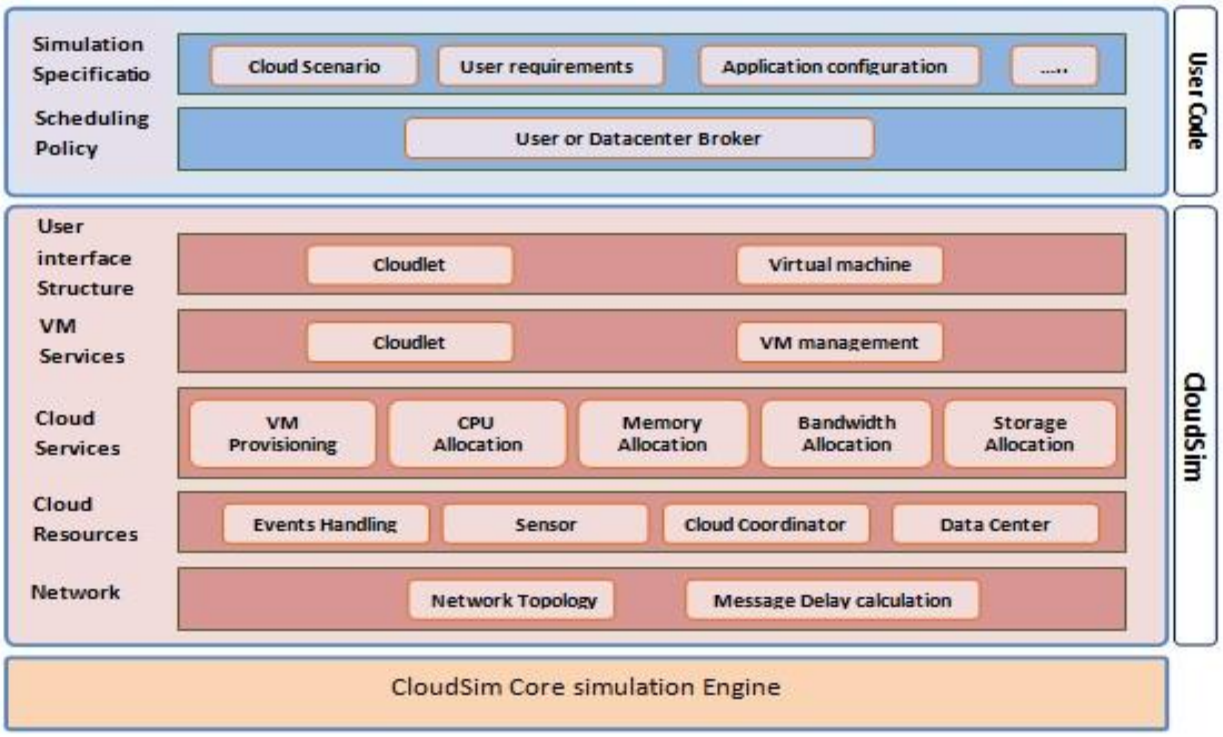


***Figure 5.2 Les couches de l'architecture du CloudSim***

## 3.5. Cloudlet :

Cette classe modélise les services d'application de base du Cloud (tels que la livraison, la gestion des réseaux sociaux, et le déroulement des opérations métier de l'entreprise) qui sont généralement déployer dans des Datacenter. CloudSim orchestre la complexité de la demande en fonction de ses exigences de calcul. Il représente la complexité d'une application en termes de ces besoins informatiques (taille en nombre d'instruction, temps de réponse.). Cette classe peut aussi être propagé pour soutenir la modélisation d'autres mesures de performance et de composition pour des applications telles que les transactions dans les applications orientées base de données.

## 3.6. Virtual Machine :

Cette classe modéliser une instance de virtuelle machine (VM), qui est gérée et hébergée pendant son cycle de vie par le composant Cloud host, un host peut simultanément instancier de multiples VMs et assigner des cœurs de processeur (espace partagé, temps partagé).

## 3.7. DataCenter :

Cette classe modélise l'infrastructure du noyau du service (matériel, logiciel) offert par des fournisseurs de ressources dans un environnement de Cloud Computing. Il encapsule un ensemble de machine de calcul qui peut être homogènes ou hétérogènes en ce qui concerne leurs configurations de Datacenter instancie un composant d'approvisionnement de ressource qui implémente un ensemble de politiques d'allocation de bande passante, de mémoire et des dispositifs de stockage. L'HOST représente un serveur informatique physique dans un Cloud, il exécute des actions liées à la gestion des machines virtuelles et a une politique définie pour l'approvisionnement en mémoire et bande passante, ainsi que d'une politique de répartition des PE (élément de traitement) sur des machines virtuelles. Un hôte est associé à un Datacenter, et peut héberger des machines virtuelles.

## 3.8. Datacenter Broker :

Cette classe modéliser 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 il déploie les tâches de service à travers les Clouds.

