الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة محمد خيضر _ بسكرة

**Universite Mohamed Kheider –Biskra**

**Faculté Des Sciences Exactes Et Des Sciences De La Nature Et De La Vie**

**Département D'informatique**

**N° d'ordre :……**
**Série :……..**

# Thèse

Présentée pour l'obtention du grade de **DOCTEUR EN SCIENCES**

En informatique

**Spécialité**: informatique

Par: AL_Mutawakel Abdullah Mohammed

**Sujet**

## Une approche coopérative décentralisée basée agent et CSP pour l'allocation des ressources dans le cloud computing

Soutenue publiquement, le 24 /05/2022,devant le jury composé de :

| | | | |
|---|---|---|---|
| M. Terrissa Labib Saddek | Professeur | Univ. Biskra | Président |
| M. KAZAR Okba | Professeur | Univ. Biskra | Rapporteur |
| M.Laouar Mohamed Ridda | Professeur | Univ. Tebessa | Examinateur |
| M. Amroune Mohamed | M.C.A | Univ. Tebessa | Examinateur |

Année universitaire : 2021-2022

*Une approche coopérative décentralise basée agent et CSP pour l'allocation de ressource dans le cloud computing*

AL_Mutawakel Abdullah Mohammed

*Universite Biskra*

*Laboratoire d'Informatique Intelligente LINFI*

بسم الله الرحمن الرحيم

{وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا}

صدق الله العظيم

# *Dédicace*

Je dédie le fruit de mes études à :

- À mon cher Père, à celui qui m'a comblé par son affection et m'a appris le sens de la responsabilité et le devoir. Que Dieu préserve sa santé, en lui souhaitant une longue vie.
- À ma très chère Maman, à celle qui a veillé sur moi et a tout tant souffert sans me faire souffrir. Que Dieu préserve sa santé, en lui souhaitant une longue vie.
- A ma chère épouse qui m'a soutenu et aidé, je remercie Dieu de m'avoir donné une épouse qui ne m'a pas fait ressentir l'absence de ma famille lors de mon séjour dans mon deuxième pays, l'Algérie.
- À mon cher frère Abdoulkader qui m'a soutenu et motivé, que je considère comme mon deuxième père, à mes sœurs, la lumière de ma vie, Samah Samar et Samia.
- À ceux que je considère comme mes frères : l'ingénieur Ahmed Al-Khatib, le Dr Kazman, le Dr Maqbool Al-Kamil, le Dr Nabil Al-Mawlid, le Dr Bin Abdul-Razzaq, le Dr Kamal Al-Maasar, le Dr Ibrahim Al -Kibsi, et l'ingénieur Hammoud Al-Hashimi.
- À tous ceux que j'aime, à tous ceux qui m'aiment et à tous mes amis et mes connaissances, je dédie le même fruit.

# Remerciements

*Tout d'abord, Mes louanges* ***à DIEU*** *le Tout Puissant pour m'avoir donné le courage, la volonté, la patience durant ces années d'étude et que grâce à Lui ce travail a été réalisé au sein du Laboratoire d'Informatique Intelligente (LIN F I) de l'Université Mohamed Khidr de Biskra sous La direction de* ***Professeur Okba KAZAR.*** *Salutations à mon pays bien-aimé, cher à mon cœur, l'Algérie. L'Algérie est un pays de fierté et d'élévation, le pays qui m'a enseigné et nourri. Si ma nationalité est yéménite, mon cœur est algérien. Ce travail de recherche n'a pas été accompli sans l'aide et le soutien de nombreuses personnes à qui et à travers ces mots, je leur exprime ma profonde gratitude. Mes premiers remerciements iront à mon encadreur* ***Pr. Kazar Okba****, professeur à l'Université Mohamed Kheider de Biskra, pour m'avoir soutenu durant toute la période de la réalisation de cette thèse. J'aimerais lui adresser toute ma profonde gratitude et je le remercie du fond du cœur pour toutes ses remarques et suggestions techniques, académiques et professionnelles. Il a toujours su me consacrer des moments de son temps, me guider, me conseiller, et me témoigner son soutien et sa confiance. Je n'oublierai jamais sa gentillesse et sa bonne humeur avec moi.*

*Je remercie très vivement* ***Mr Terrissa Labib Saddek*** *Professeur à l'université de Biskra pour l'honneur qu'il me fait en acceptant de présider le jury.*

*Mes vifs remerciements vont également aux honorables membres de jury :* ***Mr Laouar Mohamed Ridda*** *Professeur à l'université de Tebessa,* ***Mr Amroune Mohamed*** *Professeur à l'université Tebessa.*

*Ces remerciements vont aussi au corps professoral et administratif du Département d'informatique pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.*

*Un remerciement particulier* ***à mon épouse*** *pour sa patience à sa disponibilité permanente durant la préparation de cette thèse.*

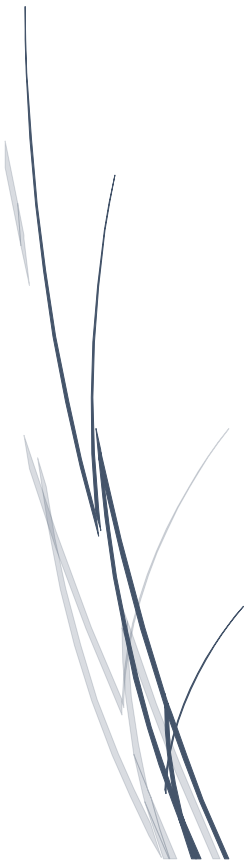
***ALMUTAWAKEL ABDULLAH***

# Table des Matières

# Table des Figures

# Table des Tableaux

# Introduction Générale



## 1. Contexte

Le cloud computing a évolué à travers un certain nombre de phases comprenant le distributed computing, cluster computing, utility computing, grid computing et la fourniture de services applicatifs et logiciels en tant que service (SaaS), mais le concept global de fourniture de ressources informatiques via un réseau mondial est ancré dans les années soixante. En 1963, l'Agence pour les projets de recherche avancée de défense (DARPA) a présenté un projet comprenant le développement d'une technologie répondant aux exigences de l'Institut de technologie du Massachusetts (MIT) permettant à un ordinateur d'être utilisé simultanément par deux personnes ou plus. Dans ce cas, l'un de ces ordinateurs gigantesques et archaïques utilisant des bobines de bande magnétique pour mémoriser était le précurseur de ce qui est maintenant devenu collectivement connu sous le nom de cloud computing.

Le cloud computing est une nouvelle ère de l'informatique à distance basée sur Internet où l'on peut accéder facilement à leurs ressources personnelles à partir de n'importe quel ordinateur via Internet. On peut le considérer un changement radical par rapport à la manière traditionnelle dont les entreprises considèrent les ressources informatiques. Les six courantes raisons pour lesquelles les entreprises se tournent vers les services de cloud computing: le coût, la rapidité, la productivité, le performance, l'échelle globale et la sécurité.

Malgré que l'idée derrière le cloud computing revient aux années soixante mais elle n'était pas applicable à cette période de temps à cause de la limite d'avancement technologiques liés à cette période (comme la vitesse de calcule et la capacité de stockage), ces limites sont disparues au début des années 2000, qui a permet de l'apparition de cloud computing, en utilisant des ordinateurs avec une vitesse de calcule très rapide et une capacité de stockage gigantesque, ces ordinateurs sont appelés Datacenter (Centres de Données). Les Datacenters exploitent le mécanisme de la virtualisation pour créer un nombre infini des ressources virtuelles pour satisfaire les besoins des consommateurs, un d'autre coté, le nombre des utilisateurs de plateforme de cloud computing augmente chaque jour. Cela va mettre le cloud computing en face d'un nouveau défi technologique qui consiste à comment assurer la bonne gestion de ce nombre énorme de ressources dans un environnement très distribués. Pour résoudre ce problème nous allons proposer dans cette thèse : une nouvelle approche coopérative décentralisée pour l'allocation des ressources dans le cloud computing, en fonction sur trois paradigmes : système multi-agents (SMA), contraint satisfaction problem (CSP) et logique floue (LF). Le SMA représente l'infrastructure physique, il permet une gestion efficace des ressources dans la distribution et l'hétérogénéité de cette infrastructure. Le CSP travaille côte à côte avec le SMA pour maintenir les politiques d'allocation des

ressources dans les datacenters, tandis que la LF est utilisée pour faciliter la représentation des valeurs dynamiques des ressources en termes linguistiques (faible, moyen, élevé…) et aide le système à déterminer la meilleure solution en fonction des critères des demandes des clients. L'objectif ici est d'assurer la gestion optimale de ces ressources afin d'assurer la satisfaction totale des utilisateurs. Autrement dit, notre approche fournit une solution aux besoins du client en raccourcissant le temps d'exécution et en réduisant les paiements des ressources demandées qui ont un caractère dynamique.

Les processus d'allocation des ressources représentent l'opération de base pour le système de manipulation des ressources dans un environnement de type cloud. Dans cette approche, l'allocation est le processus qui permet de fournir les ressources qui correspondent aux requêtes du client. Un système d'allocation repose sur quatre étapes :

- La requête de l'utilisateur : les demandes des clients sont présentées en termes d'intervalle de ressources demandées (cloudlet) et de son budget (CL, $) comme dans le Tableau 1.

**Tableau 1. Format des ressources cloudlet demandées.**

| User ID | ClouletId | Ram | | Mips | | Bw | storage | length | Budget |
|---|---|---|---|---|---|---|---|---|---|
| | | minRam | maxRam | minMips | maxMips | | | | maxCost |

- Le système intermédiaire : il a pour rôle d'analyser la requête, chercher et d'allouer les ressources sous forme de machine virtuelle (MV) qui correspondent aux caractéristiques décrites par le client.
- Les ressources cloud : qui a pour rôle de recevoir la requête sous forme de machine virtuelle (MV) et d'héberger les ressources demandées dans le Datacenter après s'être assuré que les meilleures ressources sont obtenues en s'appuyant sur une coopération décentralisée dans le système multi-agents (SMA).
- La logique floue : le rôle de ce processus est de déterminer la meilleure solution à partir d'un groupe de petites solutions cloud dans le même Datacenter en faisant fonctionner la solution locale entre les hôtes.

Dans ce qui suit nous présentons quelques concepts clés de la thèse, et ensuite la problématique, les objectifs du travail, les contributions, et enfin un plan de lecture de la thèse.

## 2. Problématique et Objectifs de la thèse

Les problèmes d'affection de tâches et d'allocation de ressources dans un contexte hétérogène comme le cloud sont des problèmes difficiles [1]. Par conséquent, la théorie d'affection de tâches et d'allocation de ressources dans le cadre du cloud computing suscite une attention croissante avec l'augmentation de la popularité de cloud. Habituellement, l'ordonnancement de tâches est le processus d'affectation des tâches aux ressources disponibles sur la base des caractéristiques et des conditions des tâches. C'est un aspect

important dans le fonctionnement efficace du cloud, car divers paramètres de tâches doivent être pris en considération pour un ordonnancement approprié. Les ressources disponibles devraient être utilisées efficacement sans affecter les paramètres de service du cloud.

En résumé, les problèmes à résoudre et les aspects auxquels nous nous sommes intéressés dans cette thèse sont multiples. Le premier aspect consiste à la soumission de cloudlets (tâches demandées par le client) aux machines virtuelles (MV) dans les hôtes. Pour résoudre ce problème, nous proposons un algorithme d'ordonnancement pour l'allocation des ressources en fonction du coût et de la charge les plus bas. Le deuxième aspect consiste à l'hébergement de nouvelles VM dans les hôtes. Pour résoudre ce problème, nous devons prendre en compte les charges dans les hôtes lors de l'hébergement de nouvelles machines virtuelles dans différents centres de données.

Le but de notre recherche est d'assurer une gestion optimale des ressources cloud afin de répondre aux besoins des utilisateurs avec la quantité de ressources requise avec les coûts les plus bas et les meilleures caractéristiques afin d'obtenir une satisfaction complète des utilisateurs. Cette recherche a un intérêt scientifique dans le domaine de cloud computing et met en évidence l'importance de concept d'allocation de ressources dans l'équilibrage de la charge du datacenter (minimisation de la consommation d'énergie), exploitation efficace des ressources et minimisation du temps d'exécution. Dans ce contexte, nous avons proposé une approche pour gérer l'infrastructure du cloud basée sur un système multi-agents pour l'allocation des ressources et propose des scénarios qui démontrent l'efficacité de l'approche pour la gestion des ressources dans le cloud computing.

## 3. Contributions

Les principales contributions de cette thèse sont de fournir aux utilisateurs les ressources publiées dans l'environnement de cloud computing et de fournir des méthodes qui les aident à choisir la meilleure de ces ressources en fonction de leurs besoins. Autrement dit, notre approche fournit une solution aux besoins du client en raccourcissant le temps d'exécution et en réduisant les paiements des ressources demandées qui ont un caractère dynamique. Dans ce contexte, de nombreuses stratégies sont utilisées pour fournir des ressources aux utilisateurs et elles sont regroupées sous le terme "allocation des ressources".

1) La première et principale contribution est une proposition d'une stratégie de gestion et d'allocutions de ressources cloud qui utilisent la technologie d'agents comme outil de modélisation et d'implémentation des composants du système (Smart and Fuzzy approach based on CSP and MAS for Cloud Resources Allocation).
2) La deuxième contribution est une solution pour gérer l'utilisation de la liste des machines virtuelles libres et la livraison des ressources cloud basée sur les systèmes multi-agents (SMA), qui met en œuvre la sélection, et tri des ressources cloud, tout en prenant en considération toutes les caractéristiques de ces ressources.

3) La troisième contribution est la proposition d'un ensemble de contraintes (CSP) sous forme de modèles mathématique pour assurer la gestion des ressources logiques et virtuelles qui prennent en compte la gestion des machines virtuelles, la gestion des unités de traitement (CPU) et la gestion des espaces de stockage.
4) Enfin, notre quatrième contribution a été pour développer une stratégie basée sur la logique floue (LF). L'approche proposée inclut la logique floue pour deux raisons: premièrement, les ressources requises des demandes des clients (RAM, CPU, budget…) qui ont un caractère dynamique. Ils se manifestent sous forme d'intervalles (par exemple, ram: [min, max], paiement: [min, max], cpu: [min, max]). Deuxièmement, le système détermine la meilleure solution en fonction de plusieurs critères (RAM, Mips, paiement) qui nécessitent un taux d'utilisation (score) pour sélectionner la meilleure réponse.

## 4. Plan et Structure de la thèse

Après une conclusion générale décrivant le contexte du travail, la problématique et contribution, la présente thèse est organisée comme suit: les deux premiers chapitres définissent le cadre général et posent le socle bibliographique de notre travail de recherche tandis que les deux autres chapitres sont consacrés à nos contributions et ses réalisations.

### Chapitre 1 : Cloud Computing

Chapitre 1 : est une synthèse bibliographique sur le cloud computing compris de tous les concepts en relation avec ce dernier pour former une sorte d'état de l'art sur le cloud computing, ce chapitre est considéré comme une introduction à le problème traité dans cette thèse.

### Chapitre 2 : Allocation de ressources dans Cloud Computing

Chapitre 2 : Dans ce chapitre, nous allons donner un aperçu de l'allocation des ressources dans le cloud computing, ainsi que les algorithmes et les stratégies utilisés pour accomplir cette tâche. Et on va terminer le chapitre par une étude et analyse d'un ensemble des travaux intérieurs qui traitent le même problème.

### Chapitre 3 : Système intelligent pour la gestion des ressources *(SFACMCR)*

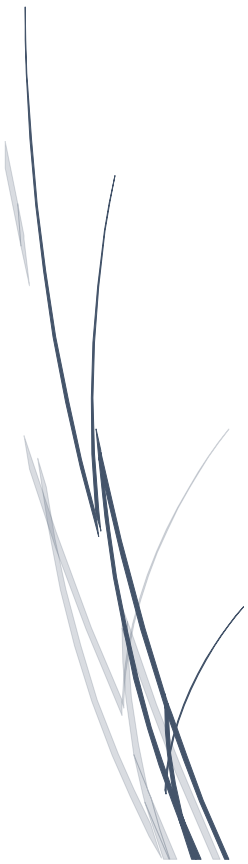
Chapitre 3 : Au début ce chapitre, nous présentons les composants de l'architecture proposée. Ensuite, nous présentons le fonctionnement et les interactions générales entres les différents composants. Enfin, nous présentons les principaux comportements d'agents ainsi que les mécanismes de gestion de ressources proposées dans ce travail.

## Chapitre 4 : Implémentation et résultats d'expérimentation

Chapitre 4: Le quatrième chapitre sera consacré à l'implémentation de notre approche, pour cela, l'environnement de développement, les choix techniques, la présentation un exemple appliqué et les résultats obtenus seront tous présentés dans ce chapitre.

## Chapitre I.

# État de l'art sur le cloud computing



## I.1 Introduction

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

Dans le présent chapitre nous allons présenter un état de l'art sur le cloud computing en présentant tous ce qui est considéré important dans ce domaine et aperçu sur l'intégration des agents et le cloud computing. Nous focalisons notre étude sur l'allocation des ressources dans le cloud computing parce que c'est le sujet de cette thèse.

## I.2 Définitions et concepts du cloud computing

### I.2.1 Définition du cloud computing

Il n'y pas une définition standard de ce terme qui est accepté par tous les acteurs liés avec le cloud computing. C'est pour cela Vaquero et ces collègues [2] ont essayé de standardiser la définition de cloud en étudiant et analysant 22 définitions qui existent dans la littérature. Le résultat de cet effort génère la définition suivante ;

"Les clouds sont une grande réserve de ressources virtuelles faciles à accéder et à utiliser (comme les matériels, les plateformes de développement). Ces ressources peuvent être reconfigurer dynamiquement pour s'adapter à des situations variables, permettant aussi une utilisation optimale de ces ressources. Typiquement cette réserve de ressources est exploitée par un modèle de paiement à l'utilisation dont laquelle des garanties sont offertes par le fournisseur de l'infrastructure" [2].

Une autre définition très intéressante proposé par George Reese [3] dans laquelle l'auteur propose un moyen pour différencier entre un service ordinaire et un service de cloud :

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

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

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

Nous définissant trois critères que nous utilisons dans les discussions pour savoir si un service donné est un service cloud :

- Le service est accessible via un navigateur Web (non propriétaire) ou une API de services Web ;
- Zéro dépense en capital est nécessaire pour commencer ;
- Vous ne payez que ce que vous utilisez comme vous l'utilisez.

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

"Le cloud computing est un modèle Informatique qui permet un accès facile et à la demande par le réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service"[4].

D'autre part, il y a une question qui est posée par ceux qui s'intéressent au cloud computing.

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

Selon [5]. La réponse à cette question peut être clarifié par trois point :

- Le premier est la capacité à exploiter des composants de différentes ressources de cloud et combinez les solutions que vous recherchez. Vous pouvez tirer parti de storage comme un service d'un fournisseur, la base de données en tant que service d'un autre, et même une plate-forme complète de développement et de déploiement d'applications à partir d'un troisième. Cette capacité à exploiter uniquement les ressources dont vous avez besoin parmi les solutions que vous souhaitez proposer, ainsi que dans les quantités appropriées, constitue une valeur évidente du cloud computing moderne.
- Deuxièmement, la banalisation de la bande passante permet aux entreprises de tirer une partie des ressources informatiques en nuage comme si elles étaient locales. Ainsi, vous pouvez tirer parti les ressources de stockage et d'exécution comme si

elles existaient dans votre datacenter, quelque chose qui était difficile il y a quelques années.

- Enfin, il y a la disponibilité de fournisseurs de cloud computing très innovants. Bien que l'architecture et le modèle du cloud computing ne soient pas nouveaux, les acteurs du cloud computing qui fournissent les services sont y compris l'infrastructure- des acteurs en tant que service, tels que l'EC2 d'Amazon et la plate-forme en tant que service des joueurs tels que Google App Engine. Avec le cloud computing de plus en plus par à pas de géant, des services de cloud computing de meilleure qualité et plus innovants sont en cours de construction et libéré en permanence.

### I.2.2 Les caractéristiques de cloud computing

Le cloud computing modèle définit par le NIST contient essentiellement cinq caractéristiques, à savoir :

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

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

❖ **Large accès au réseau**

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

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

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

❖ **Redimensionnement rapide (élasticité)**

En fonction de la demande, les ressources et les capacités peuvent être vendues rapidement et même dans certains cas automatiquement, provisionnées et libérées élastiquement. Pour le consommateur, les capacités disponibles pour l'approvisionnement semblent souvent illimitées et peuvent être appropriées à n'importe quelle quantité à n'importe quel moment.

❖ **Paiement à l'usage**

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

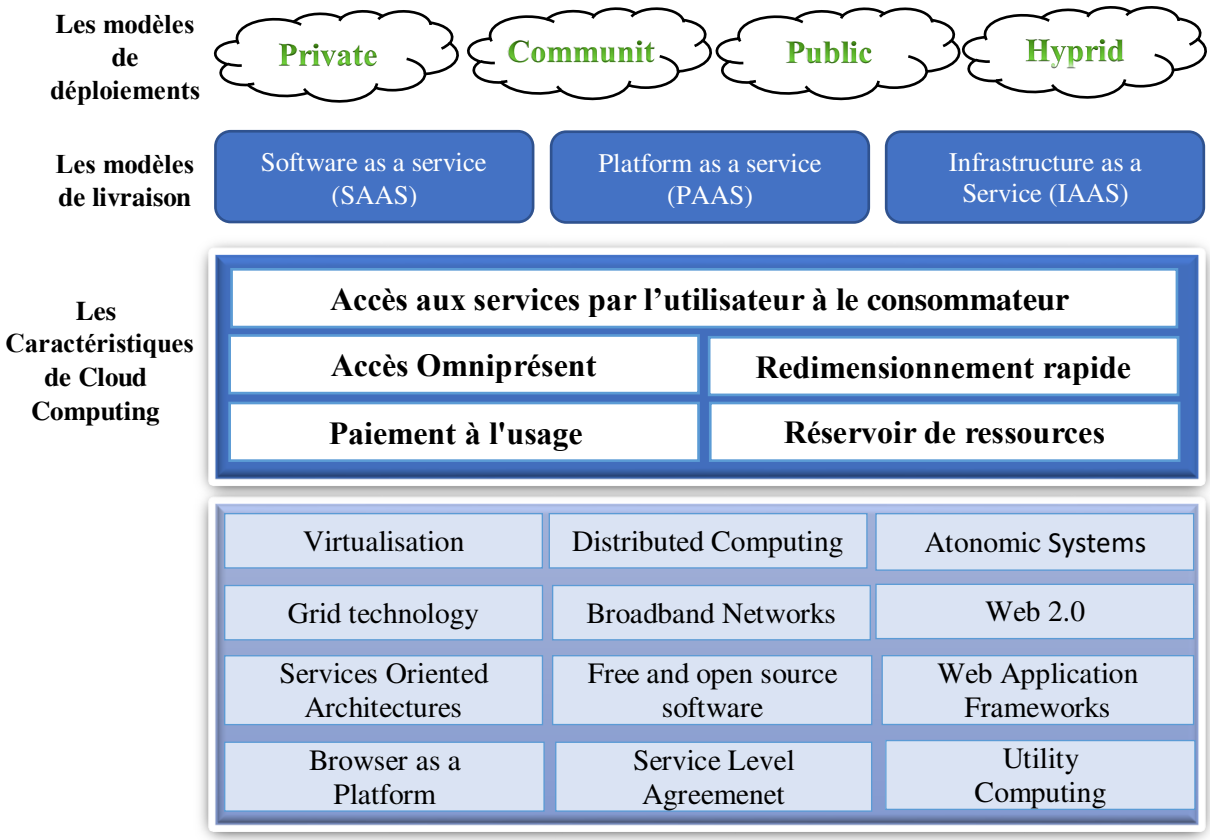



**Figure I. 1 Le modèle de cloud Computing** [6].

### I.2.3 Les modèles de livraison

Dans le cloud computing les fonctionnalités sont offertes aux consommateurs comme des services. Ces services peuvent être devisés selon la nature du service livré en trois grands groupes (cf. Figure I.1), à suivre :

### I.2.4 Software as a Service (SAAS)

Ce sont des services pour les clients de cloud. Ces services peuvent être des applications traditionnelles et simples comme le traitement de texte. C'est le modèle de livraison de service le plus utilisé dans le cloud, tel que SaaS qui permet aux consommateurs d'exploiter des applications à distance du cloud [4]. On retrouve dans cette catégorie plusieurs domaines et exemples d'utilisation (messagerie, CRM, GED des logiciels de gestion des documents, Collaboration en ligne, Sauvegardes en ligne). Il y a aussi des exemples d'offres commerciales comme (Google Apps, Salesforce, WebEx, ZOHO, Cisco).

### I.2.5 Platform as a Service (PaaS)

PaaS permet aux consommateurs de développer et déployer sur une infrastructure cloud des applications en utilisant des langages et outils de programmation fournis par le fournisseur.

Le consommateur ne gère pas et ne contrôle pas l'infrastructure interne du cloud tel que le réseau, les serveurs, les systèmes d'exploitation, le stockage, mais il a un control sur les applications déployées [4].