Le Broker agissant au nom des utilisateurs, identifie les prestataires de service appropriés du Cloud par le service d'information du Cloud CIS (Cloud Information Services) en négociant avec eux pour une allocation des ressources qui répondent aux besoins de QoS des utilisateurs**.**

## 4. Description de l'application Cloud Sim :

La version de Cloud Sim 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 simulateur, tel que la modification des paramètres de simulation, l'affichage graphique des outputs (résultat de simulation). Ainsi, comme première étape, nous avons développé une interface graphique qui facilite l'accès et la manipulation du simulateur, puis dans la deuxième étape, nous avons implémenté notre approche par l'enrichissement des classes du simulateur Cloud Sim, et enfin, nous lançons la simulation afin d'obtenir des résultats et de valider notre approche.

## 5. Description de l'implémentation :

Notre application, appelé « SMA_Balancer », est composée d'une interface principale, après le démarrage de main container, qui contient quatre boutons dont le rôle de chacun d'eux est, respectivement, la création des machines virtuelles, la création des taches (cloudlet), la classification qui permet de classer les machines virtuelles. Et Equilibrage de Charge qui permet de détailler les étapes de la simulation : en commençant par le routage, l'équilibrage local, et l'équilibrage General. Comme la montre les Figures Ci-dessous

Tout d'abord, l'interface JADE apparaît

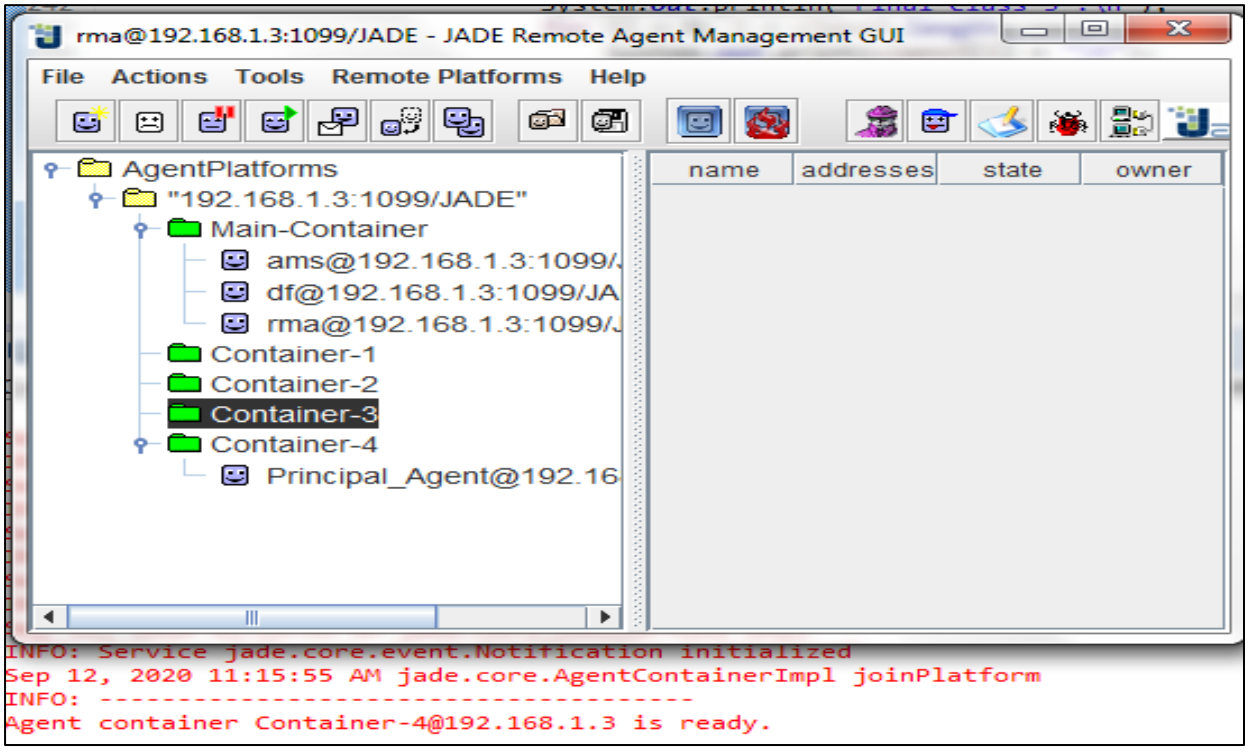


***Figure 5.3 L'interface du JADE***

Au début en lance la main container, L'interface principale apparaît, contient les quatre buttons.


***Figure 5.4 L'interface du SMA_Balancer.***

Pour créer les machines virtuelles et les tâches, il suffit de cliquer sur les deux boutons respectivement (Creation_MV, Creation_CloudLet) comme le montre la figure 5.5.


***Figure 5.5 Création des machines virtuelles et les tâches.***

Pour classer les machines virtuelles, nous avons implémenté le bouton Classification.

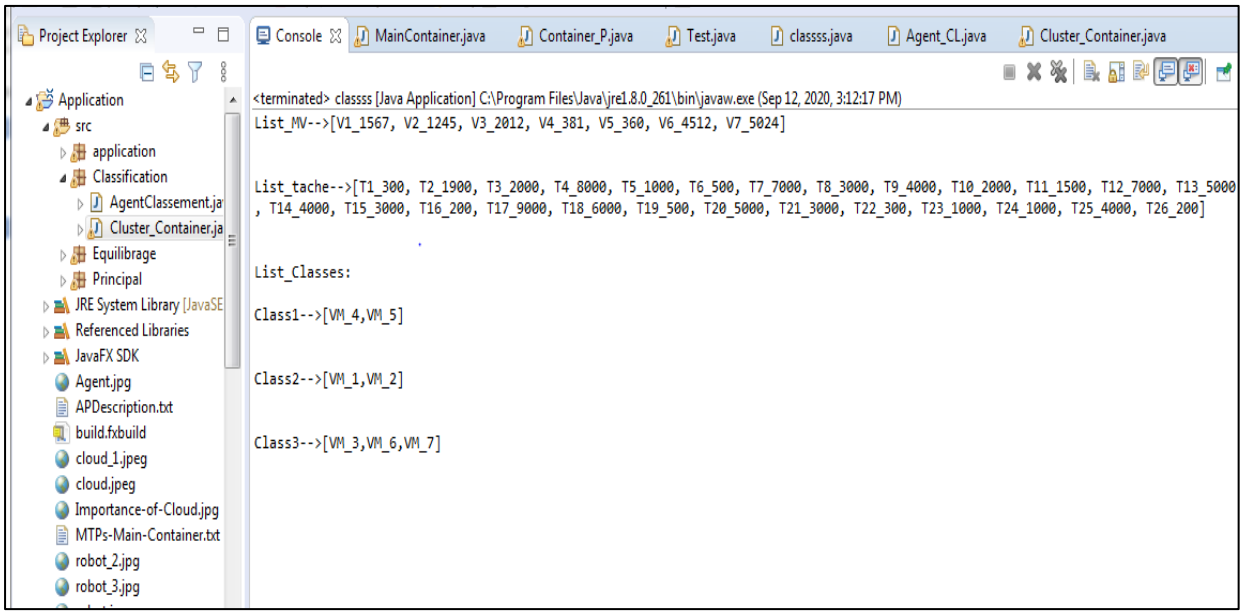


***Figure 5.6 Liste des MV et des tâches et Liste des Classes.***

Pour passer à la simulation, nous avons implémenté le bouton(Equilibrage_Charge), qui permet d'afficher l'interface qui contient les quatre buttons (routage, Agent_SJF, Agent_FCFS, Agent_RR, AgentEQu_General).


***Figure 5.7 Interface des agent d'équilibrage de charge.***

Pour router les taches vers les machines virtuelles nous avons programmé le bouton (Routage).

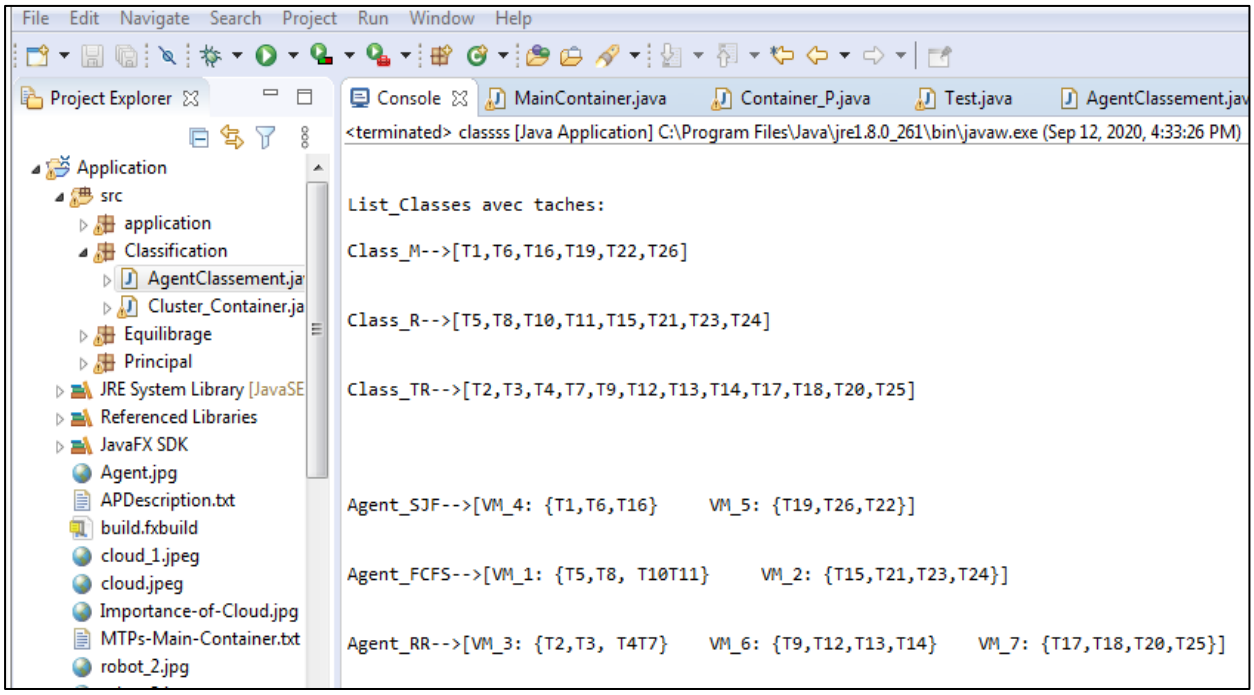


***Figure 5.8 Routage de classes vers les MV.***

Pour afficher les résultats d'ordonnancement des taches nous avons programmé les trois butons (Agent_SJF, Agent_FCFS, Agent_RR).