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

### I.2.6 Infrastructure as a Service (IaaS)

C'est la possibilité pour les consommateurs d'utiliser des ressources liées aux traitement, le stockage, les réseaux et d'autres ressources informatiques fondamentales. Le consommateur sera capable de déployer et d'exécuter des logiciels arbitraires, ce qui peut inclure des systèmes d'exploitation et des applications. Le consommateur ne gère ni ne contrôle l'infrastructure de cloud, mais il contrôle le système d'exploitation, le stockage et les applications déployées avec un contrôle limité des composants du réseau sélectionnés (par exemple, des pare-feu hôtes) [4].

Avec le modèle IaaS le fournisseur peut contrôler l'augmentation, la réduction et l'accès aux ressources de l'infrastructure [7].

Le Tableau I.1 présente les avantages et les inconvénients de chaque service cloud :

**Tableau I. 1 Avantages et Inconvénients des services cloud [8].**

| SERVICE CLOUD | Avantage | inconvénient |
|---|---|---|
| **SaaS** | -pas d'installation<br>-plus de licence<br>-migration | -logiciel limité<br>-sécurité<br>-dépendance des prestataires |
| **PaaS** | -pas d'infrastructure nécessaire<br>-pas d'installation<br>-environnement hétérogène | -limitation des langages<br>-pas de personnalisation dans la configuration<br>des machines virtuelles |
| **IaaS** | -administration<br>-personnalisation<br>-flexibilité d'utilisation | -sécurité<br>-besoin d'un administrateur système |

### I.2.7 Les modèles de déploiement

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

### I.2.8 Le cloud Privé

L'infrastructure du cloud privé est provisionnée pour une utilisation exclusive par une seule organisation avec plusieurs consommateurs. Le cloud privé peut appartenir, être géré et être exploité par l'organisation, par un tiers ou par une combinaison de ces derniers, et il peut exister à l'intérieur ou à l'extérieur des locaux de l'organisation [4].

### I.2.9 Le cloud public

L'infrastructure de cloud public est accessible par l'Internet. Elle est ouverte au public ou à de grands groupes industriels. Il peut appartenir, être géré et être exploité par une entreprise de commerce, par une organisation académique, ou par une organisation gouvernementale. Il existe sur les lieux du fournisseur du cloud public [4].

### I.2.10 Le cloud communautaire

L'infrastructure du cloud communautaire est provisionnée pour une utilisation exclusive par une communauté spécifique de consommateurs qui partagent les mêmes domaines d'intérêts. Il peut appartenir, être géré et être exploité par une organisation ou plus dans la communauté, par un tiers ou par une combinaison d'eux [4].

### I.2.11 Le cloud hybride

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

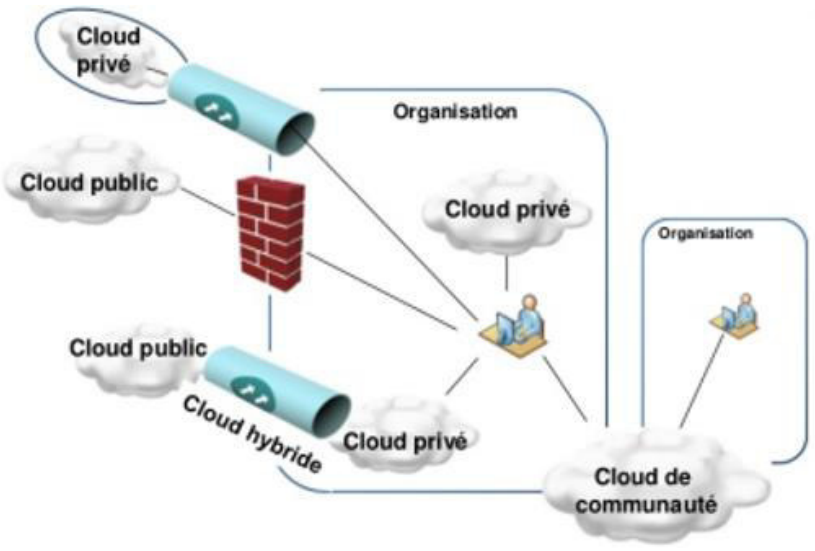



**Figure I. 2 Les modèles de déploiement dans le Cloud computing [4].**

### I.2.12 Un modèle en couches de cloud computing

D'une manière générale, l'architecture d'un environnement cloud peut être divisé en 4 couches: le matériel (couche de Datacenter), la couche infrastructure, la couche plate-forme et la couche d'application, comme indiqué sur la Figure.I.3 [9].

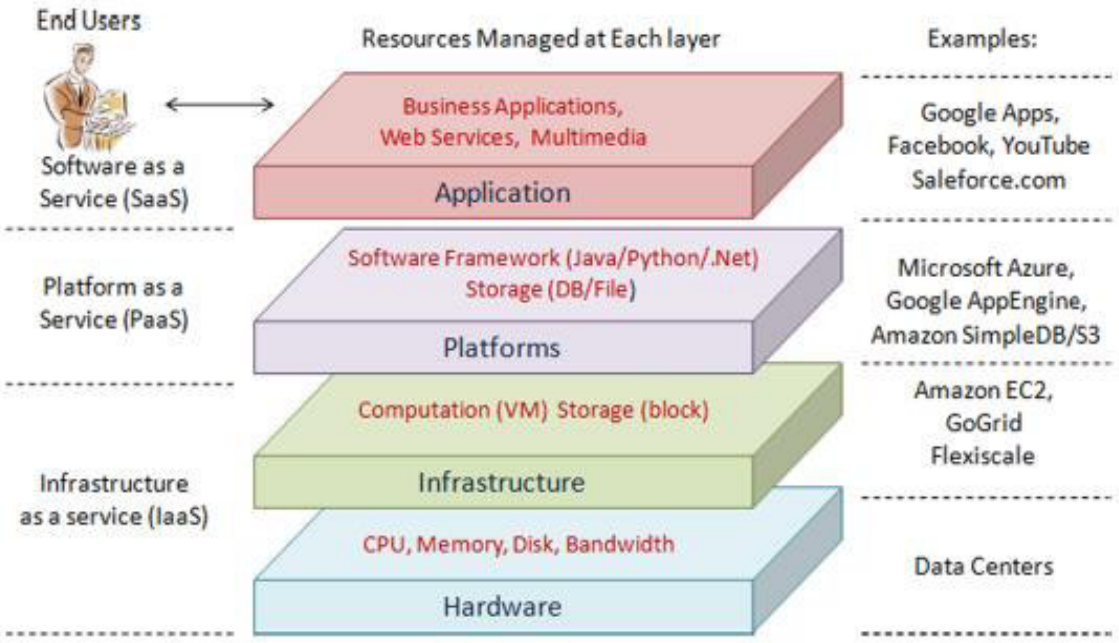



**Figure I. 3 L'architecture d'un environnement cloud** [9].

- **La couche matérielle** : cette couche est responsable de la gestion des ressources physiques du cloud y compris les serveurs physiques, les routeurs, les commutateurs, les systèmes d'alimentation et de refroidissement. En pratique, la couche matérielle est généralement implémentée dans les Datacenters. Un Datacenter contient généralement des milliers de serveurs organisés en racks et interconnectés à travers des commutateurs, des routeurs ou d'autres tissus. Les problèmes fréquemment rencontrer à la couche matérielle comprennent la configuration matérielle, la tolérance aux pannes, la gestion du trafic, la gestion de l'énergie et refroidissement des ressources.
- **La couche infrastructure:** aussi appelée la couche de la virtualisation, la couche infrastructure crée un pool de stockage et des ressources informatiques en partitionnant les ressources physiques e, utilisant des technologies de virtualisation telles que Xen [10], KVM [11] et VMware [12] . La couche infrastructure est un composant essentiel de cloud computing, car de

nombreuses fonctionnalités telles que l'affectation dynamique des ressources sont disponibles via les technologies de virtualisation.

- **La couche de plate-forme** : construite sur la couche infrastructure, la couche plate-forme se compose de systèmes d'exploitation et cadres d'application. Le but de la couche de plate-forme est de minimiser la charge de déploiement d'applications directement dans des conteneurs VM. Par exemple, Google App Engine fonctionne au niveau de la couche plate-forme pour fournir un support API pour la développent des applications cloud.
- **La couche d'application** : au niveau le plus élevé de la hiérarchie, la couche application se compose des applications cloud proprement dites. Différentes des applications traditionnelles, les applications cloud peuvent tirer parti de la fonction de mise à l'échelle automatique pour obtenir de meilleures performances, une meilleure disponibilité et une réduction des coûts d'exploitation. Par rapport aux environnements d'hébergement de services traditionnels tels que les fermes de serveurs dédiés, l'architecture du cloud computing est plus modulaire.

## I.3 Les concepts de cloud datacenter

Dans cette sous-section nous allons présenter les concepts techniques de Data Center dans le cloud.

### I.3.1 Les Datacenter

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

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

l'environnement pour maintenir la température et humidité appropriées pour les équipement électronique [13].

### I.3.2 L'architecture de Datacenter

Le datacenter est l'hébergement de la puissance de calcul, le stockage et les applications nécessaires pour prendre en charge une entreprise. L'infrastructure du datacenter est au cœur de l'architecture informatique, à partir de laquelle tout le contenu provient ou transite. Une bonne planification de la conception de l'infrastructure du datacenter est essentielle, et les performances, la résilience et l'évolutivité doivent être soigneusement prises en compte.

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

La conception du réseau de centres de données repose sur une approche en couches éprouvée, qui a été testée et améliorée au cours des dernières années dans certaines des plus grandes implémentations de centres de données au monde. L'approche en couches est la base de la conception du datacenter qui vise à améliorer l'évolutivité, les performances, la flexibilité, la résilience et la maintenance. La figure I.4 illustre la conception en couches de base [14].

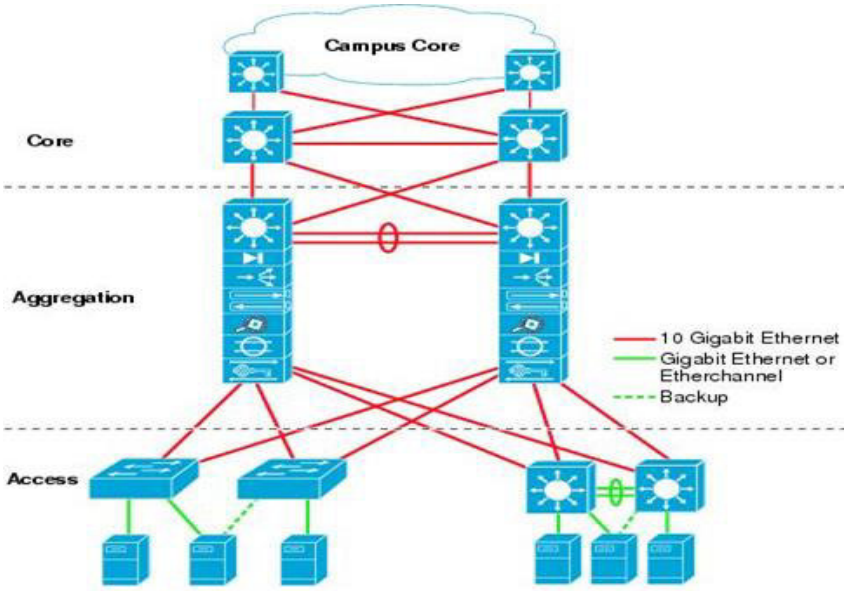



**Figure I. 4 L'architecture de Datacenter** [14].

### I.3.3 La virtualisation

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

Le but de la virtualisation c'est de cacher les caractéristiques physiques des ressources informatiques afin que les autres systèmes, les applications ou les utilisateurs puissent interagir avec ces ressources. Un système d'exploitation principal appelé « système hôte » est installé sur un serveur physique unique, il sert à accueillir d'autres systèmes d'exploitation. L'utilisation des ressources disponibles sur le serveur sont optimisées via un composant appelé hyperviseur ou bien « VMM ». L'hyperviseur gère l'allocation de ressources pour chaque machine virtuelle (CPU, bande passante et stockage) [15] [16].

❖ La virtualisation repose sur trois éléments importants :

- L'abstraction des ressources informatiques.
- La répartition des ressources par l'intermédiaire de différents outils, de manière à ce que celles-ci puissent être utilisées par plusieurs environnements virtuels.
- La création d'environnements virtuels.

❖ Les entreprises de la virtualisation attendent :

- La réduction du nombre de serveurs.
- La réduction de l'espace occupé dans les datacenters.
- La réduction de la consommation énergétique des datacenters.
- Réduction des coûts d'administration.
- Amélioration de la flexibilité et de la rapidité des services.
- Amélioration de la qualité de services.

### I.3.4 Hébergement de machine virtuelle dans l'hôte

Un hôte est capable d'héberger plusieurs machines virtuelles avec des spécifications de ressources différentes potentielles et des types de charges de travail variables. Les serveurs hébergeant des machines virtuelles hétérogènes avec des charges de travail variables et imprévisibles peuvent entraîner un déséquilibre dans l'utilisation des ressources, ce qui entraîne une détérioration des performances et une violation du contrat de niveau de service (SLA) [17]. Pour résoudre ce problème, nous proposons une solution flexible en utilisant deux techniques : Le Système Multi-Agents (SMA) et CSP où l'hébergement de nouvelles VMs dans les hôtes des différents datacenters en fonction de leurs charges.

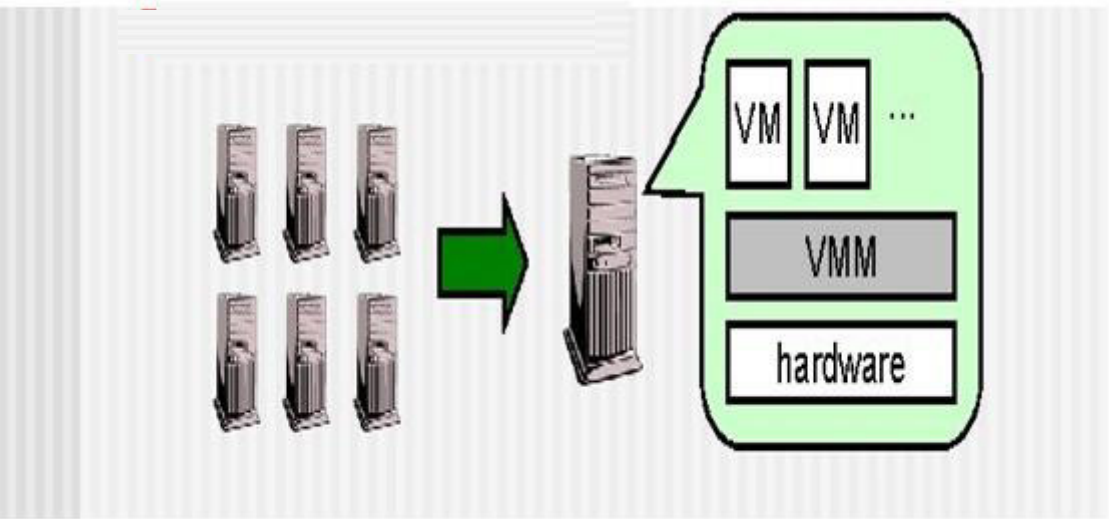



**Figure I. 5 Hébergement de machine virtuelle dans l'hôte** [18].

## I.4 Gestion des ressources dans le cloud computing

Dans le cloud computing, une ressource peut être tout service pouvant être consommé par les utilisateurs, Les ressources dans le cloud computing sont gérés par les fournisseurs de cloud computing, et comme le cloud computing est basé sur « utility computing », les auteurs de [19] classent les ressources du cloud en fonction de leur utilité. Le Tableau I.2 donne un aperçu vu sur la classification des ressources dans le cloud computing.

**Tableau I. 2 classification des ressources dans le cloud computing**

| | | | |
|---|---|---|---|
| **Classification of Cloud Resource** | Fast Computation | Processor | |
| | | Memory | |
| | | Algorithms | |
| | | Operating System | |
| | | APIs | |
| | Storage | Hard Drive | |
| | | Flash drive | |
| | | Softwares like Hadoop | |
| | | Database Servers | Application<br>Web<br>Proxy |
| | Communication | Physical | Intermediate Device<br>Hosts / Workstation<br>Sensors<br>Communication Link |
| | | Logical | Bandwidth<br>Delay<br>Protocols<br>Communication Link |
| | Power / Energy | Cooling Devices | |
| | | UPS | |
| | Security | Trust | |
| | | Authentication | |
| | | Integrity | |
| | | Privacy | |
| | | Availability | |

- L'utilité de calcul rapide (Fast Computation Utility) : ce type de ressources fournit une utilité de calcul rapide dans un environnement cloud computing. À travers l'utilité de calcul rapide, le cloud computing fournit le Calcul en tant que service (Computation As A Service CaaS). L'utilité de calcul rapide incluant la capacité de traitement, la taille de mémoire, des algorithmes efficaces, etc.
- Utilité de stockage : au lieu de stocker des données sur un périphérique de stockage local, nous les stockons sur un périphérique de stockage situé à un emplacement distant. Utilité de stockage est composé de milliers de disques durs, clés USB, serveurs de bases de données, etc. et comme les systèmes d'ordinateurs sont voués à échouer au fil de temps, la redondance des données temporelles est requise ici. Grâce à une utilité de stockage, le cloud computing fournit le stockage en tant que service (StaaS).
- Utilité de communication : elle peut également être appelé l'utilitaire de réseau ou réseau en tant que service (Network As A Service NaaS). L'utilité de calcul rapide et de stockage ne peut être réalisé sans l'utilité de communication. L'utilité de communication consiste de ressources physiques (périphériques intermédiaires, hôtes, capteurs, liaison de communication physique) et logiques (bande passante, délai, protocoles, liaison de communication virtuelle). Dans le cloud computing, chaque service est fourni via l'Internet.
- L'utilité d'énergie (Power / Energy Utility): des chercheurs aujourd'hui effectuent de nombreux travaux de recherche sur les techniques d'économie d'énergie dans le cloud computing. Le coût d'énergie peut être grandement réduit par une utilisation conscience d'énergie. A cause de l'existence de milliers de serveurs de données, la consommation électrique de ces serveurs est très élevée dans le cloud. Les appareils de refroidissement et les onduleurs sont au centre de ce type de ressources. Ils peuvent également être considéré comme une ressource secondaire.
- Utilité de sécurité : la sécurité est toujours un problème majeur dans toute zone informatique. En tant qu'utilisateur cloud, nous voulons un service très fiable, sûr et sécurisé.

### I.4.1 Les ressources

Cette section est dédiée aux différentes propriétés des ressources qui sont l'objet de négociation dans les problèmes d'allocation de ressources.

- Continues vs Discrètes

La ressource peut être allouée sur une échelle dénombrable ou infiniment dénombrable si on a affaire à une ressource de type discret, comme par exemple une pomme où la plus petite

unité indivisible lors du marchandage est une pomme elle-même. Par contre, une ressource continue est une ressource allouée sur une échelle non dénombrable comme par exemple l'énergie. Une représentation d'une allocation de ressources de type continue peut-être représentée par un vecteur de réel positifs (ou, alternativement, des numéros compris dans l'intervalle [0,1] qui désignent la proportion d'une ressource particulière détenue par l'agent recevant le paquet de ressources). D'un autre côté, les paquets de ressources discrètes peuvent être représentées par un vecteur d'entiers non-négatifs ou par un vecteur binaire comprenant des valeurs {0, 1} s'il y a au plus une ressource dans le marché pour chaque type de ressource. Les méthodes pour les problèmes d'allocation de ressources de type discret sont applicables aux types continus après discrétisation. La discrétisation d'une ressource continue se fait par la désignation de la plus petite unité que cette ressource peut être divisée lors de son marchandage. Par exemple, dans le cas où 10.000 litres de jus seront mis pour négociation, et après discrétisation de cette ressource, l'unité de cette ressource sera spécifiée à 50 litres, alors, ni l'acheteur, ni le vendeur ne peuvent se permettre de négocier une quantité de cette ressource qui n'est pas multiple de l'unité de cette dernière : 100 litres est accepté, 120 litres n'est pas accepté comme quantité à négocier [20].

- ❖ Divisibles ou non

Le fait que la ressource soit discrète ou continue relève de la propriété inhérente de la ressource, ce qui n'est pas le cas pour la propriété de la divisibilité de la ressource : divisible ou indivisible. La spécification de cette propriété de la ressource est spécifiée au niveau du mécanisme d'allocation de ressource. Dans cette thèse, nous nous concentrons seulement sur les ressources indivisibles [20].

- ❖ Partageables ou non

On dit qu'une ressource est partageable lorsqu'elle peut être partagée ou utilisée par plusieurs agents en même temps. Un exemple d'application est l'observation de la terre par satellite, où l'image satellitaire est partagée par plusieurs agents. Le cas canonique, considère que la ressource est non partageable et nous nous conformons à ce principe dans le reste de cette thèse [20].

- ❖ Statiques ou non

Une ressource est dite statique lorsque son statut reste inchangé. En général, les ressources ne peuvent pas être considérées comme statiques, mais dans les systèmes multi-agent elles sont supposées l'être. Les ressources non-statiques sont des ressources dont leurs statuts varient avec le temps. C'est des ressources consommable et périssable comme les carburants et la nourriture. Dans le reste de cette thèse nous nous conformons aux ressources statiques[20].

❖ Mono-Item vs Multi-Items

Une configuration dîtes multi-items est une configuration où chaque type de ressource est fournie en un ou plusieurs exemplaires. Par contre dans une configuration mono-item, chaque type de ressource ne représente qu'une seule ressource. En fait, tout problème Multi-Item peut être transformé en un problème mono-item en catégorisant les ressources appartenant à la même catégorie (type de ressource). D'un autre côté, tout problème mono-item peut être transformé en un problème multi-item, car une configuration mono-item est un type de configuration multi-item dégénéré. Dans la configuration multi-items, il est possible d'avoir plusieurs ressources de même type et de s'y référer en utilisant le même nom. Supposons, par exemple, qu'il existe un certain nombre de bouteilles de limonades disponibles dans le système. Dans une configuration multi-item, le concepteur va assigner l'ensemble des limonades à une ou plusieurs catégories, et donc les agents ne pourront distinguer individuellement chaque bouteille appartenant à la même catégorie. D'autre part, dans une configuration Mono-item chaque élément doit être distinguable. A cet effet, chaque bouteille doit porter un nom unique afin qu'elle soit identifiable. En optant pour une représentation multi-item on gagne en lisibilité et en compacité mais d'un autre côté, le langage devient plus riche car le domaine des variables sera étendu à un domaine qui prend des valeurs entières non négatives plutôt que des valeurs binaires pour le cas du Mono-Item, car le mono-item est une configuration multi-item où chaque catégorie ne peut contenir qu'un seul objet [20].

❖ Ressources ou tâches

À un niveau suffisamment élevé d'abstraction, un problème d'allocation de tâches peut être réduit à un problème d'allocation de ressources. Toutefois, il faut prendre en compte que les tâches sont comme des ressources ayant une utilité négative vis-à-vis de l'agent. De plus, les tâches se distinguent des ressources par le fait que ces dernières sont couplées à des contraintes définissant les combinaisons cohérentes qui peuvent leur être autorisées. Comme par exemple l'exécution d'une tâche comme précondition pour l'exécution d'une autre tâche. Dans cette thèse, cependant, nous nous concentrons sur les problèmes d'allocation de ressources en générale et non pas de tâche [20].

## I.5 Cloud computing et système multi-agents

Actuellement, le grand effort des recherches dans le cloud computing est consacré à la production des infrastructures, des technologies de virtualisation et de gestion des datacenters; un peu d'attention est consacré sur des méthodes novatrices qui permettront aux utilisateurs et aux développeurs de découvrir, demander, assembler et utiliser les ressources de cloud computing. Dans un système multi agent, les agents peuvent communiquer, négocier, et même coopérer …etc. En plus les agents ont beaucoup de caractéristiques (autonome, possède

un mécanisme de décision,.) qui les rends facile à utiliser et c'est pour cela ils sont utilisés dans plusieurs applications, parmi lesquels on cite le cloud computing [21].

### I.5.1 Cloud computing basée sur des agents

Grâce à ses caractéristiques (intelligence, réactivité, autonomie, mobilité et capacité à effectuer des prises de décision). Le SMA permet une gestion efficace des applications dans l'infrastructure cloud physique. À cette fin, nous devons façonner une nouvelle discipline, appelée cloud computing à base d'agents, qui fournit des solutions fondées sur la conception et le développement d'agents logiciels. De telles solutions pourraient améliorer les ressources cloud, la gestion et la découverte des services, la négociation des accords de niveau de service (SLA) et la composition des services.

Les agents autonomes pourraient rendre les nuages plus intelligents dans leurs interactions avec les utilisateurs et plus efficaces pour allouer la puissance de traitement et le stockage aux applications. Les agents peuvent rechercher, filtrer, interroger, et mettre à jour les volumes massifs d'informations hébergés dans les centres de données à grande échelle. Nous pourrions envisager un scénario dans lequel des agents cloud travaillant pour le compte des utilisateurs et de leurs systèmes d'exploitation fournissent des services intelligents d'accès aux données et de surveillance, mettent en œuvre des stratégies d'affectation processeur-application et aident le cloud infrastructures informatiques à utiliser de manière économe que en énergie [21].

## I.6 Conclusion

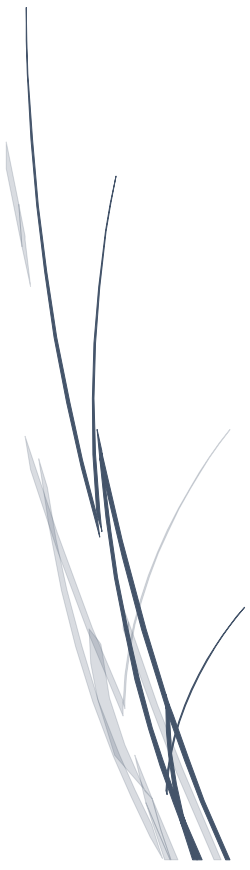
Dans ce chapitre, nous avons présenté les aspects et concepts de base dans le domaine du cloud computing et leur dépendance à l'égard des systèmes multi-agents pour créer une base théorique qui servira de point de départ pour mener nos recherches dans les chapitres suivants.

Le cloud computing basé agents est une discipline novatrice, son objectif est de fournir des solutions de cloud computing fondées sur la conception et le développement d'agents logiciels qui peuvent améliorer l'utilisation des ressources cloud. Les agents autonomes pourraient rendre les clouds plus intelligents dans leurs interactions avec les utilisateurs et plus efficace dans l'allocation des ressources.

Ainsi dans le prochain chapitre nous commencerons à explorer un autre point important dans notre sujet de la thèse en présentant les principaux travaux en relation avec l'allocation des ressources.

## Chapitre II.

# Allocation de Ressources dans Cloud Computing



## II.1 Introduction

L'allocation des ressources informatiques est un processus connu dans l'informatique classique (avant l'apparition de Cloud), qui consiste à l'affectation des ressources aux applications qui ont besoin de ces ressources.

Dans le cloud computing, l'allocation de ressources (RA) est le processus d'affectation des ressources disponibles aux services de cloud. L'allocation des ressources est une tâche délicate, la mauvaise exécution de cette tache peut rendre un service souffre de la famine et un autre profite de l'abondance des ressources.

Par conséquent, définir un mécanisme d'allocation des ressources aux utilisateurs d'une manière souple, dynamique et fiable est l'un des principaux enjeux dans le cloud computing.

## II.2 Allocation des ressources et ordonnancement : Concepts de base

Le contexte de notre travail se situe à l'intersection de la modélisation et de la résolution des problèmes d'allocation de ressources dans le cloud computing qui offrent des ressources informatiques à la demande, afin d'assurer la satisfaction totale des utilisateurs (fournir une solution aux besoins du client en raccourcissant le temps d'exécution et en réduisant les paiements des ressources demandées qui ont un caractère dynamique).

Afin que les utilisateurs puissent augmenter ou diminuer leur taux de consommation de ressources en fonction de leurs besoins. Cette section décrit un bref rappel de quelques généralités sur le problème d'allocation de ressources et d'ordonnancement de tâches.

### II.2.1 Allocation de ressources dans le cloud

Dans le cloud computing, l'allocation de ressources est le processus d'attribution des ressources disponibles pour les applications de cloud computing sur Internet. Les récentes définitions attribuées à l'allocation des ressources dans le cloud computing sont :

- L'allocation des ressources est le processus d'attribution des ressources disponibles pour satisfaire le besoin des applications du cloud. Les ressources cloud peuvent être provisionnées à la demande, d'une manière multiplexée. Dans le Cloud l'allocation des ressources correspond à la couche IaaS « infrastructure en tant que service » [22].
- « L'allocation des ressources est un élément important du cloud computing. Son efficacité influence directement sur la performance de l'ensemble de l'environnement du cloud. Il requiert le type et la quantité de ressources nécessaires pour chaque application afin de réaliser le travail de l'utilisateur » [23].
- « Dans le Cloud Computing, l'allocation des ressources (RA) est le processus d'attribution des ressources disponibles pour des applications de cloud via Internet. L'allocation des ressources des services si l'allocation n'ai été pas gérée avec précision »[24].

### II.2.2 Ordonnancement

Un problème d'ordonnancement consiste à ordonner dans le temps un ensemble de tâches contribuant à la réalisation d'un même projet. L'objectif est de minimiser la durée de réalisation du projet compte tenu des contraints d'antériorité reliant les différentes tâches [24].

### II.2.3 Tâche

Une tâche ou un job est une entité élémentaire localisée dans le temps, par une date de début et une date de fin, et dont la réalisation nécessite une durée préalablement définie. Elle est constituée d'un ensemble d'opération qui requièrent, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif [24].

### II.2.4 Ressource

La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche et disponible en quantité/capacité limitée.

### II.2.5 Ressource Cloud

Dans la littérature, il y a plusieurs définitions pour les ressources cloud. Nous avons identifié plusieurs définitions des ressources de cloud :

- Les ressources cloud peuvent être vues comme n'importe quelle ressource (physique ou virtuelle) que les utilisateurs peuvent demander au cloud. Par exemple, les utilisateurs peuvent demander des caractéristiques réseau, telles que la bande passante et le délai, et des exigences computationnelles, telles que le processeur, la mémoire et le stockage. En général, les ressources sont situées dans un datacenter qui est partagé par plusieurs clients, et doivent être attribués et ajustés dynamiquement en fonction de la demande [23].
- Les ressources de cloud consistent en des ressources physiques et virtuelles Les ressources physiques sont partagées entre plusieurs demandes grâce à la virtualisation et l'allocation automatique de ressources, La demande des ressources virtualisées est décrite par un ensemble de paramètres détaillant les besoins de traitement, de mémoire et de disque [25].

### II.2.6 L'équilibrage de charge (Load Balancing)

L'équilibrage de charge est une technique relativement nouvelle qui facilite l'exécution des tâches entre des ressources en fournissant un d´débit maximal avec un temps de réponse minimal [26]. Divisant le trafic entre les serveurs, les données peuvent être envoyées et reçues sans retard majeur. Différents types d'algorithmes sont disponibles qui aide le partage de charges entre les serveurs disponibles. Un exemple d'équilibrage de charge peut être lié à l'accès aux sites Web. Sans équilibrage de charge, les utilisateurs pourraient subir des retards,

délais d'attente et des éventuelles réponses du système longues. Des solutions d'équilibrage de charge s'appliquent habituellement sur des serveurs redondants qui permettent une meilleure répartition du trafic de communication de sorte que la disponibilité des sites web est définitivement tranchée [27].

## II.3 Importance de l'allocation des ressources dans le cloud

L'allocation des ressources est un élément important du cloud computing. Son efficacité influencera directement les performances de l'ensemble de l'environnement cloud. Il nécessite le type et la quantité de ressources nécessaires à chaque application pour terminer un travail utilisateur. Deux acteurs des environnements de cloud computing, les fournisseurs de cloud et les utilisateurs de cloud, poursuivent des objectifs différents ; les fournisseurs veulent maximiser leurs revenus en atteignant une utilisation élevée des ressources, tandis que les utilisateurs veulent minimiser les dépenses tout en répondant à leurs exigences de performance [28].

Afin de comprendre l'importance de l'allocation des ressources à l'ère du CC, l'allocation des ressources peut être vue comme une carte de progression entre la ressource et l'utilisateur (demande de l'utilisateur) qui considère les ressources disponibles et les demandes des utilisateurs comme des conditions de contrainte. L'objectif de l'allocation des ressources est de trouver la meilleure correspondance de ressources pour l'utilisateur afin d'atteindre une cible optimale telle qu'une utilisation élevée des ressources, une qualité de service pour l'utilisateur, etc [29]. Dans le cloud computing, l'allocation de ressources (RA) signifie l'affectation des ressources disponibles aux applications cloud en fonction des exigences fournies par l'utilisateur [30][24][31]. L'allocation des ressources ne sera pas efficace si l'allocation n'est pas gérée efficacement. Il est très important que l'allocation des ressources soit optimale afin de répondre aux besoins des utilisateurs [32].

En générale, les ressources computing deviendront la cinquième infrastructure publique majeure avec l'eau, l'électricité, le gaz et les télécommunications [33]. Avec la popularité croissante du cloud computing, en particulier le développement de la technologie de virtualisation, la planification des ressources informatiques ne peut plus se limiter à la restriction selon laquelle les tâches des utilisateurs doivent être attribuées à une seule machine physique. La planification peut être réalisée en répartissant les ressources à partir du point de maximisation des avantages afin que les ressources allouées à un utilisateur puissent être des ressources restantes fournies par plusieurs machines physiques. Bien que le traitement parallèle de plusieurs machines puisse générer une surcharge supplémentaire, ce type de gestion des ressources informatiques pourrait être plus efficace. La manière d'allouer des ressources aux utilisateurs de manière appropriée devient le facteur clé qui influence les performances du cloud [29].

## II.4 Directives d'allocation des ressources

Dans un environnement cloud, plusieurs stratégies, techniques et politiques d'allocation des ressources sont utilisés pour respecter les directives d'allocation efficace des ressources.

### II.4.1 Stratégies

Les stratégies d'allocation des ressources aident les deux principaux acteurs (utilisateurs et fournisseurs de services) du cloud computing à atteindre leurs objectifs. En raison de la nature orientée services du cloud computing, les utilisateurs sont préoccupés par la qualité et la fiabilité, par conséquent les utilisateurs peuvent souhaiter estimer les demandes en ressources pour terminer un travail avant l'heure estimée. Cela pourrait cependant conduire à la situation qualifiée de sur approvisionnement. D'autre part, les fournisseurs souhaitent maximiser leurs profits en utilisant moins de ressources par utilisateur afin d'accueillir plus d'utilisateurs et de faire plus de profits. Cela entraînera un sous-provisionnement. Cependant, il est difficile d'allouer les ressources de manière mutuellement optimale en raison du manque de partage d'informations entre eux. De plus, l'hétérogénéité, la variabilité de l'environnement et l'incertitude toujours croissantes des ressources dans le nœud qui ne peuvent pas être satisfaites avec l'allocation traditionnelle des ressources posent des défis plus difficiles pour les deux parties [23] éviter les problèmes de sous / sur approvisionnement des ressources. Des utilisateurs, les exigences de l'application et le SLA sont requis tandis que des fournisseurs, les offres, les ressources disponibles, l'état des ressources et le SLA sont requis [25] Les exigences des deux parties sont rassemblées afin d'allouer de manière optimale les ressources afin de satisfaire divers utilisateurs exigences, une stratégie d'allocation des ressources doit contourner les scénarios suivants, comme l'opinion de [23] [25] [24] [34].

a) La contention de ressource : Une situation de conflit de ressources survient lorsque deux les applications essaient d'accéder à la même ressource en même temps.
b) La rareté des ressources : apparaît lorsque les ressources sont limitées Ressources.
c) La fragmentation des ressources : survient lorsque le les ressources sont isolées. (Il y aura assez de ressources mais pas en mesure d'allouer à l'application nécessaire.)
d) Le sur approvisionnement en ressources : survient lorsque le l'application obtient des ressources excédentaires que celles demandées.
e) Le sous-provisionnement des ressources : se produit lorsque l'application est affectée avec moins de nombres de ressources que la demande.

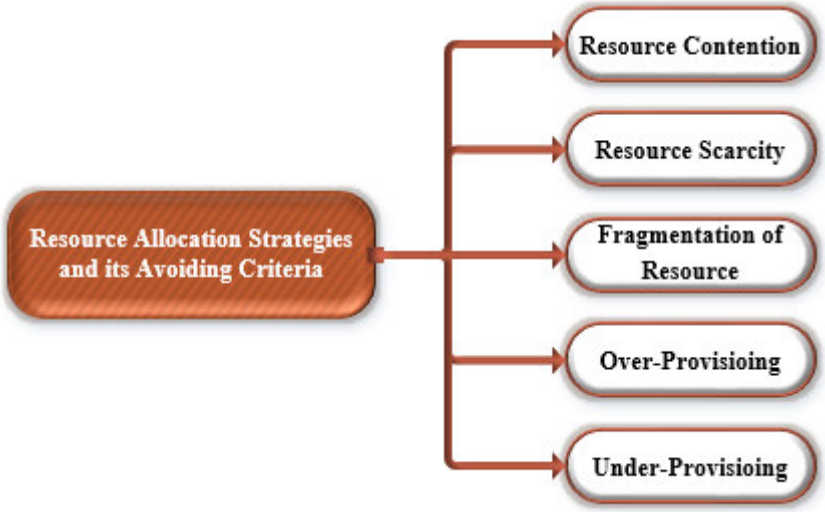



**Figure II. 1 Stratégies d'allocation des ressources et ses critères d'évitement [35].**

Plusieurs services sont souvent hébergés par le même serveur dans l'architecture orientée services et les services du même serveur sont en concurrence pour les ressources disponibles limitées du serveur, notamment le temps CPU et la mémoire et les ressources réseau telles que la bande passante. Différentes allocations de ressources entraîneront une qualité de service différente lors de l'exécution. Il est important de noter que l'utilisateur peut considérer ces ressources limitées comme illimitées et l'outil qui rend cela possible est la stratégie d'allocation des ressources [35]. Certaines des différentes stratégies d'allocation des ressources sont examinées dans les paragraphes suivants :

### II.4.1.1 Allocation de ressources basée sur la qualité de service (QOS)

L'allocation des ressources basée sur la qualité de service se concentre sur la satisfaction du client, l'utilisation rentable et efficace de la ressource. L'allocation des ressources dans le cloud computing diffère de l'environnement informatique distribué traditionnel en raison de la présence de diverses métriques de QoS telles que la mémoire du processeur, la vitesse et la stabilité. La technique d'allocation des ressources considère principalement les paramètres de qualité de service sur les fournisseurs de ressources, y compris le prix et la charge [36]. La stratégie d'allocation des ressources concentre les métriques de QoS basées sur les SLA pour améliorer le profit des fournisseurs SaaS [37]. Diverses techniques d'allocation des ressources ont été proposées par différents auteurs pour améliorer les objectifs de qualité de service (QOS). K.S.Guruprakash, S.Siva Sathya [38] ont proposé une méthode d'allocation de ressources dans le cloud en utilisant le fichier journal du serveur Web. L'objectif QOS est identifié à partir du fichier journal. Cette méthode utilise le fichier journal du serveur Web pour rechercher l'exigence. Les objectifs de qualité de service considérés ici sont la bande passante, la fiabilité, la disponibilité, la capacité de calcul, la facilité d'utilisation, l'exactitude, la fiabilité, etc. Ici, les auteurs se sont concentrés sur l'identification automatique des besoins

des utilisateurs basés sur le fichier journal afin d'allouer des ressources aux consommateurs, ce qui offre une meilleure qualité de service. L'approche de [39] présente une méthode d'allocation automatique des ressources. Les auteurs ont proposé un cadre de modélisation pour l'optimisation des ressources en tenant compte des objectifs de qualité de service. Le cadre est composé des phases suivantes : modélisation, analyse de modèle, automatisation des tests, génération de code et techniques d'optimisation. ROAR utilise un langage de spécification appelé GROWL, un langage d'optimisation d'applications Web. GROWL utilise l'espace de configuration et les informations d'objectif QOS. La spécification est traduite en plan de test qui est l'entrée du module contrôleur de ROAR.

#### II.4.1.2 Allocation des ressources basée sur les coûts

Certaines méthodes d'allocation des ressources basées sur les coûts sont proposées dans la littérature. L'auteur dans [40]a proposé une méthode pour optimiser le coût d'allocation des ressources dans le cloud computing. Cloud fournit un plan de réservation et des plans à la demande à l'utilisateur. Le prix du plan de réservation est bas par rapport au plan à la demande, car les utilisateurs doivent payer à l'avance. Avec le plan de réservation, le coût total d'allocation des ressources peut être réduit. Les demandes futures des utilisateurs ne sont pas prises en compte dans cet article et l'incertitude est associée aux prix des ressources des fournisseurs. Pour résoudre ce problème, un algorithme est proposé et il est applicable au plan à long terme. Les auteurs de [41] ont introduit une méthode d'allocation des ressources sur le marché du cloud coopératif. L'objectif est de réduire le coût tout en allouant la ressource à l'utilisateur. Cet article propose un algorithme pour améliorer le coût. L'idée de base du cloud orienté marché a été donnée par Buyya et al. [42].

#### II.4.1.3 Allocation des ressources basée sur le Temps d'exécution

Pour surmonter le défi de la contention des ressources tout en allouant des ressources aux tâches en utilisant le principe du traitement parallèle dans le cloud, Li, Jiayin et al [43] ont proposé l'utilisation du temps d'exécution réel pour planifier de manière préventive comment ces ressources sont provisionnées pour l'efficacité. Cela se fait en ajustant l'allocation des ressources de manière adaptative en fonction de la mise à jour des exécutions de tâches réelles. Cependant, il est difficile d'estimer le temps d'exécution d'un travail. Cela est généralement dû au fait que les allocations de temps d'exécution pour les travaux sont surestimées. Un travail avec un temps d'exécution surestimé peut prendre moins de temps que prévu ou peut être interrompu avant l'expiration du temps préempté, entraînant ainsi un gaspillage brut de ressources. Cela peut éventuellement conduire à une dégradation des performances du système, car les travaux qui auraient pu être occupés par des ressources inactives sont finalement refusés.

### II.4.1.4 Allocation des ressources basée sur la Machine Virtual (VM)

La virtualisation signifie créer une image virtuelle d'un élément physique tel qu'un périphérique de stockage, un système d'exploitation ou tout élément de traitement. Le cloud divise la ressource en un ou plusieurs environnements d'exécution [44]. La structure constituée d'un système virtuel est prête à être développée en tant qu'élément physique. La migration de VM offre des avantages incroyables, par exemple, l'ajustement de la charge, la solidification du serveur, la maintenance en ligne et l'adaptation proactive aux pannes non critiques. Un modèle qui propose une technologie de virtualisation pour allouer dynamiquement les ressources disponibles avec l'optimisation du nombre de serveurs utilisés et suivre les demandes des applications et prendre en charge l'informatique verte [45]. Dans l'article [18], les auteurs présentent un système utilisant la technologie de virtualisation pour allouer des ressources de Datacenter de manière dynamique en fonction des demandes des applications informatique en optimisant le nombre de serveurs utilisés. Ils introduisent le concept de « asymétrie » pour mesurer les inégalités dans l'utilisation des ressources multidimensionnelles d'un serveur. En minimisant l'asymétrie, ils peuvent combiner différents types de charges de travail et améliorer l'utilisation globale des ressources du serveur. Ils ont aussi développé un ensemble d'heuristiques qui empêchent efficacement la surcharge dans le système tout en économiser l'énergie utilisée.

Les auteurs visent deux objectifs principaux :

- Éviter les surcharges : la capacité d'une machine devrait être suffisant pour satisfaire les besoins en ressources de tous les ordinateurs virtuels en cours d'exécution. Sinon, la machine est surchargée et peut conduire à dégradation des performances de ses machines virtuelles.
- Informatique verte : le nombre de machines utilisées devrait être minimisé tant qu'ils peuvent encore satisfaire les besoins des toutes les VM. Les machines inactives peuvent être désactivées pour économiser de l'énergie.

Des études récentes sur l'allocation de machines virtuelles dans le cloud pour des tâches en temps réel se concentrent sur différents aspects, tels que les infrastructures, afin de permettre des tâches en temps réel sur les machines virtuelles et la sélection de machines virtuelles pour la gestion de l'alimentation dans le Datacenter. Mais le travail de [19], ont alloué les ressources en fonction de la vitesse et du coût des différentes machines virtuelles dans IaaS. Il diffère des autres travaux connexes en permettant à l'utilisateur de sélectionner des ordinateurs virtuels et en réduisant les coûts pour l'utilisateur.

### II.4.1.5 Allocation des ressources basée sur la Planification linéaire

L'affectation des ressources aux tâches sur une base individuelle est généralement associée à un temps d'attente/réponse élevé. Ce défi a conduit à la formulation d'une stratégie d'allocation des ressources d'ordonnancement linéaire [46] qui se concentre sur la répartition

des ressources entre les emplois capables de maximiser la fonction de coût à mesure qu'ils arrivent. Cet avantage de cette stratégie est qu'elle améliore le débit puisque les travaux qui peuvent utiliser efficacement les ressources disponibles sont pris en charge. Cependant, cette stratégie n'est pas adaptée aux systèmes en temps réel car les tâches ne sont pas traitées selon le principe du « premier arrivé, premier servi » ou « premier entré, premier sorti ». Abirami et Ramanathan [46] proposent un algorithme d'ordonnancement appelé Linear Scheduling for Tasks and Resources (LSTR), qui applique l'ordonnancement pour le traitement des tâches et des ressources respectivement. Il combine le service Nimbus [47] à un nœud de serveur pour établir l'environnement cloud IaaS et la méthode de virtualisation est KVM / Xen avec la planification LSTR pour allouer des ressources. L'allocation dynamique pourrait être effectuée par l'ordonnanceur de manière dynamique sur des demandes de ressources supplémentaires avec une évaluation continue de la valeur de seuil. Les demandes de ressources sont collectées et triées dans différentes files d'attente en fonction d'une valeur de seuil. Ensuite, les demandes sont satisfaites par les VM.

#### II.4.1.6 Allocation des ressources basée sur l'efficacité énergétique

Dans l'allocation des ressources, l'efficacité énergétique a un rôle essentiel. La reprogrammation de l'allocation améliore les économies d'énergie du datacenter cloud IaaS interne. L'avantage de l'algorithme de réallocation dans les composants informatiques de nouvelle génération est qu'il offre des performances plus élevées et consomme moins d'énergie. L'allocation dépend de la charge de travail de l'application et du nombre de réorganiser l'allocation des ressources de l'ancien et du nouveau serveur [48]. La distribution des ressources avec QoS améliore la compétence de l'énergie, l'allocation des ressources du datacenter de condition heuristique aux demandes des utilisateurs [49]. Le processeur, la mémoire, le stockage sur disque et les interfaces réseau déterminent l'utilisation de l'énergie du datacenter. Les VM peuvent être redimensionnées et fusionnées en cas de sous-utilisation des ressources. Le datacenter passe en mode veille des nœuds inactifs pour rendre l'allocation des ressources économe en énergie. L'algorithme de sélection clonale amélioré (ICSA) réduit le MakeSpan et la consommation d'énergie dans un environnement de cloud computing. Il améliore l'efficacité énergétique dans le datacenter cloud et respecte les accords SLA [50] . Le travail dans [49] satisfait l'allocation de ressources sensible à l'énergie avec la condition thermique et économe en énergie. La consommation d'énergie influe sur le coût opérationnel de l'allocation. Les divers algorithmes d'équilibrage de charge, de tourniquet et d'algorithmes gourmands ne sont pas efficaces pour une allocation efficace des ressources. La consommation d'énergie dépend de la charge de travail des demandes et de l'état actif des systèmes.

### II.4.2 Techniques d'optimisation

Le problème d'allocation des ressources est un problème d'optimisation combinatoire, où une quantité limitée de ressources doivent être alloué à un certain nombre d'événements de telle sorte que l'allocation la plus efficace des ressources est atteinte grâce à l'optimisation. Les objectifs de problème d'allocation de ressources sont généralement de réduire le coût global, le temps de traitement, gérer les conflits, maximiser le bénéfice, l'efficacité ou la compatibilité. Le problème d'allocation des ressources implique généralement un nombre énorme de variables entières et multiples objectifs dans un espace de recherche discret, ce qui provoque leur complexité à être NP-complet dans le pire des cas. Les méthodes et les techniques utilisées pour résoudre le problème de gestion et d'allocation des ressources dans les CC peuvent être classées en trois catégories : techniques traditionnelles, techniques heuristiques et techniques métaheuristiques. [51].

#### II.4.2.1 Techniques traditionnelles

Techniques essentielles pour planifier différentes tâches telles que : Shortest Job First (SJF) / plus court d'abord, L'algorithme FIFO (First In First Out) ou FCFS (First Come First Served) et le Round Robin (RR) [52]. Ces techniques sont simples, rapides, déterministes et permettent d'obtenir des solutions exactes [53]. Mais, ils ne sont pas efficaces pour comprendre le problème d'optimalité dans de nombreuses situations [54]. Ainsi, les techniques traditionnelles ne sont pas réalisables dans la planification de l'environnement cloud [55]. De nombreux travaux ont été menés pour améliorer la mise en œuvre des techniques traditionnelles [52] [56] [57]. Le round robin est l'une de ces techniques qui fonctionnent à l'aide d'une tranche de temps ou d'un quantum. L'algorithme RR a l'inconvénient d'utiliser un quantum de temps statique [58]. L'ordonnancement CPU proposé dans [56] repose sur l'ordonnancement à tour de rôle, mais change la manière des calculs d'ordonnancement. Il diminue le temps d'attente et le temps d'exécution, contrairement à l'ordonnancement RR simple, plutôt que de donner un quantum de temps statique dans l'ordonnancement CPU. L'algorithme FCFS signifie que la tâche qui vient en premier sera exécutée en premier. Les chercheurs dans [59] ont proposé un algorithme de planification des tâches basé sur des algorithmes de clustering flou pour augmenter l'utilisation des ressources et minimiser le temps d'exécution des tâches. SJF est une technique d'ordonnancement qui dépend du temps d'exécution de la tâche. Les tâches sont mises en file d'attente en fonction de la priorité, le temps le plus long est placé en dernier avec la priorité la plus faible et le temps le plus petit est placé en premier avec la priorité la plus élevée [60]. Dans cet algorithme, la CPU est affectée à la tâche avec le moins de temps de rafale. Elmougy et al. Dans [57] a proposé un algorithme hydrure de RR et SJF appelé algorithme SRDQ. Cet algorithme considère un temps quantique de tâche variable dynamique.