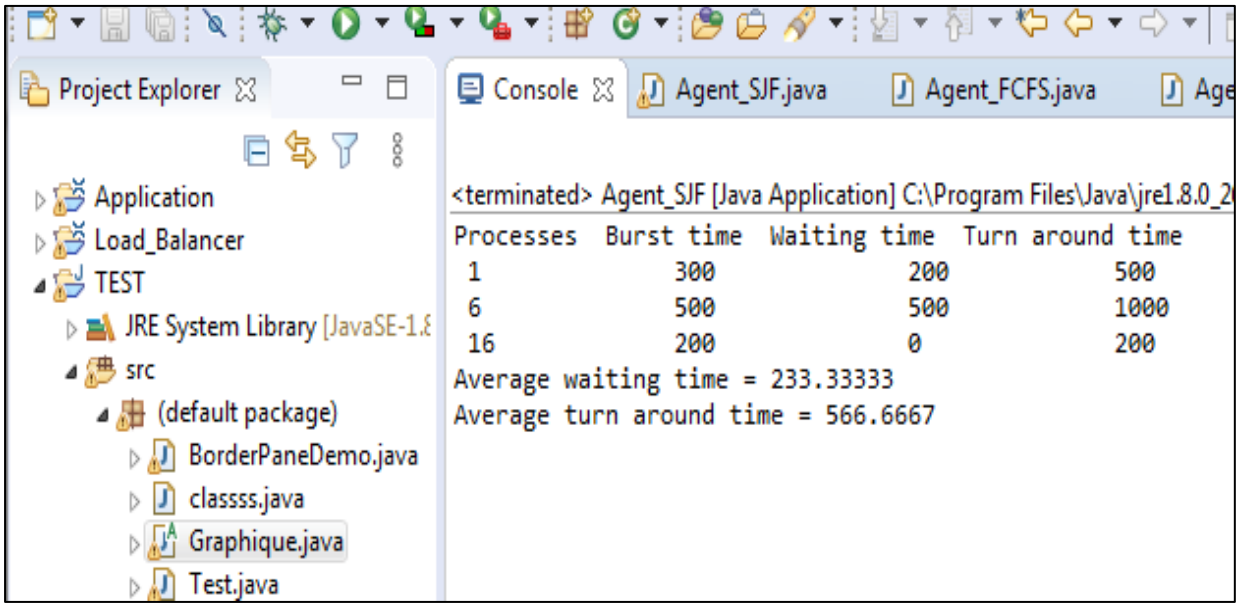


***Figure 5.9 Ordonnancement des taches (Agent_SJF).***

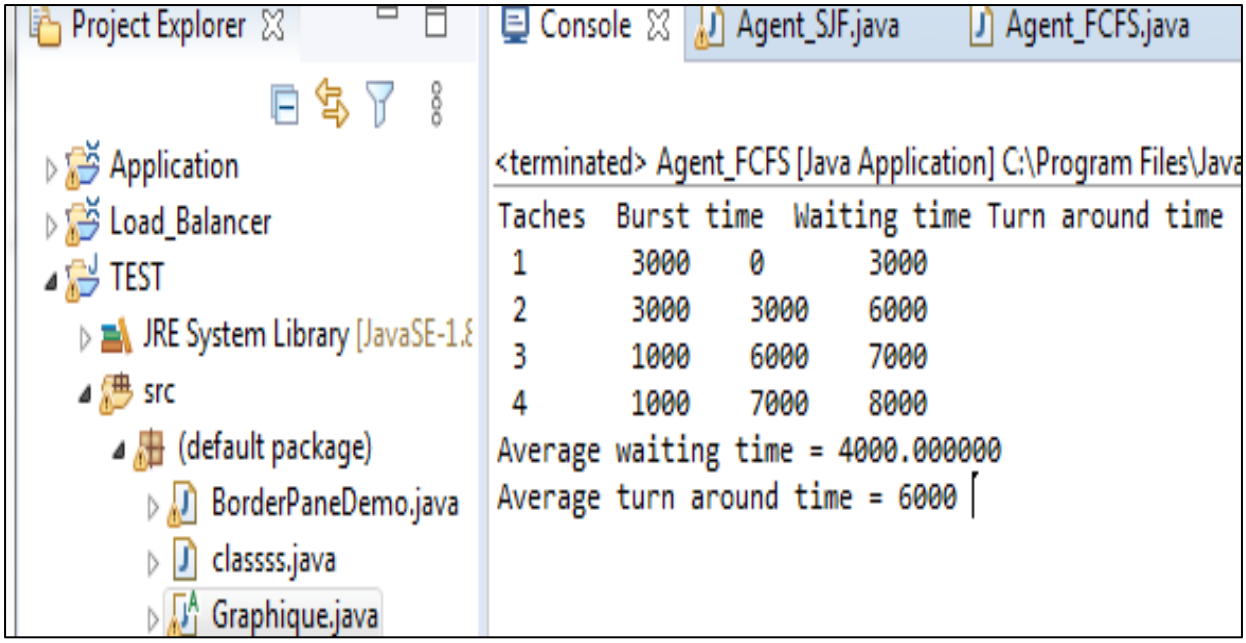


***Figure 5.10 Ordonnancement des taches (Agent_FCFS).