### II.4.2.2 Techniques Heuristiques

Ces techniques utilisent l'espace d'échantillonnage de solutions aléatoires pour trouver la solution optimale ou presque optimale [55]. Il existe de nombreuses techniques heuristiques telles que min-min, min-min basé sur la priorité, max-min amélioré et maxmin [61]. Ces techniques donnent de meilleurs résultats par rapport aux techniques traditionnelles, mais ne garantissent pas un classement élevé dans l'ordonnancement du cloud [62]. Les solutions issues des techniques heuristiques se bloquent souvent dans le problème des minimas locaux [55]. Une technique Max-min améliorée utilisant le temps d'exécution attendu pour la base de sélection au lieu du temps d'achèvement est proposée dans [63]. Il alloue une tâche avec un temps d'exécution moyen. L'algorithme augmente les chances d'affectation synchrone des tâches sur les ressources. L'algorithme de base Min-Min est un algorithme simple et efficace qui génère la meilleure planification en termes de réduction du temps d'achèvement des tâches. Cependant, le plus gros inconvénient est l'équilibrage de charge, qui est considéré comme l'un des défis majeurs pour les fournisseurs de services cloud. Les auteurs de [64] ont amélioré l'équilibrage de charge en proposant l'algorithme Load Balance Improved Min-Min (LBIMM). L'algorithme LBIMM est conçu sur la base de l'algorithme Min-Min afin d'augmenter l'utilisation des ressources et de réduire le temps d'achèvement.

### II.4.2.3 Techniques Méta-Heuristique

Le problème de l'allocation des tâches sur les ressources dans un environnement de cloud computing est un problème NP-Hard. Par conséquent, la planification des tâches est clarifiée en utilisant une méta-heuristique et une heuristique pour obtenir des solutions quasi optimales ou optimales. Les techniques heuristiques sont des sous-ensembles des techniques méta-heuristiques [65]. Les techniques méta-heuristiques sont souvent inspirées par la nature du comportement social des insectes [66]. Le mot métaheuristique est énoncé par Fred Glover en 1986, le préfixe « méta » signifie niveau supérieur et « heuristique » signifie découvrir par essais et erreurs. Nous avons adopté une définition claire du mot métaheuristique de [67]"une métaheuristique est un cadre algorithmique indépendant des problèmes de haut niveau qui fournit un ensemble de directives ou de stratégies pour développer des algorithmes d'optimisation heuristique. Le terme est également utilisé pour désigner une implémentation spécifique à un problème d'un algorithme d'optimisation heuristique selon les directives exprimées dans un tel cadre".

Toutes les techniques méta-heuristiques ont deux composantes principales, l'intensification et la diversification. La diversification génère des solutions assorties pour explorer plus en profondeur l'espace de recherche à l'échelle mondiale, tandis que l'intensification se concentre sur la recherche à l'échelle locale en utilisant les informations locales dans le processus de recherche pour générer de meilleures solutions. Ainsi, l'information locale courante peut être dérivée de la cible. Parce que les techniques heuristiques se retrouvent souvent coincées dans

le problème des optima locaux, les techniques méta-heuristiques se sont révélées les plus efficaces pour éviter cette situation, comme mentionné dans [55] [68] [69].

Les techniques méta-heuristiques sont classées en deux catégories : l'intelligence en essaim (SI) et la bio-inspirée. Bioinspired a pénétré dans presque tous les domaines des sciences, de l'exploration de données, du génie biomédical, des systèmes de contrôle et du traitement parallèle. Il existe de nombreux algorithmes bio-inspirés tels que: les algorithmes génétiques (GA), recuit simulé (SA) et l'algorithme compétitif impératif (ICA). L'intelligence en essaim est une technique relativement nouvelle pour résoudre les problèmes d'optimisation sans contrainte et s'inspire d'un comportement social des colonies d'insectes et d'autres animaux tels que : optimisation par essaim de particules (PSO), optimisation par colonie de fourmis (ACO), colonie d'abeilles artificielles (ABC), algorithme d'essaim de vers luisants (GSA), BA, algorithme Firefly (FA), recherche de coucous (CS), optimisation d'essaim de chats (CSO). Les chercheurs essaient toujours de trouver de meilleurs algorithmes, en particulier pour la planification des tâches dans le cloud computing.

Les chercheurs de [70] ont présenté un algorithme d'ordonnancement indépendant des tâches dans le calcul en grille par une fusion de PSO avec la recherche locale d'émulation gravitationnelle (GELS) pour éviter le problème des minimas locaux. L'algorithme de fusion PSO-GELS montre une réduction significative du temps Makespan. Une nouvelle approche a été présentée dans [71], elle utilise un algorithme génétique familial (FGA) pour augmenter l'utilisation des ressources en affectant efficacement les machines virtuelles aux machines physiques appropriées. CSO-GA [72] est une combinaison d'algorithmes CSO et GA. Cet algorithme hybride optimise le Makespan par rapport à d'autres techniques d'ordonnancement. Les chercheurs de [73] ont proposé un nouvel algorithme basé sur les colonies de fourmis pour réduire le temps de réponse en équilibrant la charge via la recherche sous le nœud chargé. Cet algorithme utilise FCFS pour allouer les tâches aux machines virtuelles.

### II.4.3 Politiques

La plupart des méthodes d'allocation des ressources actuelles [74] (soit fondées sur des algorithmes d'allocation de ressources statique ou dynamique) supposent que tous les besoins en ressources sont déterminés à l'avance [75] [76]. Mais en fait, les besoins en ressources des systèmes de cloud computing ne peuvent pas être prédéterminés car les ressources sont distribuées les unes après les autres au fil du temps. Les méthodologies d'allocation des ressources précédentes peuvent compléter la distribution des ressources avec précision dans les serveurs à petite échelle. Mais face à des milliers de serveurs dans des systèmes de cloud computing, ces algorithmes d'allocation de ressources deviendront très compliqués [77] [78]. Les algorithmes d'allocation de ressources précédents supposent souvent que les ressources sont homogènes et que chaque ressource serait traitée de la même manière. Mais en réalité, il

existait un grand nombre de ressources hétérogènes dans les systèmes de cloud computing, qui ne peuvent pas être traitées de la même manière. Cela devient un point chaud que nous définissons une politique d'allocation des ressources pour comparer la capacité des ressources hétérogènes [79]. Il existe actuellement trois politiques d'allocation des ressources dans les systèmes de cloud computing, qui constituent la base de la politique [74].

***Politique 1*:** Lorsque les besoins en ressource arrivent, le système de cloud computing analyse l'état d'utilisation des ressources en fonction des SLA dans tous les serveurs, pour trouver les ressources restantes satisfaisant SLA et les affecter aux utilisateurs.

Les inconvénients de cette politique sont :

- ☒ S'il Ya beaucoup de types de serveurs différents dans le datacenter, le balayage de chaque serveur serait demandé beaucoup de temps et conduirait à la plus mauvaise réponse et la performance de l'allocation des ressources serait plus faible.
- ☒ Lorsque le nombre d'utilisateurs des ressources augmente dans une période de temps, l'allocation des ressources pour chaque exigence en analysant chaque exigence séparément augmenterait considérablement le temps d'attente et conduire à une efficacité inférieure du système.

***Politique 2:*** L'information dynamique sur les ressources physiques est enregistrée dans un centre de gestion des ressources. Les systèmes de cloud computing collectent les changements des ressources dans chaque serveur en temps régulier par un programme spécial et mettre à jour leurs données d'utilisation. Lorsque les besoins en ressources arrivent, les données d'utilisation des ressources sont recherchées et les ressources sont allouées si les ressources répondant au SLA ont été découvert. Cette politique peut améliorer grandement le temps de réponse et réduire le cycle d'attente moyen des utilisateurs.

Les inconvénients de cette politique sont :

- ☒ Le manque de planification coordonnée pour l'allocation globale des ressources conduirait à un taux de rejet plus élevées des besoins en ressources.

***Politique 3:*** elle est basée sur la politique 2, quand un grand nombre d'utilisateurs des ressources arrivent dans un certain temps, les exigences sont triées par SLA. Les systèmes de cloud computing satisfont d'abord les besoins qui méritent une grande granularité des ressources et en suite satisfont les exigences qui méritent des petites granularités des ressources.

- ☒ Les inconvénients de cette politique sont les suivantes : Le changement des exigences de ressources dans les SLA est presque inconsidéré et la QoS est encore très faible. Surtout, le cycle moyen d'attente des ressources de l'utilisateur est significativement augmenté en raison du processus de file d'attente.
- ☒ Le manque de précision dans l'allocation des ressources conduirait à une fragmentation excessive des ressources.

Globalement, le cycle d'attente moyen des ressources dans la politique 1 est plus long que d'autres politiques.

## II.5 Utilisation du SMA dans cloud computing

La technologie SMA, dérivée de l'intelligence artificielle distribuée, a montré son efficacité pour résoudre les problèmes des systèmes distribués tels que l'allocation de ressources dans les systèmes de cloud computing, le routage en robotique, l'équilibrage de charge basé sur des agents mobiles dans les grilles, l'allocation de tâches basée sur la négociation dans les réseaux de capteurs et les réseaux sociaux, et la modélisation de réseaux sociaux basée sur les agents [80]. Dans les systèmes multi-agent, la ressource signifie tout objet qui peut être acquis par l'agent via le processus de négociation. Chaque sous-ensemble possible de l'ensemble des ressources disponible aura un effet particulier sur l'utilité de l'agent selon les préférences de ce dernier. À cet égard, l'intégration entre les systèmes de cloud computing et les multi-agents est une utilisation efficace de l'infrastructure informatique, de la prestation de services, du stockage de données, des techniques de virtualisation évolutives et de l'efficacité énergétique. En résumé, on peut dire qu'en cloud computing, l'objectif principal de la recherche est l'utilisation efficace de l'infrastructure à des coûts réduits. Dans l'allocation de ressources est utilisé le système Multi-agent pour la distribution d'un ensemble de ressources parmi un ensemble d'agents intelligents (ces agents travaillant dans un environnement de réseau intelligent sens, communiquent, collaborent et agissent les uns avec les autres) pour une gestion efficace des ressources dans la répartition et l'hétérogénéité de cette infrastructure.

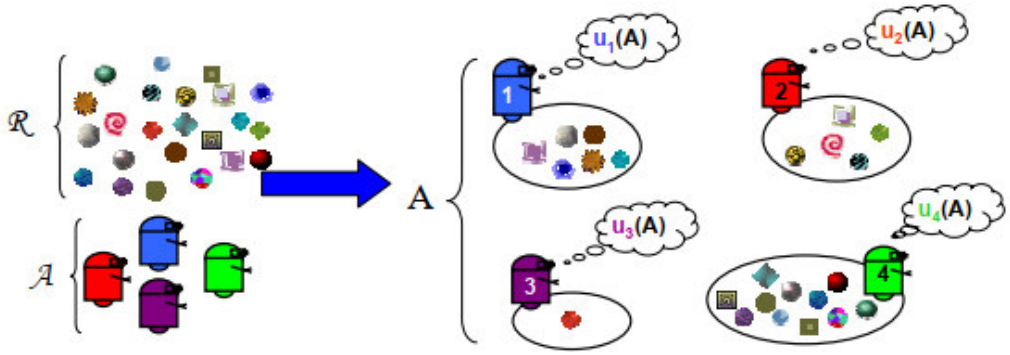



**Figure II. 2 Les ressources sont représentées par le système multi-agents** [81].

À cet égard, il existe deux grandes catégories pour adresser le problème d'allocation de ressources basé sur des systèmes multi-agents : soit de façon centralisée ou de façon distribuée. Dans les procédures centralisées, les décisions finales relatives à l'allocation de ressources sont entreprises par une entité centrale qui doit être impartial afin que les agents acceptent de collaborer avec cette dernière. Tandis que, une procédure décentralisée permet à l'ensemble des agents de participer à la prise de décision concernant l'allocation de ressources. Dans cette procédure, les décisions d'allocation de ressources se font à travers l'interaction des agents entre eux et non pas à travers une entité centrale qui raisonne à travers

les informations des différents agents. Tout cela doit être fait en respectant les contraintes du problème qu'on doit adresser et parmi ces contraintes il y a les règles du protocole de négociation à respecter.

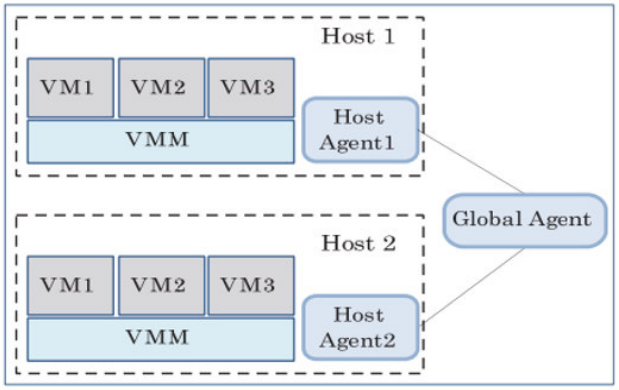



**Figure II. 3 Centralisation allocation architecture** [82].

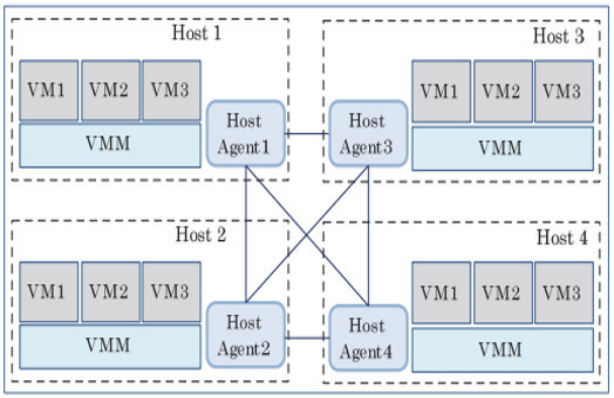



**Figure II. 4 Distributed allocation architecture** [82].

Dans cette thèse, nous présentons une nouvelle approche d'allocation de ressources VM distribuée qui utilise une fonction utilitaire basée sur des systèmes multi-agents. Les agents, attachés à chaque hôte physique, sont chargés de prendre des décisions pour l'hébergement en direct des machines virtuelles dans un hôte. De plus, nos contributions sont des procédures décentralisées pour les avantages suivant qu'elles procurent :

- La complexité de la négociation est répartie sur l'ensemble des agents du système.
- Evite la problématique de divulguer des informations privées à une entité centrale.
- Permet de laisser une certaine autonomie aux agents.
- Semble plus appropriée à certains types de problèmes, spécifiquement lors du passage à l'échelle.

## II.6 Utilisation du Logique floue dans cloud computing

La logique floue (fuzzy logic, en anglais) est une technique utilisée en intelligence artificielle. Elle a été formalisée par Lotfi Zadeh en 1965 et utilisée dans des domaines aussi variés que l'automatisme (freins ABS), la robotique (reconnaissance de formes), la gestion de la circulation routière (feux rouges), le contrôle aérien, l'environnement (météorologie, climatologie, sismologie, analyse du cycle de vie), la médecine (aide au diagnostic), l'assurance (sélection et prévention des risques) et bien d'autres [83].

Un avantage de la logique floue pour formaliser le raisonnement humain est que les règles sont définies en langage naturel. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi une condition d'être dans un état autre que vrai ou faux, la logique floue fournit une valeur très flexible de raisonnement qui permet de prendre en compte les inexactitudes et incertitudes. La logique floue [83] [84] est un outil qui traite de l'incertain, des informations imprécises ou qualitatives, ainsi qu'avec des informations précises dans des systèmes qui ne sont pas définis avec un modèle mathématique formel. En logique booléenne,

un élément x peut ou non appartenir à un ensemble A, avec un degré d'appartenance égal à 1 ou 0. Au lieu de cela, en logique floue, le degré d'appartenance de x à un ensemble flow F une valeur dans un intervalle continu entre 0 et 1.

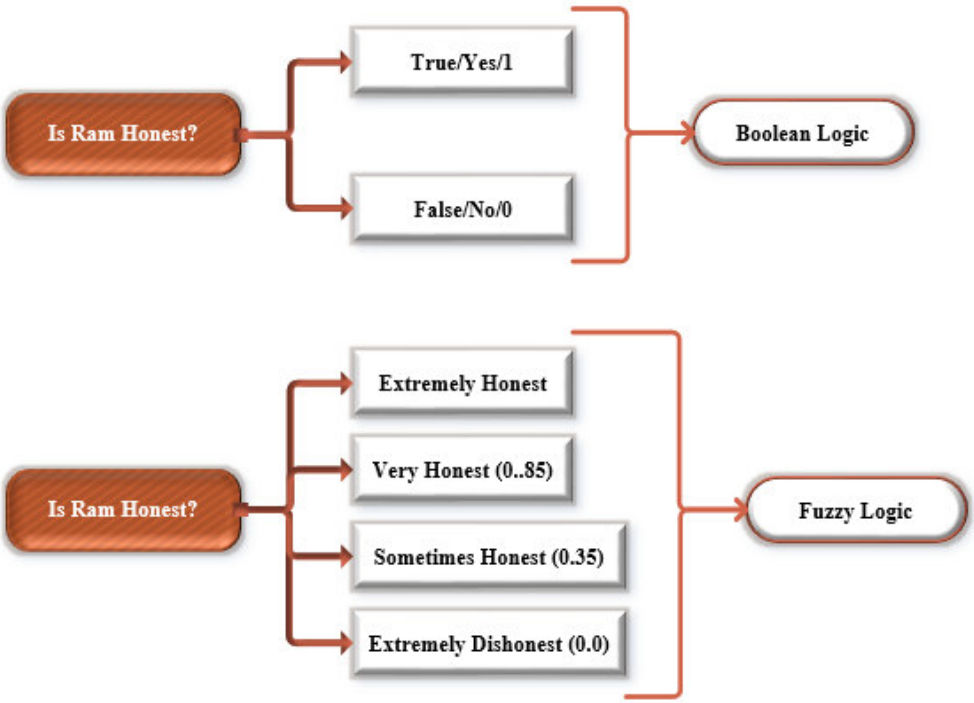



**Figure II. 5 Logique floue vs logique booléenne** [85]

En d'autres termes, FL ne fonctionne pas avec des valeurs strictes comme 0 et 1, il fonctionne avec des valeurs comprises entre [0 ; 1]. La capacité de travailler avec des valeurs linguistiques telles que Faible, Normal, Élevé, Très élevé pour l'obtention de résultats plus précis et certains avantages économiques lorsqu'ils sont appliqués à un système technique.

À cet égard, La gestion des systèmes cloud peut tirer parti de la logique floue de différentes manières, allant de la représentation des connaissances et des faits observés d'une manière similaire à celle des humains (c'est-à-dire avec le langage naturel et ses imprécisions), au raisonnement approximatif comprenant des informations incomplètes ou inexactes (comme les humains sont capables de faire). Nous identifions et analysons les différentes utilisations de la logique floue dans la gestion des systèmes cloud, en soulignant son adoption dans divers composants et les avantages qu'elle offre [86]. La logique floue peut être utilisée pour permettre aux utilisateurs de définir facilement la pertinence relative ou leurs préférences parmi les exigences. Par exemple, dans une application Big Data, un utilisateur peut spécifier que les besoins de stockage ont une « importance élevée », les performances de l'application ont une « importance moyenne » et l'interface utilisateur et l'interaction ont une « importance faible ». La fusion des exigences en tenant compte de leurs préférences relatives peut guider

le moteur d'allocation des ressources dans la recherche d'une solution réalisable, en résolvant un problème d'optimisation multi-objectifs.

La théorie de la logique floue a joué un rôle important dans le cloud computing. Frey et al. [87] ont décrit une méthode utilisant un contrôleur flou pour contrôler l'échelle du nuage. Grâce à certaines informations supplémentaires, la charge maximale est prédite, de sorte que la mise à l'échelle du cluster virtuel peut être effectuée à l'avance. Dans [88] les auteurs gèrent la réallocation dynamique des ressources dans le cloud computing DC en utilisant une version multi-agent du contrôleur Fuzzy. Chaque agent est responsable de l'allocation des ressources d'une seule VM en parallèle avec d'autres agents. Cette approche considère le CPU et la mémoire comme des ressources à gérer. Ramezani et al. [89] ont décrit une méthode de prédiction basée sur la logique floue. Les types de charge de travail des machines virtuelles et le temps de migration des machines virtuelles ont été prédits sur la base de la méthode de prédiction floue, et enfin l'allocation automatique des ressources pour l'application a été réalisée.

Les utilisateurs ont souvent tendance à surestimer leurs exigences en termes de SLA, afin d'éviter d'éventuels problèmes lors du fonctionnement de leurs applications. Cela conduit cependant à un gaspillage de ressources payantes et réduit la disponibilité des ressources dans le cloud. Pour limiter ce gaspillage de ressources (et d'argent pour les utilisateurs finaux). Nous proposons dans nos travaux actuels l'utilisation de la logique floue pour faciliter la représentation linguistique des valeurs de ressources dynamiques (faibles, moyennes, élevées...) et aider le système à déterminer la meilleure solution en fonction des critères du client demander. La logique floue permet aux clients de définir les besoins de leurs applications de manière flexible et de capturer des expressions en langage naturel, lorsque les utilisateurs ne sont pas spécialisés dans les systèmes et technologies d'information et lorsque les besoins ne sont pas définis avec précision ou facilement identifiables.

## II.7 Revue de littérature sur l'allocation de ressources dans le cloud

Dans cette section, nous allons présenter les travaux qui sont relatives à l'allocation de ressources dans le cloud. Pandey et Singh [90] ont introduit une planification des tâches dans le cloud basé sur FL pour obtenir l'équité de l'allocation des ressources en fonction de la qualité de service. L'approche proposée traite deux types de défis : Le premier défi est de savoir comment sélectionner les meilleures VM pour soumettre la tâche, tandis que le deuxième défi réside dans la justification de la tâche en fonction de la qualité de service. Les techniques utilisées montrent une meilleure association mathématique entre les paramètres pour ajuster le vecteur d'attente de la tâche classée. En outre, il y a plus de paramètres de qualité de service à étudier tels que le coût. Les auteurs de la référence [91] ont présenté une technique pour améliorer l'allocation des ressources en utilisant les technologies du Web sémantique basées sur le système multi-agents pour faciliter l'interopérabilité entre les

utilisateurs et les différents fournisseurs de ressources. L'avantage d'utiliser à la fois les technologies du Web sémantique et le système multi-agent est l'exécution des demandes des clients selon les règles des fournisseurs. Mais cette méthode présente un certain nombre de défis, notamment la dynamique et l'évolutivité des ressources virtuelles dans le cloud. Les auteurs de [92] ont proposé une technique d'allocation de ressources basée sur le processus de hiérarchie analytique (AHP) communément appelée synthèse de préférences multicritères (MPS). Cette méthode dépend des tâches des utilisateurs, qui sont basées sur des critères d'erreur afin d'assurer la préservation et la cohérence en fournissant les priorités aux tâches des utilisateurs. Lu et al. [93] présente un modèle d'allocation des ressources basé sur un cadre d'évaluation de la justice en utilisant deux sous-modèles (Dynamic Demand Model (DDM) et Dynamic Node Model (DNM)) pour décrire la demande de ressources. Les auteurs utilisent plusieurs algorithmes typiques dans l'allocation des ressources, tels que les algorithmes basés sur les utilitaires, pour prouver leur efficacité. Comme point fort, ce modèle supporte les demandes de ressources dynamiques, mais il ne prend pas en compte le temps de réponse.

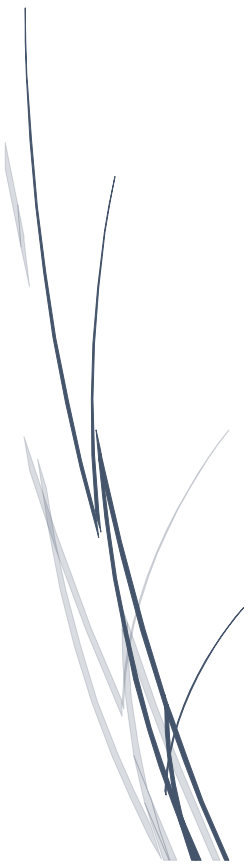
## II.8 Conclusion

Le cloud computing est un paradigme émergent dans le monde de l'informatique, où l'allocation des ressources est l'un des éléments les plus importants. Une stratégie efficace d'allocation des ressources est essentielle pour atteindre la satisfaction des utilisateurs et maximiser les profits des fournisseurs de services de cloud computing.

Dans ce chapitre, les aspects fondamentaux ( Importance d'allocation des ressources dans cloud computing et stratégies, techniques et politiques d'allocation des ressources) sont discutés ainsi que les principaux techniques (Comme la logique floue et le système multi-agents), afin de fournir des ressources à la demande avec l'avantage de leur meilleure utilisation, en réduisant les coûts au profit du client.

## Chapitre III.

# Une Approche Intelligente Basée sur CSP pour l'Allocation des Ressources dans Cloud Computing



## III.1 Introduction

Le cloud est défini comme un réseau d'accès à la demande. Il permet aux clients d'utiliser un ensemble de ressources informatiques telles que le réseau, le stockage et les applications [94]. Il est également connu sous le nom de services virtuels cloud, qui incluent les infrastructures en tant que service (IaaS), la plateforme en tant que service (PaaS) et le logiciel en tant que service (SaaS) [95] [96]. Dans le cloud computing, le processus d'allocation des ressources consiste à planifier les cloudlets (tâches client), où le problème important est d'attribuer des cloudlets aux machines virtuelles (VM) optimales appropriées en tenant compte des particularités de l'infrastructure du système et des exigences du client. Les principales particularités de l'infrastructure cloud sont l'hétérogénéité, l'équilibrage de la charge du datacenter, la capacité des ressources et la politique des ressources. De l'autre côté, les principales exigences des clients sont un temps d'exécution court et un faible paiement. L'équilibrage de charge du système est un facteur important pour satisfaire les exigences des clients. Lorsque la charge est inférieure, le temps d'exécution sera court et le paiement sera bon marché [97] [44] [98] [99]. Dans ce travail, nous suggérons des solutions à deux problèmes principaux d'allocation des ressources. Le premier problème survient lors de la soumission de cloudlets aux VM dans les hôtes. Pour le résoudre, nous utilisons la logique floue (LF). Le deuxième problème réside dans l'hébergement de nouvelles VMs dans les hôtes de différents datacenters en fonction de leurs charges. Pour résoudre ce problème, nous proposons une solution flexible en utilisant deux techniques : le système multi-agents (SMA) et CSP. Pour chaque solution, deux algorithmes (internes et externes) sont distribués à deux niveaux de l'infrastructure cloud (où l'algorithme externe pour le niveau des centres de données et l'algorithme interne pour le niveau des hôtes). Dans cette étude, les ressources des demandes clients (Ram, Cpu, Budget ...) ont un caractère dynamique. Ils se manifestent sous forme d'intervalles (par exemple Ram: [min, max], Payment: [min, max], Cpu: [min, max]). La réponse proposée (solutions) est représentée sur un terme des triples (Ram, Mips , Paiement), qui nécessite une utilisabilité de taux (Score) pour sélectionner la meilleure réponse. Cela illustre l'importance d'utiliser le LF pour faciliter la formulation des demandes des clients et pour sélectionner la meilleure solution dans un ensemble de solutions pour un cloudlet. LF [100] est proposé par Zadeh en (1965) comme un outil formel pour traiter des systèmes intelligents caractérisés par des informations insuffisantes et incomplètes [101]. Le système multi-agents est utilisé pour assurer une gestion efficace des particularités de l'infrastructure physique du cloud telles que l'hétérogénéité et la distribution dues à son intelligence, son autonomie, sa réactivité, sa mobilité et sa capacité à effectuer un ensemble de décisions et de tâches [102]. Pour réduire la complexité de la politique et les restrictions de l'allocation des ressources dans le cloud, nous appliquons la technique des problèmes de satisfaction de contraintes (CSP) [103]. Les trois concepts de base du CSP sont (a) un

ensemble de variables, (b) des domaines ou des plages pour définir ces variables, et (c) des contraintes sur les variables. Ces concepts doivent être satisfaits pour résoudre les problèmes de CSP.

Ce chapitre permet de d´écrire nos contributions, d'expliquer leurs démarches, de détailler leurs différentes phases, et de décrire les algorithmes nécessaires, ainsi que les diagrammes pour modéliser les démarches de l'ensemble des différentes étapes.

## III.2 Problème d'allocation des ressources

Cette section est consacrée à la discussion de la problématique de l'allocation des ressources suivant notre point de vue, on présente alors une description de contexte et les concepts importants et conclut par une définition formelle de la problématique et des questions pertinentes qui ont le fondement de notre contribution

### III.2.1 Spécification du problème

Les problèmes d'affection de tâches et d'allocation de ressources dans un contexte hétérogène comme le cloud sont des problèmes difficiles. Par conséquent, l'approvisionnement en ressources dans le cadre du cloud computing permet la variation du type et de la quantité des ressources à utiliser pour l'exécution des processus en question. Habituellement, l'ordonnancement de tâches est le processus d'affectation des tâches aux ressources disponibles sur la base des caractéristiques et des conditions des tâches.

En général, l'allocation des ressources est un problème d'optimisation combinatoire. Il est défini par le type NP-complets (non déterministe polynomial). Par conséquent, le type de sa solution est soit heuristique, soit métaheuristique selon la complexité du problème [104]. Dans cette thèse, nous considérons une extension du modèle d'allocation des ressources, qui présente des contraintes.

Il y a ***M*** machines virtuelle***s*** (***VM***) à héberger dans ***K*** machines physiques (***hôtes*** ou ***hôtes***). Ces hôtes sont distribués sur différents ***datacenters*** du Cloud. On résume les règles de gestion dans les points suivants :

- Il existe également ***N*** clients dont leurs demandes (également appelées ***cloudlets***) doivent être soumises aux ***VM.***
- Chaque cloudlet de client demande un ensemble de ressources informatiques: CLj (ram, stockage, cpu), j = 1, 2,. . ., n.
- Chaque cloudlet de client a un budget total: CLj (budget), j = 1, 2,. . ., n.
- Chaque machine virtuelle a une capacité de ressources informatiques virtuelles allouées: VMi (ram, stockage, cpu), i = 1, 2,. . ., m.
- Chaque ***hôte*** a une capacité de caractéristiques informatiques physiques: $Host_h$ (ram, stockage, cpu, coût, énergie), h = 1, 2, ..., k.

- Chaque ***hôte*** a un prix unitaire pour l'exploitation de chaque ressource de calcul: $Price_h$ (ram), $Price_h$ (stockage), $Price_h$ (cpu), h = 1, 2,. . ., k.

Autrement dit, nous nous concentrons sur le problème d'allocation des ressources, qui est divisé en deux niveaux:

1) Nous devons trouver la meilleure VM pour soumettre chaque cloudlet en respectant la capacité de VM, où chaque VM peut soumettre plus d'un cloudlet.
2) Nous devons trouver le meilleur hôte pour héberger chaque machine virtuelle en respectant la capacité de l'hôte, où chaque hôte a la capacité d'héberger plus d'une machine virtuelle en fonction de sa capacité et son énergie.

### III.2.2 Coût de la consommation d'énergie et paiement des ressources

Le coût est un facteur décisionnel important soit pour le fournisseur de services (datacenters), soit pour le consommateur de services (client cloud). Pour le fournisseur, le coût opérationnel est principalement lié à la consommation d'énergie des ressources physiques dans l'infrastructure de cloud computing (hôtes). Alors que pour le client, le paiement est lié au coût des ressources virtuelles consommées dans les machines virtuelles telles que le CPU, la RAM et le Stockage [105].

Dans ce travail, nous utilisons le terme de **Coût** pour indiquer le au coût de consommation d'énergie, et le terme de **Paiement** pour exprimer les dépenses sur l'exploitation ressources virtuelles consommées par les clients du cloud.

### III.2.3 Critères d'allocation

Avant de commencer notre conception, nous devons déterminer les critères d'allocation des ressources qui sont utilisés dans le cloud computing. Nous présentons les principaux critères utilisés dans le cloud computing :

*Paiment*

Le paiement de cloudlet est calculé par l'utilisation des ressources par unité de temps. Dans la solution d'allocation de ressources fournie, le paiement ne doit pas dépasser le budget du cloudlet.

*RAM*

Chaque demande de RAM est présentée sur l'intervalle **[minRAM, maxRAM]**, **minRAM** est la valeur demandé de RAM la plus basse, tandis que la **maxRAM** est la valeur la plus élevée de la RAM. Les deux sont définies par le client en fonction de la demande de cloudlet.

*Mips*

Représente l'unité de capacité demandée de CPU, elle est définie en termes de classement en millions d'instructions par seconde.

Chaque demande de Mips est présentée sur intervalle **[minMIPS, maxMIPS]**. Le **minMIPS** est la valeur la plus basse de MIPS tandis que **maxMIPS** est la valeur la plus élevée de MIPS. Les deux sont définis par le client en fonction de la demande de cloudlet.

*Taux d'utilisabilité de la solution (score):*

Dans IaaS [105], l'utilisabilité de la solution est représentée en termes de taux (Paiement, RAM et Mips). Il est considéré comme la meilleure utilisabilité de la solution, c'est-à-dire que le système cherche à maximiser la valeur du taux d'utilisabilité (intervalle [0-1]).

## III.3 Architecture globale du système d'allocation de ressources

L'architecture de notre système est illustrée dans Figure III.1. Dans ce qui suit, nous décrivons les différentes couches de l'architecture.

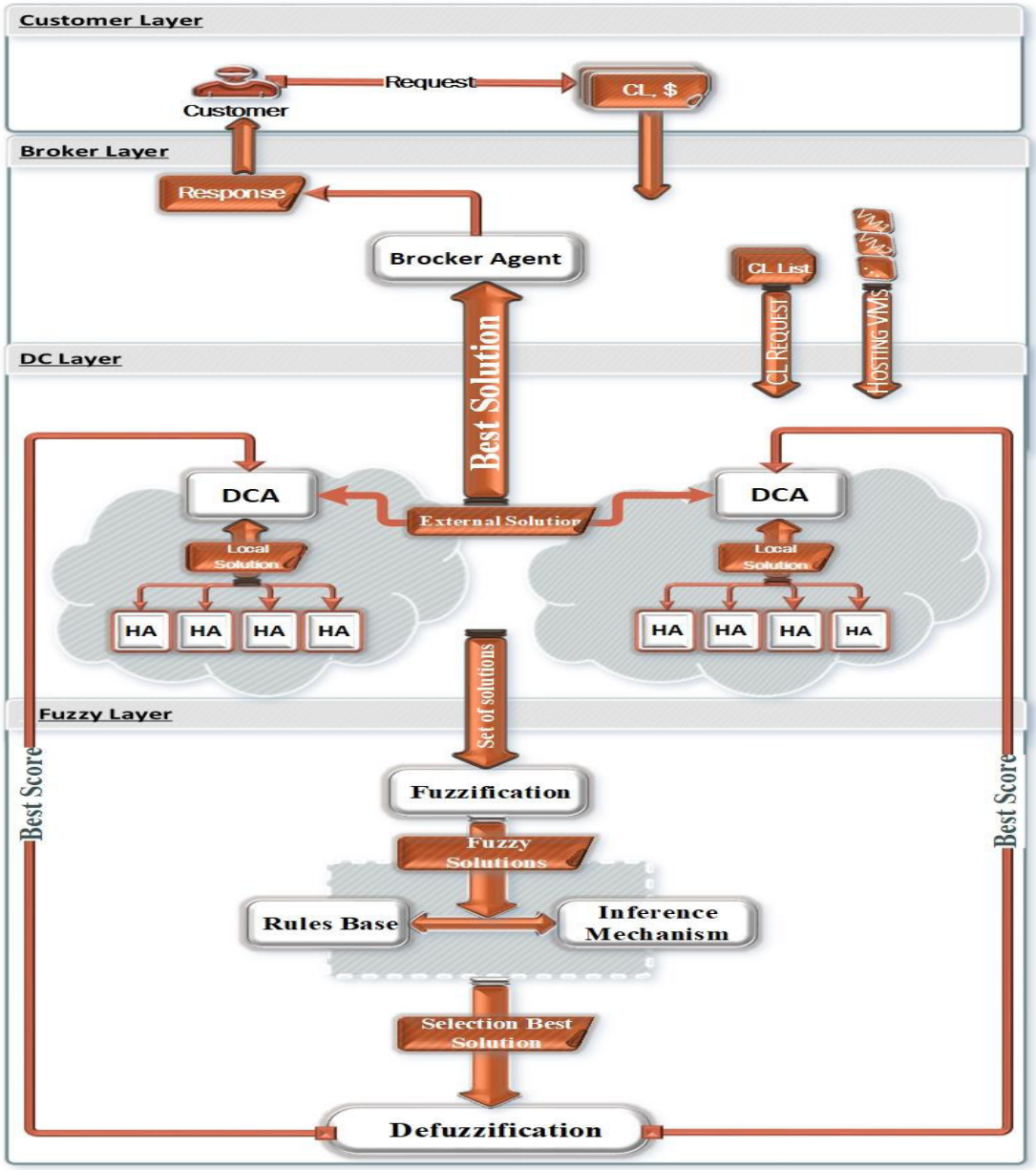



**Figure III. 1 Architecture du système d'allocation des ressources**

### III.3.1 Couche Client (Customer Layer)

Dans cette couche, le système identifie les demandes des clients dans le cloud. Ces demandes sont présentées en termes d'intervalles de ressources cloudlet et de son budget (CL, $) comme dans la Tableau III.1.

**Tableau III. 1 Format des ressources d'une cloudlet**

| User | ID |
|---|---|
| ClouletId | |
| Ram | minRam |
| | maxRam |
| Mips | minMips |
| | maxMips |
| Bw | |
| storage | |
| length | |
| Budget | max Payment |

### III.3.2 Couche de courtier (Broker Layer)

Cette couche contient une liste de VM libres et l'agent de courtage (Broker Agent ou BA) qui gère l'utilisation, les performances et la livraison des ressources cloud. Le rôle de cette couche est la médiation entre la couche Datacenter et la couche client.

### III.3.3 Couche de datacenter (Datacenter Layer)

Il contient des agents de datacenter (Datacenter Agent ou **DCA**), des agents hôtes (Host Agent ou **HA**) et des agents **HOffA** (pour tout un hôte à l'état **OFF**). Les principales opérations dans cette couche sont la soumission de Cloudlets et l'optimisation.

L'opération de soumission de Cloudlets vise à rechercher la meilleure allocation de ressources pour la demande client sur deux niveaux : Local (entre agents HA dans le même datacenter) et Externe (entre agents DCA du cloud). De plus, l'opération d'optimisation vise à optimiser l'hébergement des VM dans les hôtes physiques pour équilibrer la charge et la consommation d'énergie.

### III.3.4 Couche floue (Fuzzy Layer)

Le rôle de cette couche est la prise en charge de trouver une solution locale entre les agents HA d'un même datacenter. Cette opération utilise la logiques floue pour évaluer et sélectionner la meilleure solution dans l'ensemble de solutions pour chaque cloudlet. Cette couche se compose de trois principaux axes :

### III.3.4.1 Fuzzification

C'est le processus de conversion d'une valeur réelle en une valeur floue via certaines fonctions d'appartenance (Membership) [106] [107] telles que la valeur Ram, nous convertissons en termes de (**Low, fair, medium, high**) comme dans la Figure III.2.

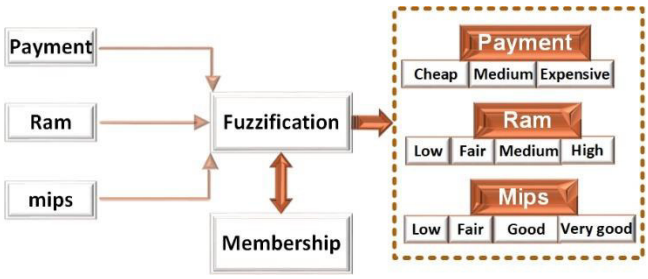



**Figure III. 2 Architecture du module de Fuzzification**

Dans la littérature, plusieurs types de fonction d'appartenance (Membership Function ou MF) permettent d'effectuer une fuzzification. Gaussien, triangulaire et trapézoïdal. Les fonctions d'appartenance sont un point très important dans la conception du FL [108].

Dans cette thèse, nous choisissons la fonction triangulaire MF tel que décrit dans le Tableau III.2.

**Tableau III. 2 Critères RA sur nombre flou**

| Critères | Terme linguistique | Nombre flou triangulaire |
|---|---|---|
| Ram | low | [0,0,5] |
| | fair | [3,8,10] |
| | medium | [7,15,20] |
| | high | [17,25,30] |

À la fin, nous obtenons l'ensemble des propositions de solutions (pour le même cloudlet) du datacenter. Cet ensemble passe par la fuzzification pour servir d'entrée pour l'inférence floue.

### III.3.4.2 Inférence floue

L'inférence floue reçoit l'entrée sous forme de matrice de la fuzzification comme indiqué dans la relation (1) [32].

$$\tilde{M} = \begin{vmatrix} \tilde{X}_{11} & \tilde{X}_{12} & \tilde{X}_{1n} \\ \tilde{X}_{21} & \tilde{X}_{22} & \tilde{X}_{2n} \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \tilde{X}_{m1} & \tilde{X}_{m2} & \tilde{X}_{mn} \end{vmatrix} \quad (1)$$

Où, *S1, S2, ..., Sm* sont les solutions des hôtes, *R1, R2, ..., Rn* sont les paramètres des ressources (Ram, mips, Payment) et *Xij* est une valeur floue présentant la valeur de la ressource *Rj* pour les solutions *Si* . Après avoir défini la matrice de l'inférence floue, nous définissons le niveau de satisfaction des règles via les règles de production de l'inférence floue comme le montre la Figure III.3 [109].

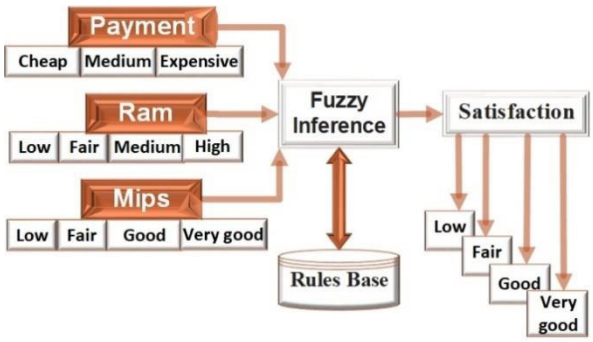



**Figure III. 3 Architecture du module d'inférence floue**

Les règles floues sont toujours écrites sous la forme de structure suivante :

RULE 1: ***IF*** ram is ***low AND*** mips is ***low AND*** Payment is ***cheap THEN*** score is ***low***;

RULE 2: ***IF*** ram is ***medium AND*** mips is ***fair AND*** Payment is ***cheap THEN*** score is ***good***;

..

RULE 48: ***IF*** ram is ***high AND*** mips is ***good AND*** Payment is ***medium THEN*** score is ***very_good***;

### III.3.4.3 Défuzzification

Après avoir obtenu la sortie d'appartenance, nous convertissons la sortie d'appartenance (variables linguistiques) en une valeur réelle (Defuzzification).

Nous pouvons conclure que la défuzzification est un processus inverse de la Fuzzification. C'est le processus de combinaison des sorties floues afin de donner une valeur réelle on continue [110].

1) Il existe plusieurs façons de traiter la défuzzification, telles que :
   (1) Méthode du centre des sommes (COS): (Center of sums method, en anglais)
   (2) Méthode Centre de gravité (COG) : (Center of gravity method, en anglais)
   (3) Méthode du centre de gravité de la zone (COA) : (Centroid of area method, en anglais)

(4) Méthode Bisector of Area (BOA) : (Bisector of area method, en anglais)
(5) Méthode de la moyenne des maxima (MOM) : (Mean of maxima method, en anglais)

Dans notre travail, nous utilisons deux méthodes COG et MOM car elles sont largement utilisées et efficaces.

- **Centre de gravité (COG)**

Le principe de base de la méthode est de trouver le centre de gravité de la zone délimitée par la sortie du contrôleur MF [111]. Mathématiquement, le **COG** peut être défini par[32]:

$$\mathbf{F}_{\mathbf{COG}} = \frac{\int \boldsymbol{x}\, \boldsymbol{c}(\boldsymbol{x})\, \boldsymbol{dx}}{\int \ \boldsymbol{c}(\boldsymbol{x})\, \boldsymbol{dx}} \qquad (2)$$

Où,:

$\mathbf{F}_{\mathbf{COG}}$ est le résultat de la défuzzification.

**C** fonction d'appartenance.

**x** est la variable de sortie.

- **Méthode de la moyenne des maxima (MOM)**

Le but de cette méthode est de déterminer la moyenne arithmétique de tous les maximums obtenus à la surface de l'élément de sortie [32]. Mathématiquement, le MOM peut être exprimé comme suit [112]:

$$\mathbf{F}_{\mathrm{MOM}} = \sum_{j=1}^{l} \frac{w_j}{l} \qquad (3)$$

Où,:

**MOM** est le résultat de la défuzzification

**L** Limite maximale de défuzzification

**W** est la variable de sortie

## III.4 DCSP Modulation

Dans cette section, le problème DCSP est formulé. Les variables et les contraintes sont distribuées à plusieurs agents. Chaque agent dans SMA fait son plan de proposition (solution) en utilisant la négociation distribuée et en satisfaisant ses contraintes. Les détails sur les principales idées de l'approche sont discutés dans les sous-sections.

Les différentes variables et contraintes sont identifiées. Après cela, le scénario de calcul est décrit en conséquence.

### III.4.1 Définition des variables

Dans cette section, nous montrons les variables et leur définition Tableau III.3.

**Tableau III. 3 Définition des variables**

| Variable | Définition | Domaine |
|---|---|---|
| **D** | Nombre de Datacenters | Nombre Naturel |
| **H** | Nombre de the hôtes dans Datacenter | Nombre Naturel |
| **P** | Nombre de CPUs dans the hôte | Nombre Naturel |
| **VM** | Machine virtuelle | $\{vm_1, \dots, vm_l, \dots vm_v\}$ |
| **V** | Nombre de Machine virtuelles | Nombre Naturel |
| $\mathbf{V_j}$ | Ensemble de VMs hébergées dans l'hôte *J*. | $\{ V_1, \dots, V_j, \dots V_v \}$ |
| $\mathbf{SV_j}$ | sous ensemble de VMs hébergées (dans the hôte *J)* sélectionnées pour exécuter un ensemble de cloudlets. | $\{ SV_1, \dots, SV_j, \dots SV_v \}$ |
| $\mathbf{MigV_{j\,j'}}$ | sous ensemble de VMs hébergées (dans le hôte *J*) sélectionnées pour être migrées vers l'hôte *J'*. | $\{ V_1, \dots, V_j, \dots V_v \}$ |
| **C** | Nombre de cloudlets | Nombre Naturel |
| **DC** | Datacenter | $\{ dc_1, \dots, dc_i, \dots dc_d\}$ |
| **Host** | Hôte _j dans DC_i | $\{ Host_{11}, \dots, Host_{ij}, \dots Host_{dh}\}$ |
| **Variable** | **Définition** | **Domaine** |
| **PE** | CPU dans un hôte | $\{ Pe_{11}, \dots, pe_{jk}, \dots pe_{hp}\}$ |
| **PEV** | CPU dans une Machine virtuelle | $\{ Pev_{11}, \dots, pev_{lk}, \dots pev_{vp}\}$ |
| **CL** | Cloudlet | $\{cl_1, \dots, cl_m, \dots cl_c\}$ |
| **Host (ram)** | Capacité physique de RAM d'un hôte | Nombre Naturel |
| **Host(bw)** | Capacité physique de la bande passante d'un hôte. | Nombre réel |
| **Host(Storage)** | Capacité physique de stockage d'un hôte. | Nombre Naturel |
| **Host(nbr_pe)** | Nombre de CPUs dans the hôte. | Nombre Naturel |
| **Host(state)** | L'état de l'hôte. | {ON, OFF} |
| **Host(mips)** | Somme des Capacités de tous les CPUs dans un hôte | Nombre réel |
| **Host(used_mip)** | Somme des Capacités des CPUs alloués par des Machines virtuelles hébergée dans un hôte. | Nombre réel |
| **Host(mips_load)** | La charge de l'hôte, la capacité de CPUs utilisée conformément à la capacité totale de CPUs dans l'hôte. | Nombre réel (%) |
| **Host(Cost_ON)** | Coût de démarrage d'un hôte. | Nombre Naturel |
| **Host(Cost)** | Coût d'un hôte : coût total des CPUs alloués par des Machines virtuelles dans l'host (inclut le coût de démarrage). | Nombre Naturel |
| **Host(oldCost)** | Coût de l'hôte avant l'optimisation | Nombre Naturel |
| **Host(newCost)** | Coût de l'hôte après l'optimisation | Nombre Naturel |
| **Pe(mips)** | Capacité du CPU en Mips | Nombre Naturel |
| **Variable** | **Définition** | **Domaine** |
| **Host(mips_price)** | Prix unitaire de mips dans l'hôte. Obtenu à partir du modèle proposé pour chaque hôte | Nombre réel, |
| **VM(size)** | Capacité du stockage alloué par une VM | Nombre Naturel |
| **VM(ram)** | Capacité de la RAM allouée par une VM | Nombre Naturel |
| **VM(bw)** | Bande passante allouée par une VM | Nombre réel |
| **VM(nbr_pe)** | Nombre de CPUs allouée par une VM | Nombre Naturel |
| **VM(mips)** | Capacité des CPUs alloués par une VM | Nombre réel |

| | | |
|---|---|---|
| **CL(length)** | Langueur du programme de la Cloudlet. | Nombre réel |
| **CL(fithe size)** | Taille totale des fichiers d'une Cloudlet. | Nombre réel |
| **CL(output size)** | Taille du résultat de l'exécution de CL | Nombre réel |
| **CL(nbr_pe)** | Nombres de CPUs alloués pour une Cloudlet. | Nombre Naturel |
| **CL(mips)** | Capacité des CPUs alloués pour une Cloudlet. | Nombre réel |
| **Dem(mips)** | Capacité des CPUs demandée par une Cloudlet.(Client) | Nombre réel |
| **Cost** | Le coût d'utilisation de la capacité des CPUs dans un hôte (coût de la charge). | Devise ($) |
| **Cl(budget)** | La valeur de paiement maximale d'une cloudlet, (définie par l'utilisateur) | Devise ($) |
| **Cl( payment )** | Le coût total des ressources à utiliser par une solution proposée pour une cloudlet, (calculé par le système). | Devise ($) |
| **Sol(Cost)** | Le coût de l'énergie d'une solution dans une Hôte | Devise ($) |
| **Variable** | **Définition** | **Domaine** |
| **Sol(Score)** | Le score de solution. Il est utilisé dans le processus de classement des solutions, basé sur la logique floue | Nombre réel positif dans [0 ... 1] |
| **minLoad** | La valeur minimale de l'énergie dans l'hôte optimisé | Nombre réel % |
| **maxLoad** | La valeur maximale de l'énergie dans l'hôte optimisé | Nombre réel % |
| **maxRam** | La valeur maximale demandée de la RAM | Nombre réel |
| **minRam** | La valeur minimale demandée de la RAM | Nombre réel |
| **maxMips** | La valeur maximale demandée des Mips | Nombre réel |
| **minMips** | La valeur minimale demandée des Mips | Nombre réel |
| $E(Cl_m, Host)$ | L'hôte élu pour exécuter la cloudlet m | @IP, name de domain, … |

### III.4.2 Définition de contraintes

Dans cette section, on propose un ensemble de contraintes en fonction les variables définies auparavant qui correspondent aux exigences du système d'allocation des ressources.