***

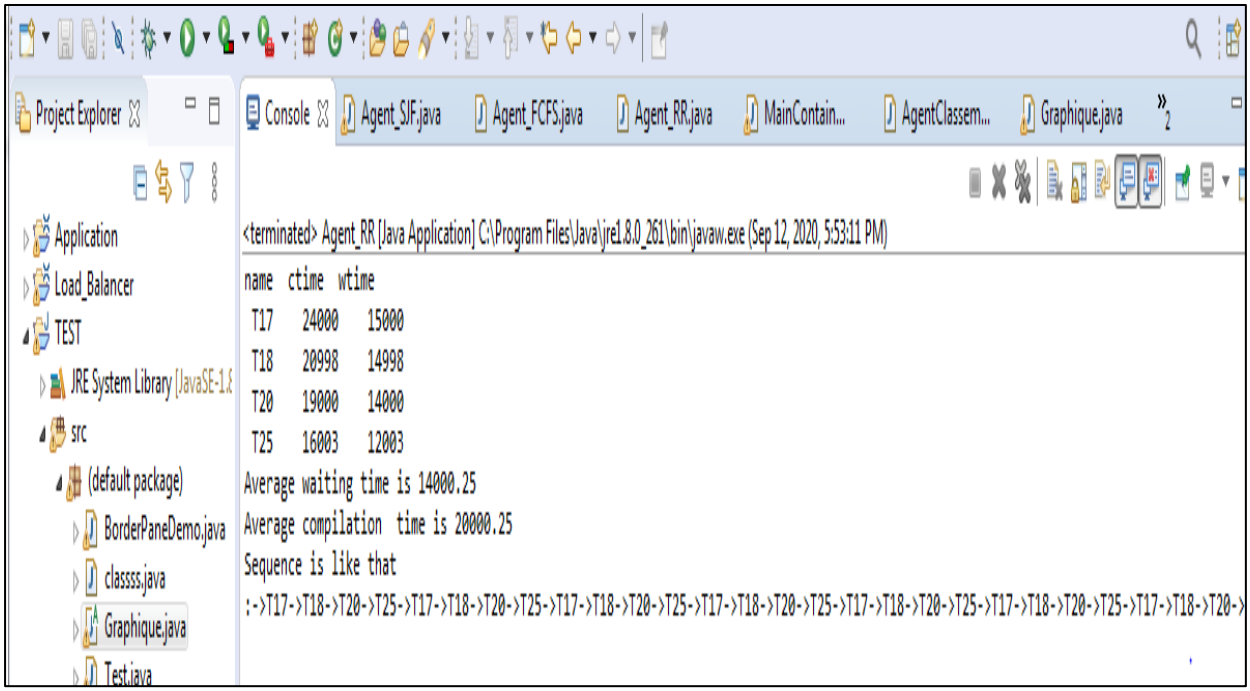


***Figure 5.11 Ordonnancement des taches (Agent_RR).***

Pour l'équilibrage général nous avons programmé le bouton (AgentEQu_Generel)