***Contrainte 1:*** pour permettre une machine virtuelle ***l*** d'exécuter un ensemble de ***M*** cloudlets:

$$\sum_{m=1}^{M}\left(Cl_m(minRam)\right) \leq VM_l(ram) \quad (4)$$

$$\sum_{m=1}^{M} Cl_m(file\ size) \leq VM_l(storage) \quad (5)$$

$$\sum_{m=1}^{M} Cl_m(bw) \leq VM_l(bw) \quad (6)$$

$$\sum_{m=1}^{M} Cl_m(minMips) \leq VM_l(mips) \quad (7)$$

*Où:*

$$VM_l(mips) = \sum_{k=1}^{P} PEV_{lk}(mips) \quad (8)$$

***Contrainte 2:*** pour permettre une machine virtuelle ***l*** d'exécuter un nouveau cloudlet ***m'***, sachant que cette machine exécute déjà un ensemble de ***M*** cloudlets :

$$(minRam) + \sum_{m=1}^{M}\left(Cl_m(minRam)\right) \leq VM_l(ram) \quad (9)$$

$$Cl_{m'}(file\ size) + \sum_{m=1}^{M} Cl_m(file\ size) \leq VM_l(storage) \quad (10)$$

$$Cl_{m'}(bw) + \sum_{m=1}^{M} Cl_m(bw) \leq VM_l(bw) \quad (11)$$

$$Cl_{m'}(minMips) + \sum_{m=1}^{M} Cl_m(minMips) \leq VM_l(mips) \quad (12)$$
$$and \ \boldsymbol{m'} \ \neg\epsilon \ [1, \boldsymbol{M}].$$

***Contrainte 3:*** pour permettre un hôte j d'héberger un ensemble de V machines virtuelles:

$$\sum_{l=1}^{V} VM_l(ram) \leq Host_j(ram) \quad (13)$$
$$\sum_{l=1}^{V} VM_l(storage) \leq Host_j(storage) \quad (14)$$
$$\sum_{l=1}^{V} VM_l(bw) \leq Host_j(bw) \quad (15)$$
$$\sum_{l=1}^{V} VM_l(mips) \leq Host_j(mips) \quad (16)$$

*Où:*

$$Host_j(mips) = \sum_{i=1}^{P} PE_i(mips) \quad (17)$$

***Contrainte 4::*** pour permettre à l'hôte ***J*** d'héberger une nouvelle machine virtuelle ***l'***, sachant que cet hôte héberge déjà un ensemble de ***V*** machines:

$$VM_{l'}(ram) + \sum_{l=1}^{V} VM_l(ram) \leq Host_j(ram) \quad (18)$$
$$VM_{l'}(storage) + \sum_{l=1}^{V} VM_l(storage) \leq Host_j(storage) \quad (19)$$
$$VM_{l'}(bw) + \sum_{l=1}^{V} VM_l(bw) \leq Host_j(bw) \quad (20)$$
$$VM_{l'}(mips) + \sum_{l=1}^{V} VM_l(mips) \leq Host_j(mips) \quad (21)$$
$$et \ \boldsymbol{l'} \ \neg\epsilon \ [1, \boldsymbol{V}]$$

***Contrainte 5:*** elle représente le classement des agents hôtes en fonction du score. En cas de préférence entre deux solutions qui ont la même valeur de score, nous choisissons la solution dont son hôte a la valeur **mipsLoad** la plus basse:

$$Best \ Sol \equiv min(Sol(Score)) \quad (22)$$
$$if: Sol_j(Score) \equiv Sol_{j'}(Score) \quad then \quad (23)$$
$$Best \ Sol \equiv min(Host_j(mipsLoad), Host_{j'}(mipsLoad)) \quad (24)$$

*Where:*

$$Host_j(mipsLoad) = \frac{Host_j(usedMips)}{Host_j(mips)} \quad (25)$$
$$Host_j(usedMips) = \sum_{l=1}^{N} VM_{j\,l}(mips) \quad (26)$$

***Contrainte 6:*** nous disons qu'un hôte ***j*** (héberge un ensemble de ***N*** machines virtuelles) doit être optimisé s'il vérifie la condition suivante:

$$Host_j(mipsLoad) \leq Host_j(minLoad) \quad (27)$$

$$Ou$$

$$Host_j(mipsLoad) \geq Host_j(maxLoad) \quad (28)$$

***Contrainte 7:*** pour permettre d'optimiser un hôte ***j*** par consolidation à l'aide d'un autre hôte **j':**

$$Host_j(mipsLoad) \leq Host_j(minLoad) \quad (29)$$

$$Host_j(newLoad) = 0\% \quad (30)$$

$$Host_{j'}(newLoad) \leq Host_{j'}(maxLoad) \quad (31)$$

$O$ù :

$$Host_j(newLoad) = Host_j(mipsLoad) - MigV_{j\,j'}(mips) \quad (32)$$

$$Host_{j'}(newLoad) = Host_{j'}(mipsLoad) + MigV_{j\,j'}(mips) \quad (33)$$

***Contrainte 8:*** elle est utilisée pour permettre d'optimiser un hôte en équilibrant la charge à l'aide d'un autre hôte ***j'*** :

$$Host_j(mipsLoad) \geq Host_j(maxLoad) \quad (34)$$

$$Host_{j'}(newLoad) \leq Host_{j'}(maxLoad) \quad (35)$$

$$Host_j(newCost) + Host_{j'}(newCost) \leq Host_j(oldCost) + Host_{j'}(oldCost) \quad (36)$$

***Contrainte 9:*** elle assure qu'une solution soit valable pour une cloudlet, son paiement ne doit pas dépasser le budget proposé par le client:

$$CL(payment) \leq CL(budget)) \quad (37)$$

## III.5 Modélisation de SMA

Nous présentons dans cette section les rôles des agents du système (Tableau III.4) et leurs architectures internes. Ces architectures ayant une forme en composants, et cela pour bénéficier de ses avantages tels que, l'adaptabilité, l'évolution et la réutilisation, chacun implémente une partie des fonctionnalités d'agent en ajoutant un raisonnement décentralisé afin de fournir l'autonomie et la flexibilité aux agents.

**Tableau III. 4 Rôles des agents du système**

| Agent | Définition | Rôle |
|---|---|---|
| **BA** | Broker Agent (Agent Courtier) | Il gère la liste des VMs libres et reçoit les demandes du client et identifie leurs critères (MIPS, RAM, Budget). De plus, il est chargé de diffuser les commandes aux DCA et afficher leur réponse au client. |
| **DCA** | Datacenter Agent | Son rôle est le suivi l'état de l'infrastructure du datacenters. DCA reçoit la solution interne qui est établie par les agents HA dans le même datacenter. Aussi, Les agents DCA assurent la négociations externes entre les datacenters afin d'établir un solution externe au niveau de cloud. |
| **HA** | Host Agent (Agent Hôte) | Chaque HA Gère une machine physique (hôte), qui se caractérise par ses ressources en termes de (mémoire, MIPS, stockage, …). HA exécute des algorithmes de l'allocation des ressources pour la soumission des cloudlets et l'hébergement des machines virtuelles dans l'hôte. |
| **HOffA** | Host OFF Agent | C'est un agent HA dont l'hôte a été suspendu |

### III.5.1 Architecture interne de l'agent BA

L'architecture interne de l'agent BA et les principaux composants qui permettent l'implémentation de cet agent sont les suivants (Figure III.4) :

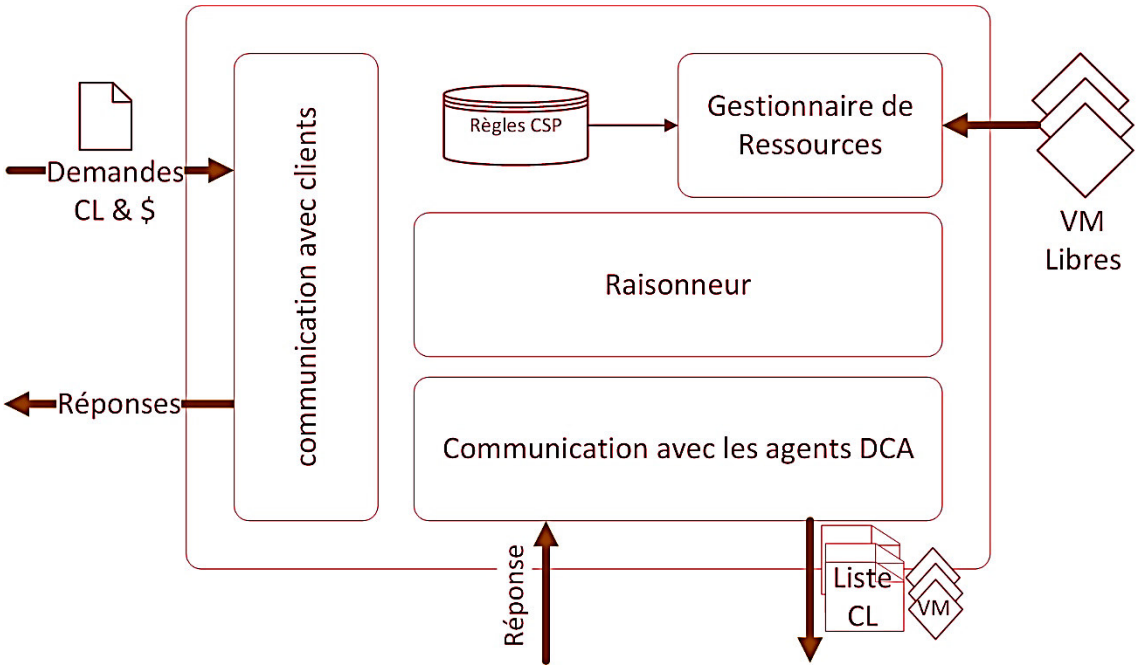



**Figure III. 4 Architecture interne de l'agent BA**

### III.5.2 Composant raisonneur

Le raisonneur joue le rôle de connecteur entre les autres composants de l'agent. Il permet à l'agent de sélectionner la meilleure action à exécuter parmi l'ensemble des plans liés aux variations du contexte d'exécution, cette sélection est basée sur des mesures multicritères.

### III.5.3 Composant d'Interaction avec clients

Grace à ce composant, l'agent BA offre un moyen au client pour introduire sa demande des ressources en termes de (cloudlets et Budgets) et afficher la réponse de ses demandes à la base des solutions proposées par les agents DCA dans le Cloud.

### III.5.4 Composant gestionnaire des ressources

Son objectif est la gestion des ressources locales (cloudlets des clients et règles CSP) et les externes (les VM libres) afin d'obtenir une bonne gestion dans le processus de la location des de ressources.

### III.5.5 Composant de communication avec les agents

Ce composant est pour objectif d'assurer toutes les communications avec les agents DCA de SMA, il comporte des mécanismes bien déterminés à savoir, synchronisation, authentification, interaction. BA provoque un seul scénario de communication avec DCA (**BA ↔DCA**) qui est pour les sujets d'interactions **Soumission des Cloudlets** et **Hébergement des VM libres**. Dans ce scénario, l'agent BA envoie à l'agent DCA une liste de cloudlets à soumettre (ou des VM à héberger pour le cas du sujet d'hébergement des VM libres). Par conséquent, l'agent DCA répond par l'envoi d'une solution selon le résultat de l'algorithme de l'allocation de ressources appliqué pour la demande.

### III.5.6 Architecture interne de l'agent DCA

L'architecture interne de l'agent DCA et les principaux composants qui permettent d'implémenter de cet agent sont les suivants (Figure III.5):

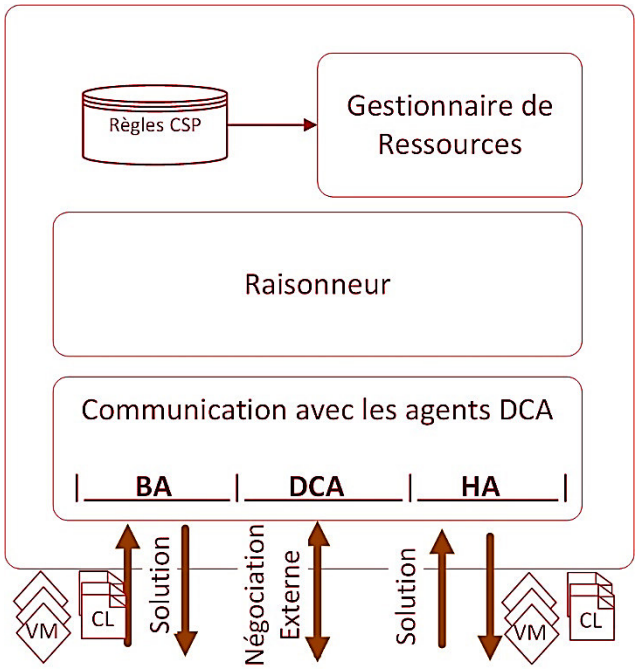



**Figure III. 5 Architecture interne de l'agent DCA**

### III.5.7 Composant gestionnaire des ressources

Les ressources gérées par ce composant sont les règles CSP et les VM libres (envoyées par BA).

### III.5.8 Composant de communication avec les agents

L'agent DCA peut interagir avec les agents BA, HA et même les autres agents DCA. Grace à ce composant, Chaque agent DCA peut provoquer des scénarios de communications avec les agents suivants :

- **Avec HA (DCA ↔ HA) :** l'agent DCA diffuse la demande reçue de l'agent BA vers agents. Puis, chaque agent HA construit sa solution selon l'algorithme appliqué et envoie sa solution au DCA après la négociation avec les autres agents HA (dans le même datacenter).

- **Avec DCA (DCA ↔ DCA) :** après avoir établi des solutions au niveau de chaque datacenter, les agents DCA partagent leurs solutions et négocient afin de construire une solution globale partagée entre ces agents.

- **Avec BA (DCA ↔ BA) :** ce scénario de communication est présenté lors de l'envoi d'une réponse d'une demande d'allocation vers l'agent BA.

### III.5.9 Architecture interne de l'agent HA

Nous présentons dans la Figure III.6 l'architecture interne de l'agent DCA et les principaux composants comme le gestionnaire de ressources, le raisonneur, le composant de l'interaction avec le système de la logique floue et la Communication avec les autres agents du système.

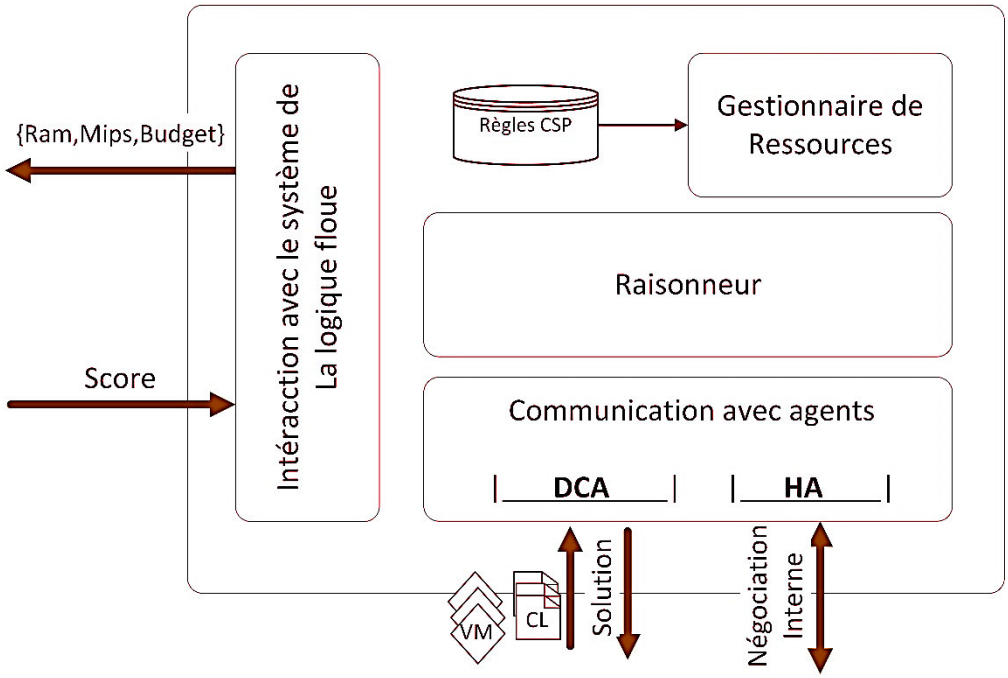



**Figure III. 6 Architecture interne de l'agent HA**

### III.5.10 Composant gestionnaire des ressources

Les ressources gérées par ce composant sont les règles CSP, l'hôte physique et les VM hébergées. En d'autres termes, il gère les ressources physiques et virtuelles.

### III.5.11 Interaction avec le système de la logique floue

Ce composant a pour fonction d'assurer l'accès à la couche de la logique floue (FL) pour calculer la valeur de score pour une solution d'allocation des ressources pour soumettre une cloudlet. Pour calculer le score, on fait appel à la fonction *score_computing ()* dans la couche FL et passe les critères (Ram, Mips, Payement).

### III.5.12 Composant de communication avec les agents

Il permet d'interagir avec les agents DCA et les autres HA dans le même datacenter. Dans ce composant on trouve les scénarios de communications suivants :

- **Avec HA (HA ↔ HA) :** après la réception de la demande, les agents HA partagent leurs solutions et négocient afin de construire une solution interne au niveau de leur datacenter.
- **Avec DCA (HA ↔ DCA) :** ce scénario de communication est présenté lors de l'envoi d'une réponse (d'une demande d'allocation) vers l'agent DCA.

## III.6 Interactions dans le système

Dans cette section, nous discutons les principales interactions entre agents qui représentent les principales fonctionnalités du processus de RA dans le système proposé. Pour cela, la Figure III.7 représente les interactions globales du système, ces interactions sont distinguées en trois sujets de négociations à savoir : **(1) Hébergement de VMs**, **(2) Soumission de cloudlets** et **(3) Processus d'optimisation**.

D'un autre côté, et tout dépendant de types d'agents (HA et DCA), on divise les sujets d'interactions (1) et (2) on deux niveaux : local et externe. Le niveau local encapsule les négociations entre les agents de type HA dans le même datacenter afin d'établir une solution interne au niveau de leur DCA. Cependant, Le niveau externe (en dehors des datacenters) décrit les négociations entre les agents de type DCA afin de composer une solution externe (solution globale de Cloud) à partir des solutions internes de ces DCA.

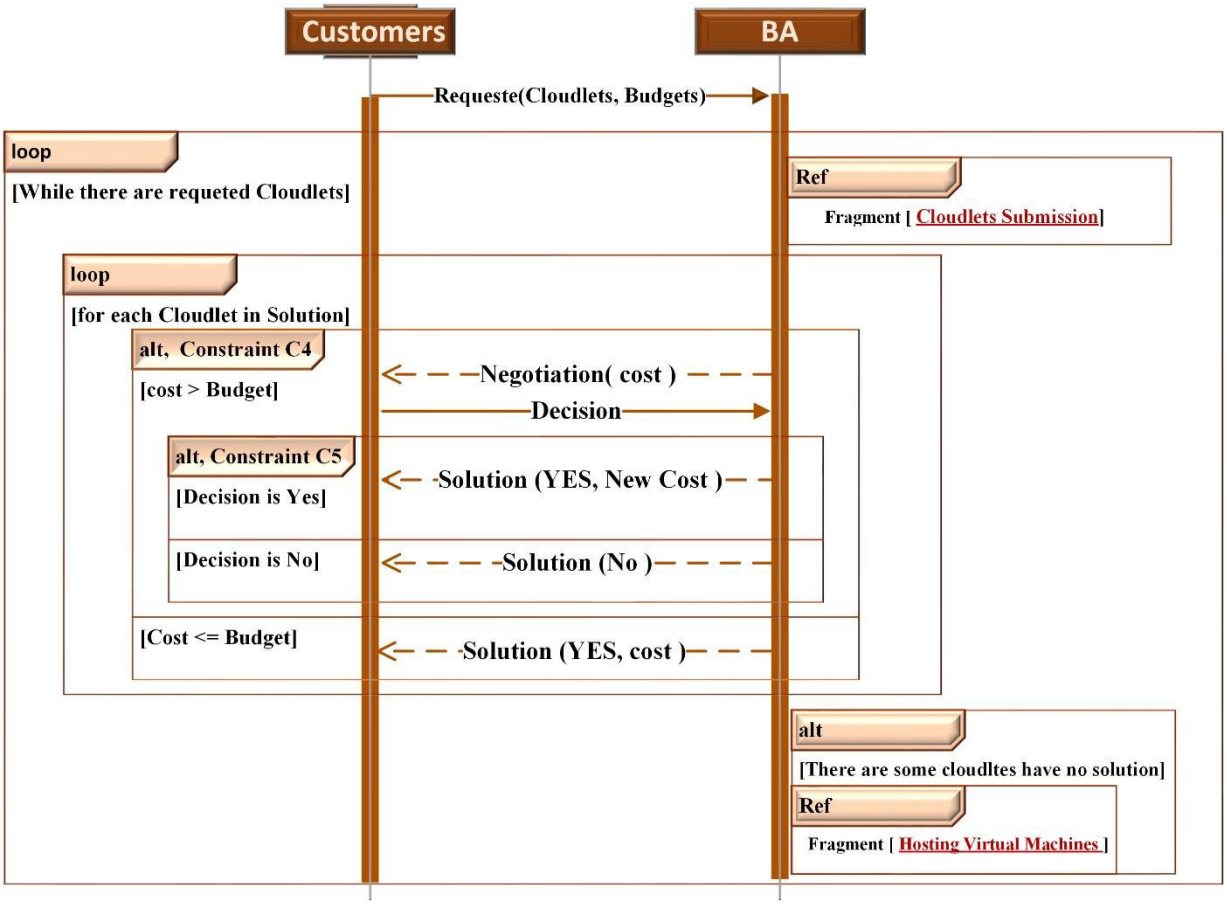



**Figure III. 7 Diagramme Interaction globale du SMA**

### III.6.1 Hébergement de VM

Dans cette division, BA diffuse une demande d'hébergement d'un ensemble de nouvelles VM aux agents DCA, par conséquent chaque DCA transmet la demande à ses agents HA., L'algorithme 1 et l'algorithme 2 illustrent les interactions entre les agents du système et leur

collaboration. Ces interactions sont résumées dans le digramme d'interaction dans Figure III.8.