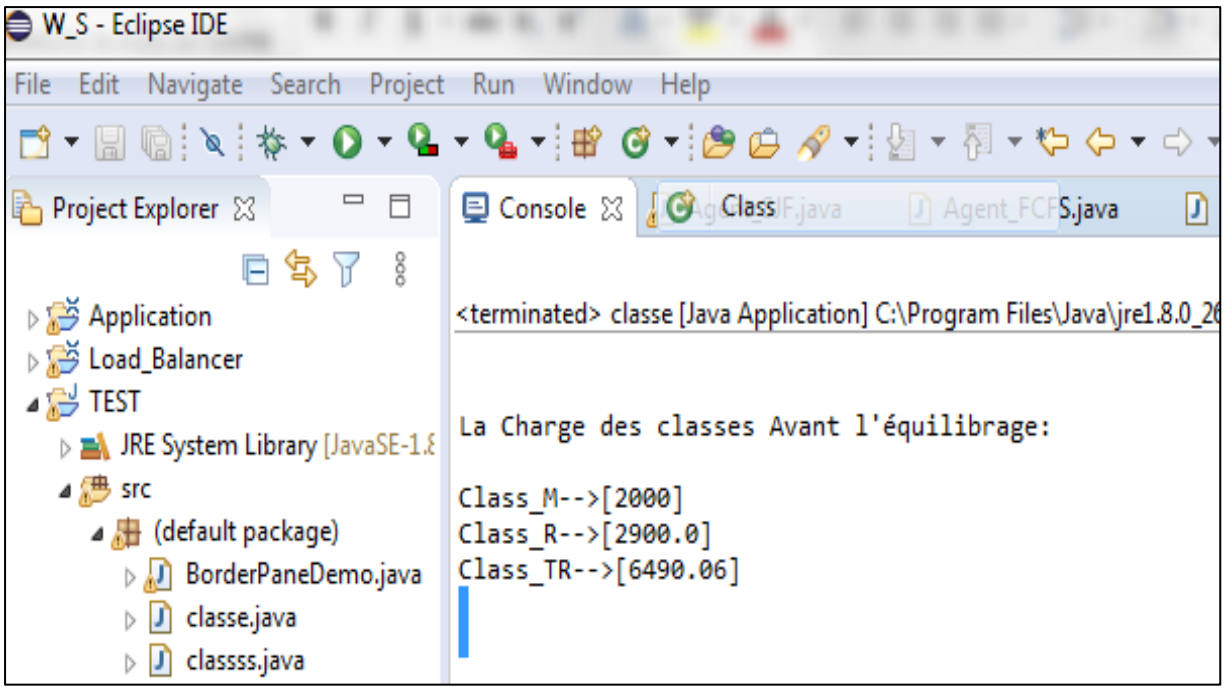


*Figure 5.12 La charge des classes avant l'équilibrage*

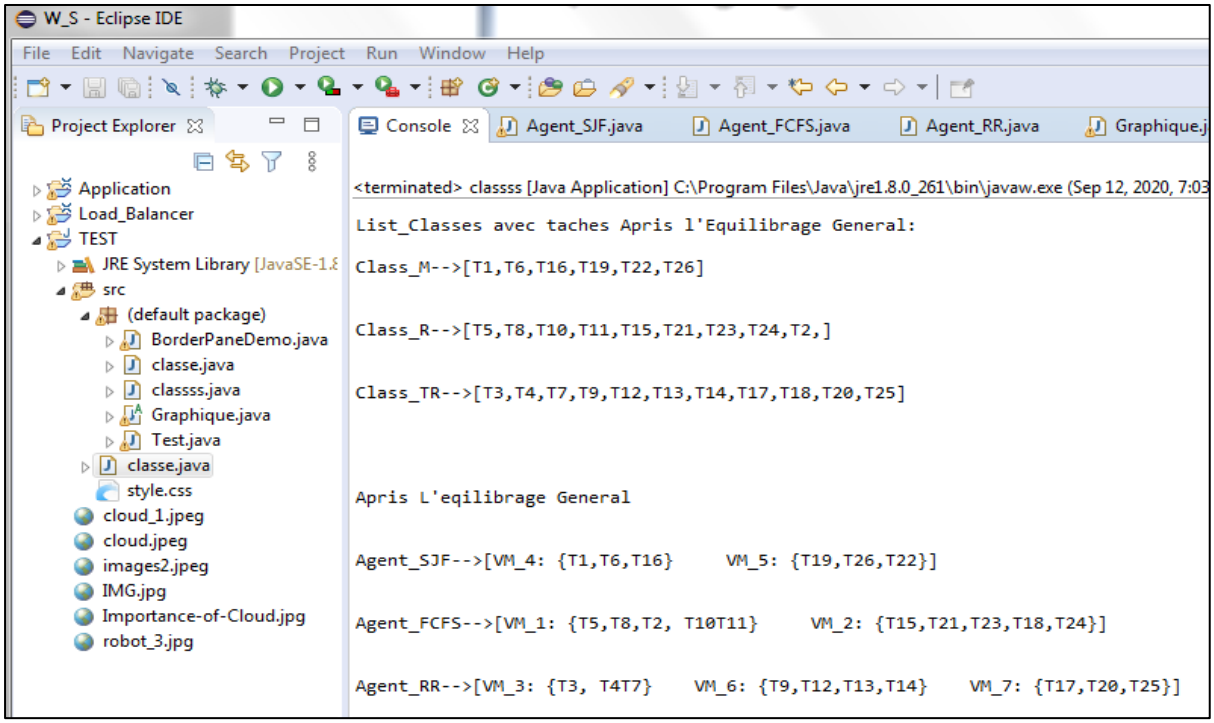


*Figure 5.13 Simulation après L'équilibrage général.*

## 6. Conclusion :

Nous avons présenté dans ce chapitre les différentes fonctions de notre Application et son mode d'utilisation, le but de notre méthode proposée consiste à créer des classes de machine virtuelles en fonction de leur performance en utilisant pour cela la méthode K-MEANS d'une part, et d'équilibrer la charge entre les MV à l'intérieur de chaque classe basée sur les systèmes multi agents, et aussi d'équilibrer la charge entre les classes.

Nous avons validé notre approche en mettant en œuvre les méthodes et les algorithmes proposés dans le chapitre 4. Les résultats obtenus montrent que notre approche permet de faire un équilibrage de charge entre les différents MV dans le contexte du Cloud computing.

# CONCLUSION GÉNÉRAL ET PERSPECTIVES

## Conclusion général et perspectives :

Le Cloud compuitng est l'un des paradigmes computationnels les plus populaires et les plus intéressants, en raison de ses nombreux avantages, Il souffre de plusieurs problèmes parmi lesquelles le problème du déséquilibrage de charge, où on a essayé de proposer une méthode d'équilibrage de charge dynamique basé sur les systèmes multi agents et les algorithmes d'ordonnancement.

Notre modèle permet d'assurer l'équilibrage de charge dynamique entre les différentes machines virtuelles dans le Cloud computing, il est composé de quatre composants essentiels :

**L'agent de classement :** Qui permet de classifier les machines virtuelles selon la vitesse du CPU et la taille de RAM, en trois classes nommées respectivement (très rapide, Rapide, Moyenne).

**L'agent principal** : qui permet de calculer les distances entre la tâche reçue et les centres de gravité des cinq classes, puis il envoie la tâche à la classe la plus proche par rapport aux distances précédemment calculées.

**L'agent d'équilibrage local :** permet d'équilibre la charge entre différentes machines virtuelles, appartenant à la même classe, en tenant compte du pire temps d'exécution d'une tâche et de sa taille. Il fournit trois agents (Agnet_SJF, Agent_FCFS, Agent_RR). De manière décentralisée, et chaque agent fournit des agents de manière centralisée, en fonction du nombre de machines virtuelles.

**L'agent d'équilibrage général :** chargé d'équilibrer la charge entre les classes en migrant les tâches d'une classe surchargée vers une classe la moins chargée.

Comme future travail, nous envisagerons d'améliorer notre modèle en combinant d'autres techniques.

# RÉFÉRENCES BIBLIOGRAPHIQUES


[1] : Mlle. KADDOUR Halima : Mise en œuvre d'une application SMA Dans les réseaux P2P

(Supervision system). Juin 2015. 38

[2] : HARROU, Abdelaziz & Guermoudi, Abdelkader : Développement d'un agent BDI pour la prise de décision d'ordonnancement dans un environnement Cloud Computing**.[en ligne].** Université Abou Bakr Belkaid– Tlemcen Faculté des Sciences Département d'Informatique. Présenté le 23 Juin 2016.72

[3] : L'informatique en nuage IP/A/IMCO/ST/2011-18 Mai 2012.

[4] : Analyses et Synthèses – Les risques associés au Cloud Computing – Autorité du Control Prudentiel (ACP – Banque de France).

[5] : L'informatique en nuage IP/A/IMCO/ST/2011-18 Mai 2012

[6] : Gestion de tolérance aux fautes dans les Cloud Computing avec CloudSIM – Mémoire pour l'obtention d'un mémoire de magister présenté par : Mr.LILAM Said

[7] :Interface de communication pour les réseaux InfiniBand : Projet de fin d'étude présenté par : Aïchatou RABBA - Nadir GHEZALI.

[8] : Migration Alpha – Itanium sous OpenVSM - Auteurs : ABOUNADA Sami, BOUKADIDA Mohamed J, KSOURI Sonia

[ 9] : Le projet européen de grille de calcul (DataGrid) : Concept de grilles de calcul. CHARRON Thierry

[10] : Comparaison et mise en place des plateformes de Cloud Computing : OpenStack et Eucalyptus : Mémoire MASTER ACADEMIQUE - Présenté par : AOUAMEUR Zakia, TAHRINE Halima - Université Kasdi Merbah de Ouargla.

[11] :Abdelhafid Benouda: Cotribution à la conception et à l'implementaiond'un langage de spéfication formelle dédie a la e-maintenance des systèmes de production par l'aprouche des système multi-agent.université farhat abbes setif .2006

[12] : A. El-Fallah Seghrouni et S. Haddad, ≪ A coordination algorithm for multi-agent planning Dans: Proceedings of International Conference on Multi-agent Systems-1996 (ICMAS'96), AAAI, Press, Kyoto, Japan, 1996.

# RÉFÉRENCES BIBLIOGRAPHIQUES

[13] : Georgé, J.-P., Edmonds, B. et Glize, P. (2004). «Making Self-Organizing Adaptive Multi-Agent Systems Work - Towards the engineering of emergent multi-agent systems (chapter 8)». In : Metho- dologies and Software Engineering for Agent Systems. Kluwer, p. 319–338 (cité pp. 12, 14)

[14] : Hübner, J. F. (2003). «Um Modelo de Reorganização de Sistemas Multiagentes». Thèse de doct. Universidade de São Paulo, Escola Politécnica. Hubner-tese.pdf (cité pp. 10, 11). Hübner,J.,Boissier,O.,Kitio,R.etRicc

[15] : Jorquera, T. (2013). «An adaptive multi-agent system for self-organizing continuous optimization».