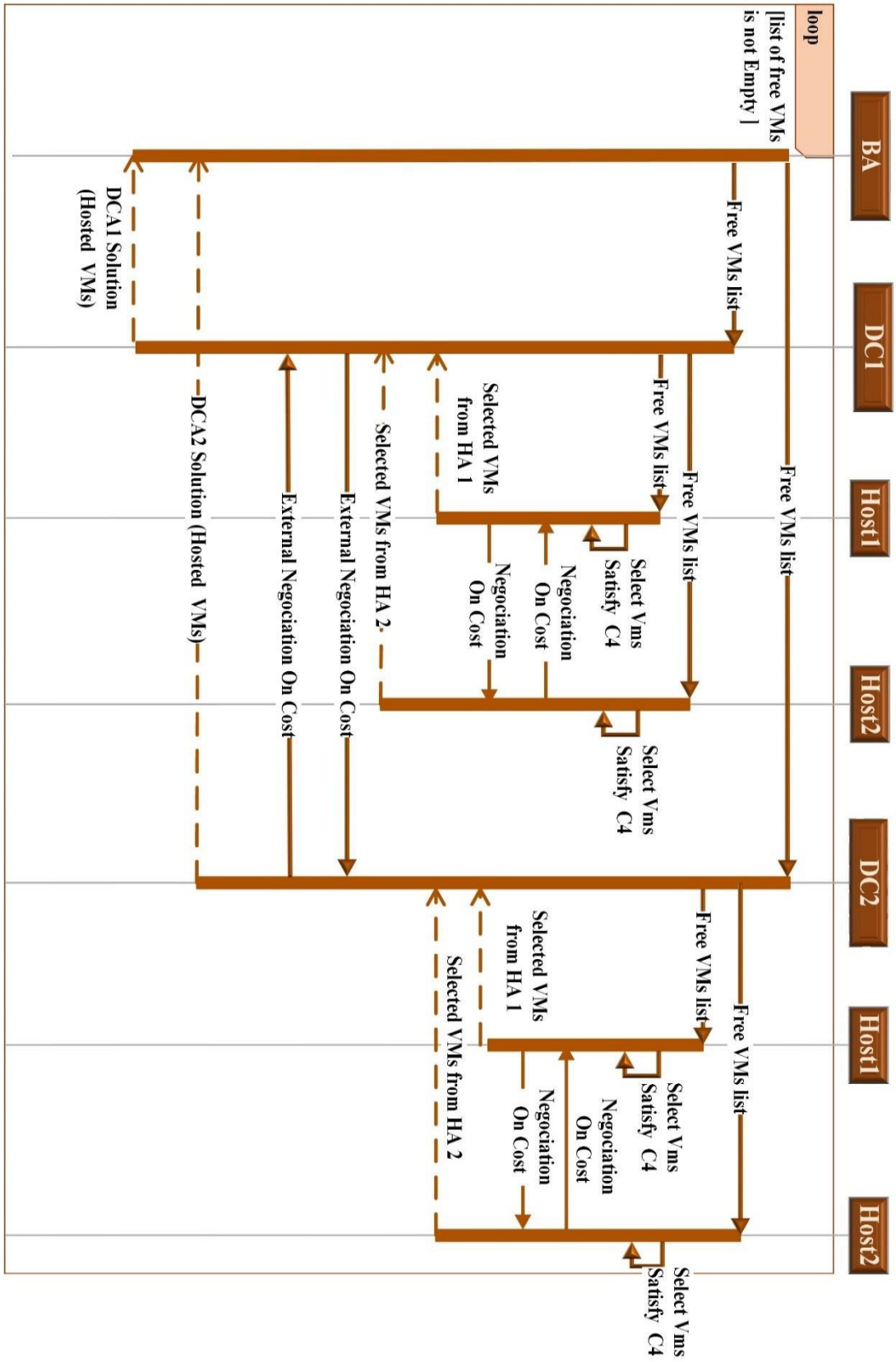



**Figure III. 8 Digramme d'interaction d'hébergement de VM**

### III.6.2 Hébergement de VM au niveau Local

Au niveau du datacenter, les agents HA collaborent afin de déterminer le meilleur hôte pour chaque VM à héberger. Les étapes de ce niveau sont détaillées dans l'**algorithme 1** ci-dessous :

**Algorithme 1 : Hébergement de VM au niveau local**

```
1  Inputs:
2  VMs: List of Virtual machines
3  Hosts: List of Hosts in the same datacentre
   Outputs:
4  LS: Local Solution it is a List of (host,vm) contains
   the selected Virtual machines with their hosts

5  LS = Ø
6  for host in Hosts do
7    for vm in VMs do
8      if (host,vm) satisfy C3 & C4 then
9        if ∃ (host', vm) in LS then
             //Negotiation between host and
             //host' on Cost
10         if(host'(Cost)>host(Cost)) then
11           remove (host',vm) from LS
12           insert (host,vm) in LS
13         end if
14       else
           //first solution for this vm
15         insert (host,vm) in LS
16       end if
17     end if
18   end for
19 end for
20 return LS
21 end.
```

### III.6.3 Hébergement de VM au niveau Externe

Afin de composer une solution externe, les agents DCA partage leurs solutions internes et négocient pour sélectionner la meilleur pour chaque VM, comme il est décrit dans l'algorithme2.

**Algorithme 2 : Hébergement de VM au niveau externe**

```
1  Input:
2  DS : List of (dc,LS(dc)) contains the datacentres and
   their local solutions
3  Output:
4  ES : External Solutions, it is a List of (dc,vm)
   contains the selected Virtual machines with their
   Datacentres
```

```
5   ES = ∅
6   for dc in DS do
7     for vm in LS(dc) do
8       if ∃ (dc',vm) in ES then
        //Negotiation between dc & dc'
        //on Cost
9         if(dc'(Cost) > dc(Cost)) then
10          remove (dc',vm) from ES
11          insert (dc,vm) in ES
12        end if
13      else
          //first solution for this vm
14        insert (host,vm) in ES
15      end if
16    end for
17  end for
18  return ES
19  end.
```

### III.6.4 Soumission de cloudlets

Après avoir hébergé les VM, BA diffuse la demande du client aux agents DCA sous la forme d'un ensemble de requêtes Cloudlets. En conséquence, chaque DCA transmet la demande à ses agents HA pour établir une solution de soumission de cloudlets comme il est illustré dans (Figure III.9).

Notre idée principale pour la soumission de cloudlets est d'introduire le module de la logique floue pour calculer le score, qui est considéré comme un critère clé servant à évaluer l'utilisabilité des solutions dans les deux niveaux d'interactions : Niveau local (**Algorithme 3)** et Niveau externe (**Algorithme 4**).

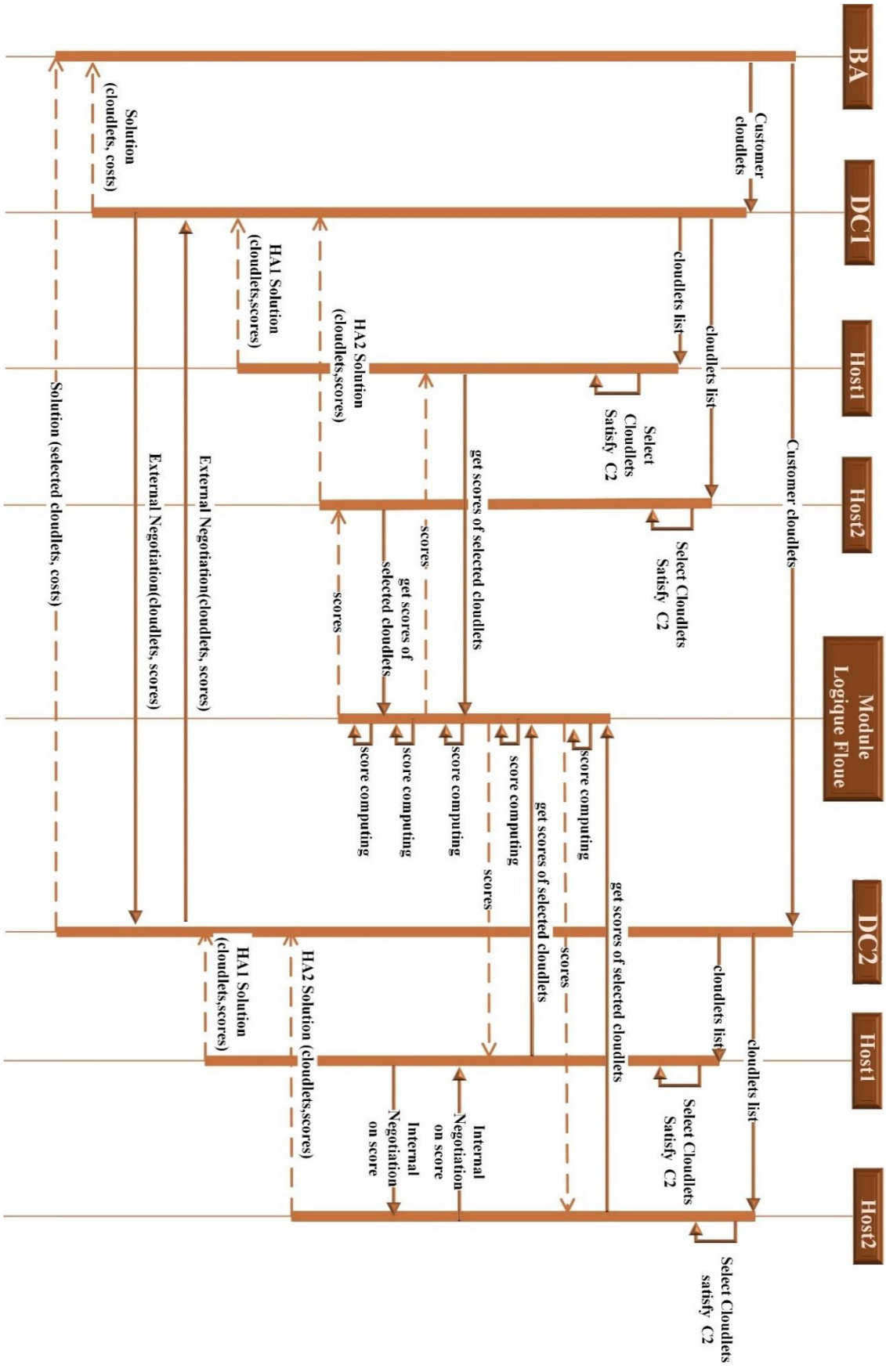



**Figure III. 9 Digramme d'interaction d'hébergement de VM**

### III.6.5 Soumission de cloudlets au niveau local

**Algorithme 3 : Soumission de cloudlets au niveau local**

```
1   Input:
2   Cloudlets: List of cloudlets
3   Hosts: List of Hosts in the same datacentre
    Output:
4   LS: Local Solution, it is a List of (host,cl) contains
    the cloudlets with their selected hosts

5   LS = ∅
6   for host in Hosts do
7    for cl in Coudlets do
8     if ∃ vm hosted in host and
9        (vm,cl)satisfy C1 and C2 then
10        calculate cl(payment)
11      if cl(payment) with cl(budget)
12         satisfy C9 then
           // using fuzzy logic
13        calculate host (Score)
14        if ∃ (host',cl) in LS then
            //Negotiation between host and host' on score
15          if host'(Score) with
16             host (Score) satisfy C5 then
17            remove (host',cl) from LS
18            insert (host,cl) in LS
19          end if
20        else
            //first solution for this cloudlet
21          insert (host,cl) in LS
22        end if
23      end if
24    end if
25   end for
26  end for
27  return LS
28  end.
```

### III.6.6 Soumission de cloudlets au niveau Externe

**Algorithme 4 : Soumission de cloudlets au niveau Externe**

```
1   Input:
2   DS : List of (dc,LS(dc)) contains the datacentres and
    their local solutions
3   Output:
4   ES: External Solution it is a List of (dc,cl) contains
    the selected cloudlets with their Datacentres

5   ES = ∅
6   for dc in DS do
7     for cl in LS(dc) do
8       if ∃ (dc',cl) in ES then
          //Negotiation between dc & dc' on Score
9         if(dc'(Score) < dc(Score))
```

```
10          remove (dc',cl) from ES
11          insert (dc,cl) in ES
12        end if
13      else
        // first solution for this cloudlet
14        insert (dc,cl) in ES
15      end if
16    end for
17  end for
18  return ES
19  end.
```

### III.6.7 Processus d'optimisation

Nous lançons le processus d'optimisation (Algorithme 5) en cas de changement d'état d'un hôte de OFF à ON, ou en cas d'une libération (cas de sortir) d'une machine virtuelle lors de la fin d'exécution de tous les cloudlets de cette VM.

**Algorithme 5 : Processus d'optimisation**

```
1   Input:
2   Config : List of (host,host(VMs)) contains the hosts
    and their hosted virtual machines.
    Output:
3   modified_Config: the modified Config list after
4   optimization
```

```
5   While ∃ host in Config satisfies C6 do
6     if ∃ host' in Config and
7          (host',host(VMs))          satisfies  C7
8     then
       //Migrate all VMs from host to host'
9       add vm to host'(VMs)
        remove (host,host(VMs)) from Config
10      shutdown host
11    else
12      for vm in host(VMs) do
          if ∃ host' in Config
            and (host', vm) satisfy C8 then
13        // migrate vm from host to host'
14          add vm to host'(VMs)
15          remove vm from host(VMs)
16        end if
17      end for
18    end if
19  end while
20  modified_Config = Config
21  return modified_Config
    end.
```

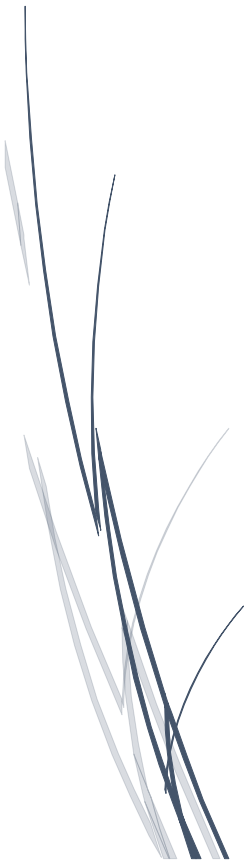
## III.7 Conclusion

Au cours de ce chapitre, nous avons décrit notre contribution conceptuelle pour l'allocation de ressources, nous avons présenté également l'architecture proposée, les différents types d'agents et leurs interactions dans le système d'allocation de ressources. Nous avons détaillé les interactions en trois types selon les sujets d'ordonnancement traités par les agents. Le premier pour gérer le problème d'hébergement des VM libre, le second type d'ordonnancement est basé sur deux algorithmes pour résoudre le problème de soumission de cloudlets et le dernière type d'ordonnancement est destinée pour équilibrer la charge entre les datacenters. Du point de vue technique, nous avons utilisé un ensemble d'algorithmes, formules et diagrammes UML pour faciliter la discussion et la présentation des idées et donner un schéma conceptuel général de notre contribution.

Dans le chapitre suivant, nous nous intéressons à la concrétisation des fonctionnalités, et cela par l'implémentation des algorithmes présentés et les différentes interprétations des résultats obtenus par la simulation.

## Chapitre IV.

# Validation et Résultats Expérimentaux



## Introduction

Ce chapitre décrit la phase de validation de notre approche proposée pour l'allocation des ressources dans le cloud computing. Elle consiste à exploiter les paradigmes de SMA, DCSP et logique floue pour concevoir un système d'allocation des ressources. Le but de ce travail est de répondre aux besoins initialement définit, qui nécessite le développement d'un système a pour objectifs : optimiser l'allocation des ressources dans le cloud computing, atteindre la satisfaction de l'utilisateur et de maximiser les bénéfices des fournisseurs de services de cloud. Au cours de ce travail, nous avons expérimenté nos propositions à l'aide de CloudSim et Aglet. Plusieurs séries de simulation ont été menées, nous présentons quelques résultats et leurs interprétations et par la suite nous discuterons les résultats obtenus.

## IV.2 Outils et environnements de développement

L'objectif de ce travail consiste à satisfaire le client en termes de qualité de traitement en minimisant le coût de ressources à allouer et par conséquent réduire le paiement des clients. Afin de satisfaire ces objectifs nous avons élaboré un système d'allocation des ressources sur le cloud computing à la base de l'approche proposée dans le Chapitre III. Dans cette section, nous discutons l'architecture et modules du système implémenté. Nous citons également les outils, les méthodes et plateformes utilisées.

### IV.2.1 Outils et matériels de développement

Nous avons réalisé ce travail sur une machine avec les caractéristiques suivantes :

- **Microprocesseur** : Intel Core™ i7, Quad-Core 2.3 GHz.
- **Mémoire** vive : 8 Go.
- **Disque dur** : 1 To.
- **Système d'exploitation** : Windows 7 et Windows 10 de 64 bits.
- **Langage de programmation :** Java (JDK 1.8.0_144 ).
- **Environnement de développement** : NetBeans IDE 8.2, est un environnement de développement intégré, lancé en open source par Sun depuis juin 2000. Il Permettant potentiellement de créer des projets de développement mettant en œuvre de nombreux langages de programmation tels que, Java, C, le C++, le JavaScript, le XML, PHP et HTML, ou d'autres par l'ajout de greffons (NetBeans plugins).

### IV.2.2 Principaux modules implémentés et leurs plateformes utilisées

L'architecture de notre système implémenté est montrée dans la Figure IV.1. Dans les sections (**IV.2.2.1**) et (**IV.2.2.2**), nous détaillons les principaux modules du système et citons leurs méthodes et technologies utilisées dans leur implémentation.

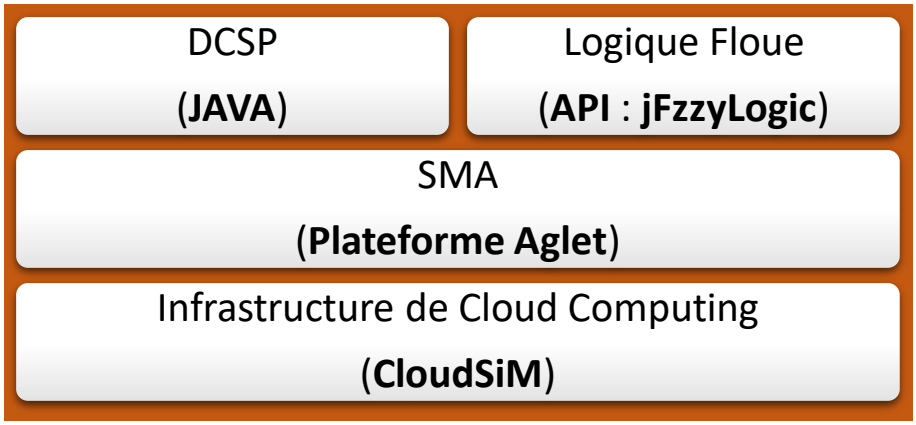



**Figure IV. 1 Principaux modules du système implémenté**

### IV.2.3 Module d'Infrastructure de Cloud Computing

Nous avons utilisé la plateforme **CloudSim** [113] dans l'implémentation du module d'Infrastructure de cloud computing. Cette plateforme permet la simulation orientée événements d'une infrastructure de type cloud. Principalement elle est destinée à la recherche en matière de conception et d'évaluation de l'architecture sous-jacente des plateformes de services informatiques à la demande. Elle permet d'obtenir un modèle du comportement du datacenter dans son ensemble. Par exemple, des travaux menés avec CloudSim par Beloglazov et Buyya ont permis d'analyser l'influence de différentes stratégies d'allocation de ressources sur la qualité de service rendue aux utilisateurs et la consommation électrique des datacenters. La Figure IV.2 représente les package CloudSim (version CloudSim 3.0.3) utilisé dans notre travail.

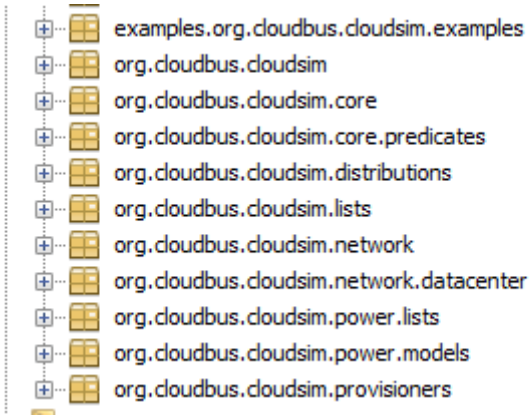



**Figure IV. 2 Package de cloud Sim**

### IV.2.4 Module SMA

On a implémenté les agents et leur environnement à l'aide de la plateforme d'agent mobile **Aglet**. Depuis 1996, Big Blue (IBM) met à disposition "**Aglets Workbench**" qui regroupe une librairie sous forme d'un package Java, ainsi qu'un serveur d'aglets (**Conteneur Tahiti**). "Aglets Workbench" sera renommé en 1998, ASDK (Aglets Software Développement Kit). ASDK est destiné à l'implémentation des systèmes multi-agent. Le terme *Aglets* vient de mots

« Agents Applets », ils sont des objets Java mobiles qui peuvent se déplacer d'un hôte à une autre. Ainsi, on peut interrompre un Aglet qui s'exécute sur une machine, le déporter vers un hôte distant et continuer cette exécution. Dans notre travail nous avons utilisé la version Aglets-2.0.2.

La Figure IV.3 montre un exemple de containeur Aglet pour un datacenter contenant trois hôtes, elle comporte les agents **DCA**, **HA_1**, **HA_2** et l'agent **HA_3** (dans l'état **OFF**).

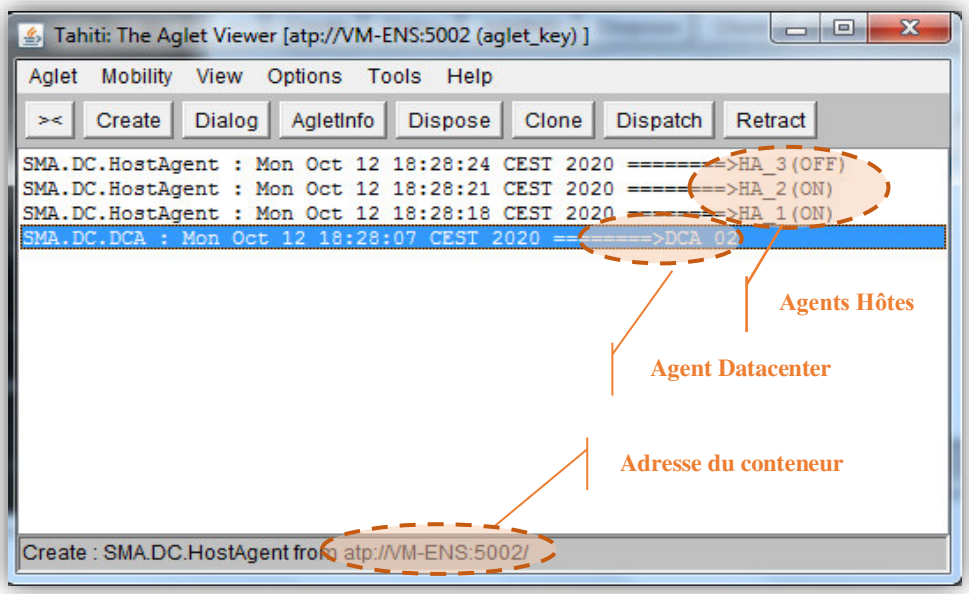



**Figure IV. 3 Simulation de l'environnement cloud computing**

### IV.2.5 Module DCSP

On a créé la classe **CONSTRAINTS.java** pour faciliter l'intégration de ce module dans l'environnement des agents Aglet. Les méthodes de cette classe (Figure IV.4) permettent aux agents à vérifier les contraintes DCSP pendant leur exécution.

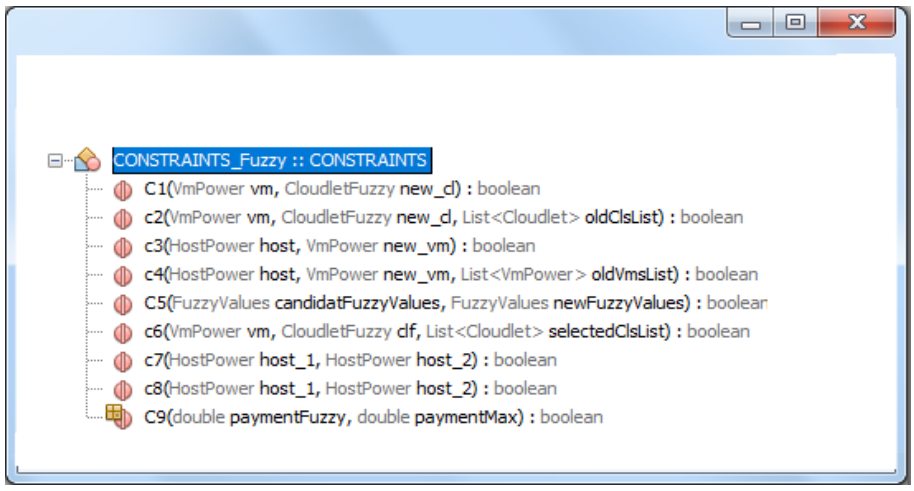



**Figure IV. 4 Simulation de l'environnement cloud computing**

### IV.2.6 Module de la Logique Floue

On a utilisé l'API **jFuzzyLogic** [114] dans l'implémentation de ce module. **jFuzzyLogic** offre une bibliothèque open source en Java pour implémenter des systèmes flous et permet de concevoir des contrôleurs en logique floue prenant en charge le standard de programmation de contrôle flou. La Figure IV.5 illustre le contenu du package cette bibliothèque.