Thèse de doct. Université de Toulouse (cité pp. 25, 37)

[17] : Benaouda A and Zerhouni N. and C. Varnier, Spare part management for e-maintenance platform, IEEE: Mechatronics & Robotics 2004, pp 1152-1157, Vol III, September 2004, Edited by Paul Drews, Aachen, Germany.

[16] : Duvallet, Claude. Des systèmes d'aide a la décision temps réel et distribués: modélisation par agents. Diss. Université de Paris, 2004.

[18] : Load Balancing auteurs : MASSAOUDI MOHAMED, CHAHINEZ HACHAICHI, AMENI DHAWEFI, ERIJ MAIJED, EMNA BOUGHANMI

[19] : Modèle d'Équilibrage de Charge pour les Grilles de Calcul auteur : YAGOUBI Belabbas.

[20] : Equilibrage de charge pour les grilles de calcul : classe des tâches dépendantes et indépendantes. Auteurs : Meriem Meddeber et Belabbas Yagoubi .

[21] : Modèle Arborescent pour l'Equilibrage de Charge dans les Grilles de Calcul - YAGOUBI Belabbas - Université d'Oran Es Sénia .

[22] : Load Balancing Task Scheduling based on Genetic Algorithm in Cloud Computing Tingting Wang, ZhaobinLiu , Yi Chen, Yujie Xu .

[23] : Contribution à la parallélisation des algorithmes exacts de résolution de contraintes, Thèse de Magister, présentée et soutenue publiquement par SAIDI

Samira- 2013.

# RÉFÉRENCES BIBLIOGRAPHIQUES

[24] : Algorithme de diffusion génétique pour l'équilibrage de charge dans les grilles de calcul. Auteur : DRIF Ahlem.

[25] : Hemam, S.M., O., Hioual, et O., Hioual (2017). Load balancing between nodes in a volunteer Cloud Computing by taking into consideration the number of Cloud services replicas. In: 3rd International Conference of Cloud Computing Technologies and Applica-tions (CloudTech), Rabat, Morocco

[26] : HENNANE Hemza, BELBACHIR Iméne, (2015), Equilibrage de Charge dans Les Systèmes Distribués, Université Abou Bakr Belkaid– Tlemcen,thèse de Master.

[27] : Chien, N.K., et H.D. Loc (2016). Load balancing algorithm based on estimating finish time of services in cloud computing. In: 18th International Conference on Advanced Communication Technology, Pyeongchang, South Korea.

[28] : European Commission. (2013), Definition of a research and innovation policy leveraging Cloud Computing and IoT combination. Tender specification, SMART 2013/0037.

[29] : Nkosi, M., Mekuria, F., (2010). Cloud computing for enhanced mobile health applications. In: Cloud Computing Technology and Science (CloudCom), IEEE.

[30] : Alexander Gluhak, SrdjanKrco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and TahiryRazafindralambo, (2011), A survey on facilities for experimental internet of thingsresearch. Communications Magazine, IEEE, 49(11):58,67.

[31] : Olejnik, R., I. Alshabani, B. Toursel, E. Laskowski, et M. Tudruj (2009). Load balancingmetrics for the SOAJA framework. Scientific International Journal for Parallel and Distributed Computing, Scalable Computing: Practice and Experience, 10 (4): 1-10.

[32] : Kumar, N., (2013). Smart and intelligent energy efficient public illumination system with ubiquitous communication for smart city, (ICSSS), pp. 152–157.

[33] . Olejnik, R., I. Alshabani, B. Toursel, E. Laskowski, et M. Tudruj (2009). Load balancing metrics for the SOAJA framework. Scientific International Journal for Parallel and Distributed Computing, Scalable Computing: Practice and Experience, 10 (4): 1-10.

# RÉFÉRENCES BIBLIOGRAPHIQUES

[34] : Lazarescu, M., (2013). Design of a wsn platform for long-term environmental monitoring for iot applications. Emerging and Selected Topics in Circuits and Systems, IEEE, 45–54.

[35] : Mell, P., et T. Grance (2009). The NIST definition of cloud computing, National Institute of Standards and Technology, version 15.

[36] : Prati, A., Vezzani, R., Fornaciari, M., Cucchiara, R., (2013). Intelligent Video Surveillance as a Service. In: Intelligent Multimedia Surveillance. Springer, pp. 1–16.

[37] : Fischer, A., (2014). Deux méthodes d'apprentissage non supervisé : synthèse sur la méthode des centres mobiles et présentation des courbes principale. Journal de la Société Française de Statistique, 155(2): 2-35.

[38] :Brakni Ilhem : Planification multi-agents pour la composition dynamique Université de Tébessa -algerie - Ingénieur d'état en informatique 2010

[39] : Mell, P., et T. Grance (2009). The NIST definition of cloud computing, National Institute of Standards and Technology, version 15.