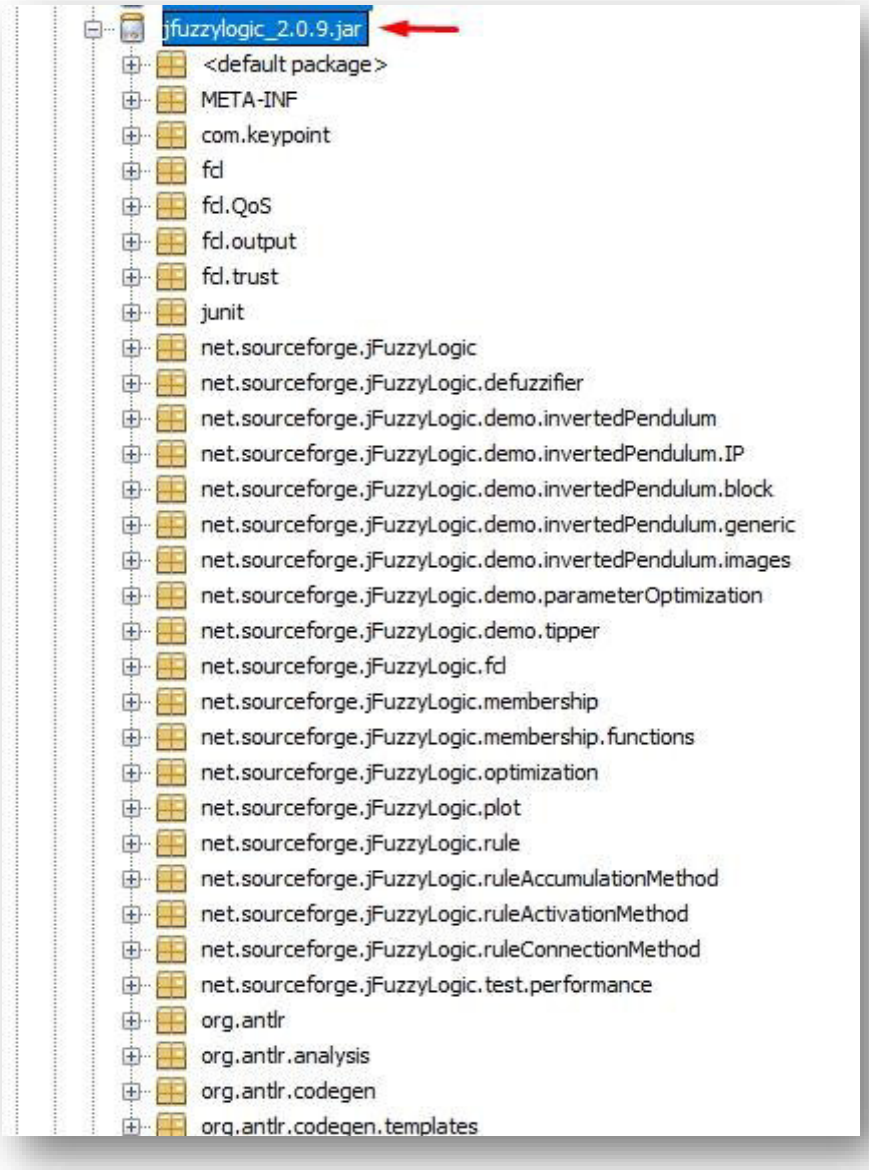



**Figure IV. 5 Package de la bibliothèque jFuzzyLogic**

Nous implémentons la classe **FuzzyModule.java** (Figure IV.6), qui adapte la bibliothèque jFuzzyLogic et offre une interface souple aux agents du système afin de leur permettre à évaluer leurs solutions via l'appel à la méthode **FuzzyModule.getScore()**. Cette méthode accepte comme paramètres d'entrées les valeurs des variables floues ***ram***, ***mips*** et ***payement***, et retourne la valeur de la variable floue ***Score***.

```java
package Fuzzy;

import net.sourceforge.jFuzzyLogic.FIS;
import net.sourceforge.jFuzzyLogic.rule.Variable;

public class FuzzyModule  {

    static String FCL_file = "D:\\fcl\\RA_definition.fcl";

    static double getScore(double ram,double mips,double payment) {

        // Load FIS form an FCL file
        String fcl = Gpr.readFile(FCL_file);
        FIS fis = FIS.createFromString(fcl, true);

        // Load input fuzzy variables
        fis.getVariable("ram").setValue(ram);
        fis.getVariable("mips").setValue(mips);
        fis.getVariable("payment").setValue(payment);

        // Evaluate and calculate the output fuzzy variables
        fis.evaluate();

        // return the value of the score
        return fis.getVariable("score").getValue();
    }
}
```

**Figure IV. 6 Pseudo code de la classe FuzzyModule.java**

## IV.3 Paramétrage de la simulation

Nous avons rassemblé les paramètres des expérimentations dans ce travail sous deux groupes selon les modules concernées (Module Cloud et Module Logique Floue), comme il est décrit dans Tableau IV.1 et Tableau IV.2.

### IV.3.1 Paramètres du module CloudSim

Nous définissons les paramètres de ce module comme décrit dans le Tableau IV.1.

**Tableau IV. 1 Paramètres du module CloudSim**

| Paramètres | Valeurs |
|---|---|
| Longueur maximale de cloudlet | 50 |
| Nombre total de cloudlets | 500–3000 |
| Nombre total de VM | 530 |
| Mémoires (RAM) des Machines Virtuelles | 100–1000 |
| Nombre de demandes de CPUs | 500–1500 |
| Nombre de datacenters | 3 |
| Nombre des hôtes | 47 |

### IV.3.2 Paramètres du module Logique Floue

Pour ce module, nous avons utilisé deux méthodes (COG et MOM) et les 48 règles floues y sont appliquées pour calculer le Score. Ces règles sont basées sur 4 variables. Comme l'illustre le Tableau IV.2, le paiement, le Ram et le Mips sont considérés comme des variables floues d'entrée, tandis que le score est considéré comme une variable de sortie.

**Tableau IV. 2 Paramètres du module Logique Floue**

| Variables floues | Type de variable | Terme linguistique | Intervalle flou |
|---|---|---|---|
| **Paiement** | Entrée | Cheap | [0.1,0.1,5] |
| | | Medium | [3,9,18] |
| | | Expensive | [14,22,30] |
| **Ram** | Entrée | Low | [0,0,5] |
| | | Fair | [3,8,10] |
| | | Medium | [7,15,20] |
| | | High | [17,25,30] |
| **Mips** | Entrée | Low | [0,0,8] |
| | | Fair | [5,11,16] |
| | | good | [12,19,28] |
| | | very_good | [24,30,39] |
| **Score Variables floues** | Sortie Type de variable | Low | [0,0,0.3] |
| | | Fair | [0.2, 0.35, 0.5] |
| | | good | [0.4, 0.6, 0.8] |
| | | very_good | [0.7, 0.85, 1] |

## IV.4 Déroulement et Résultats de simulation

Dans cette section, nous visons à clarifier la faisabilité de notre proposition en citant des exemples de déroulement d'exécution et discutant les résultats de simulations obtenus.

### IV.4.1 Déroulement d'exécution

Nous discutons le déroulement d'exécution de deux exemples illustratifs dans les niveaux du module de cloud computing et le module de la logique floue.

### IV.4.2 Exemple illustratif d'exécution au niveau des modules Cloud et SMA

Nous proposons un exemple illustratif pour expliquer le déroulement de traitement d'un ensemble de demandes des clients (cloudlets) et les étapes d'élaboration de leurs solutions selon les algorithmes d'allocation de ressources dans notre approche proposée.

➢ **Etape 0 : (Configuration du système)**

On suppose qui un système de cloud a une infrastructure qui comporte la configuration des datacenters suivante :

a) **Datacenter (DCA_0) :** avec trois hôtes **: HA_1, HA_2, HA_3.**
b) **Datacenter (DCA_1) :** avec les hôtes **: HA_1, HA_2, HA_3.**
c) **Datacenter (DCA_1) :** ses hôtes sont **: HA_1, HA_2, HA_3.**

➢ **Etape 1 (Diffusion de demandes des Clients ) :**

La Figure IV.7 montre le journal de l'agent BA qui diffuse une requête contenant une liste de dix cloudlets (CL9, ……, CL18) à soumettre aux agents DCA_0, DCA_1 et DCA_2.

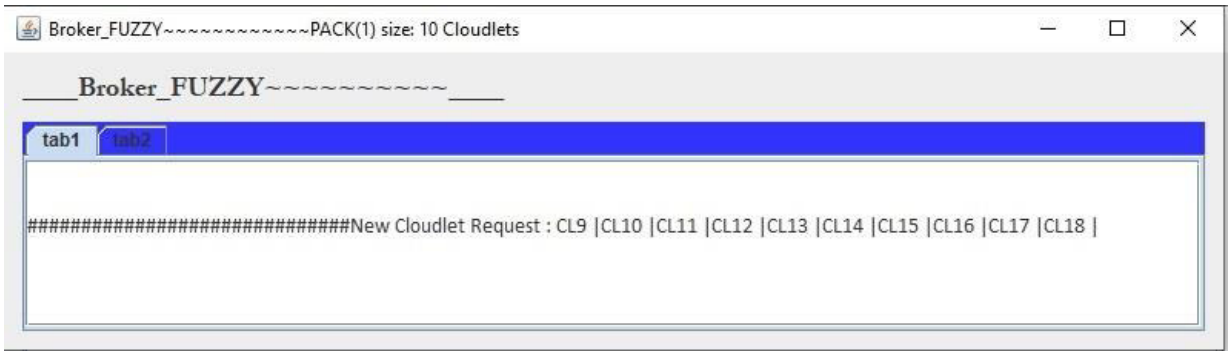



**Figure IV. 7 Journal de l'agent BA dans l'étape 1**

➢ **Etape 2: (Elaboration des solutions locales)**

Les figures Figure IV.8, Figure IV.9 et Figure IV.10 montrent les interfaces graphiques des journaux pour les agents DCA_0, DCA_1 et DCA_2, chaque agent DCA reçoit la liste des cloudlets par l'agent BA, et il la transmet à ses agents HA du même datacenter.

Chaque agent HA essaye de trouver une solution pour chaque cloudlet dans la liste en exécutant l'algorithme d'allocation de ressources au niveau local. A la fin, les HA négocient et élaborent une solution locale et l'envoi à leur agent DCA.

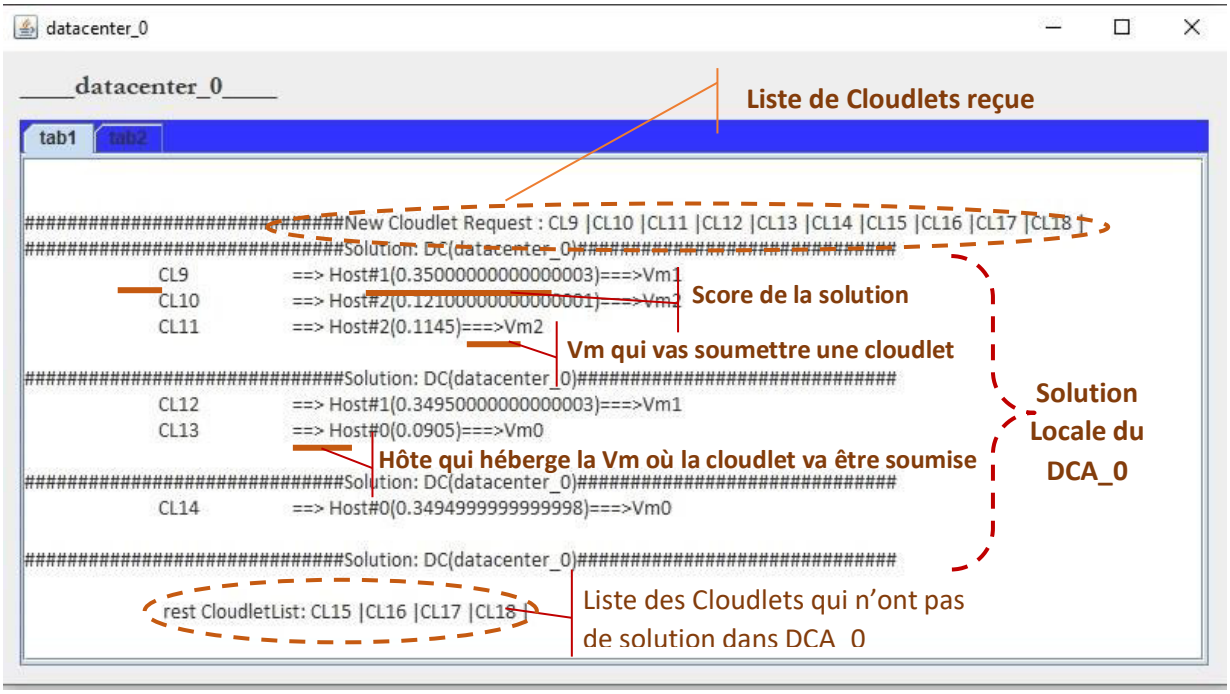



**Figure IV. 8 Journal de l'agent DCA_0**

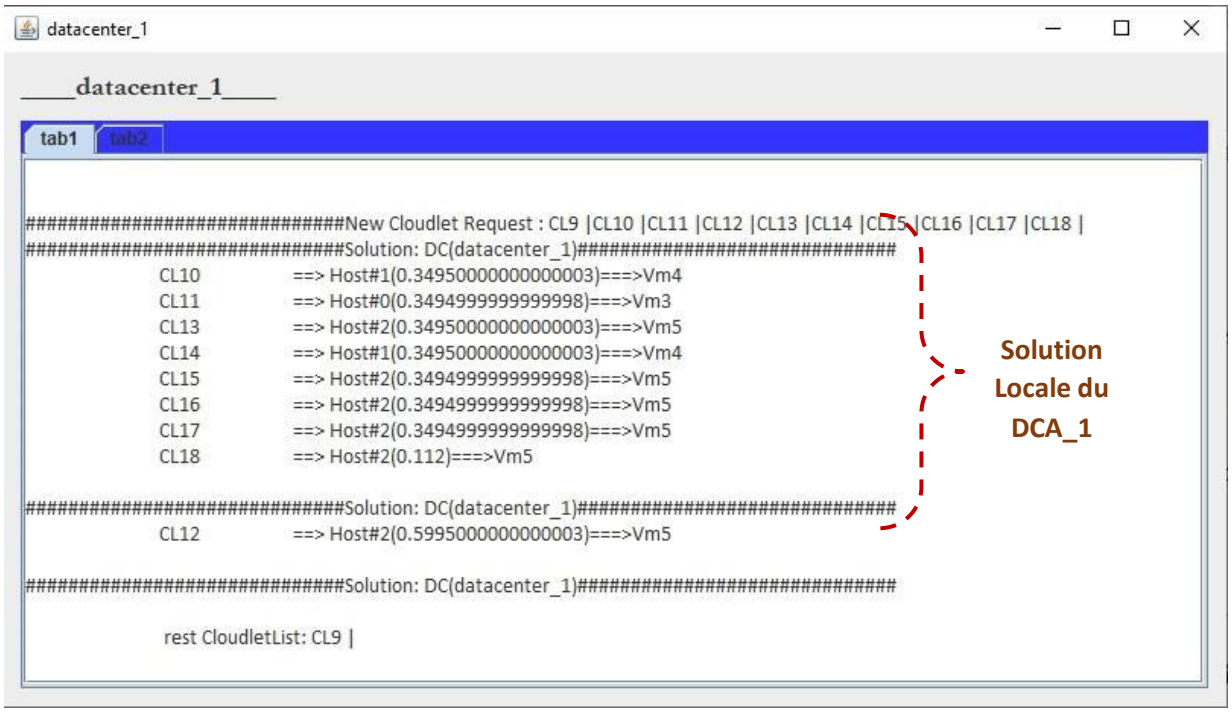



**Figure IV. 9 Journal de l'agent DCA_1**

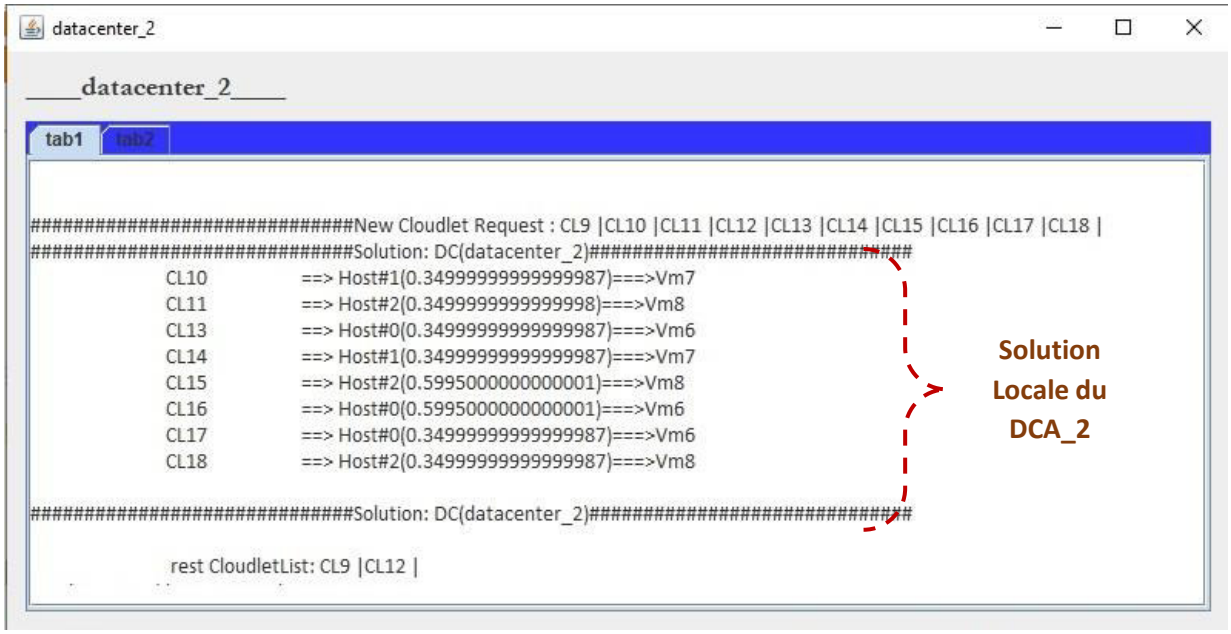



**Figure IV. 10 Journal de l'agent DCA_2**

➢ **Etape 3: (Elaboration de la solutions globale)**

Après avoir élaboré les solutions locales pour chaque cloudlet, les agents DCA négocient entre eux afin de tirer la meilleure solution pour chaque cloudlet et en suite l'envoyé à leur agent DCA. La Figure IV.11 illustre dans le journal de l'agent BA la solution globale pour chaque cloudlet demandé dans la liste.

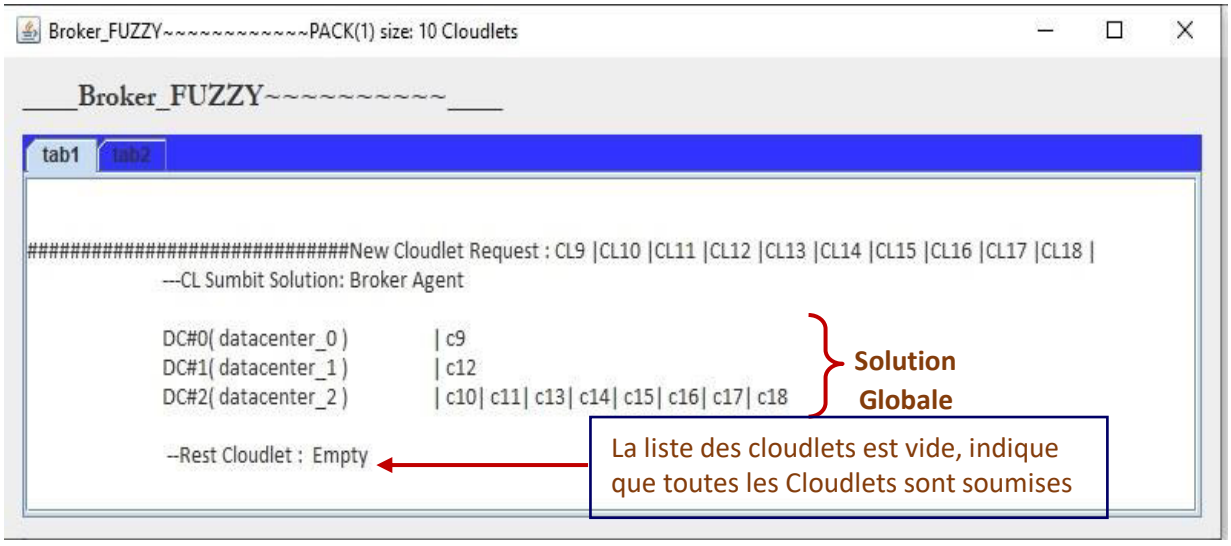



**Figure IV. 11 Journal de l'agent DCA_2**

### IV.4.3 Exemple illustratif d'exécution au niveau du module logique floue

L'exemple discuté dans cette section permet d'illustrer le déroulement d'étapes d'exécution dans le module de la Logique Floue. La Figure IV.12 montre l'objectif et la relation de ce module avec le module du cloud. Les Agents Hôtes trouvent des solutions pour soumissionner

une cloudlet, puit ils évaluent leurs solutions en calculant les scores à l'aide du module de la Logique Floue, et à la fin, ils tirent la meilleure solution qui possède le meilleur score (max).

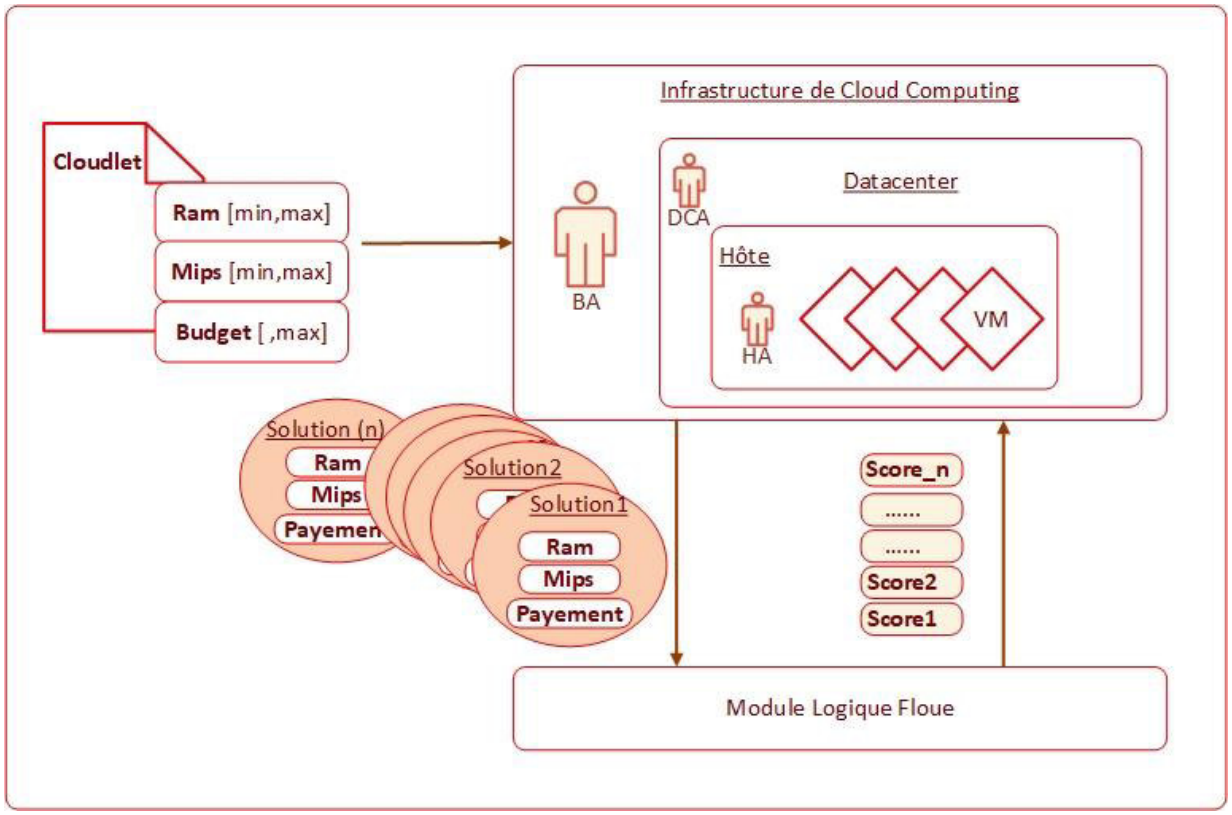



**Figure IV. 12 La relation entre le module de la Logique Floue avec le module de Cloud**

Nous proposons un exemple illustratif pour expliquer le déroulement de fonctionnement du Module de La logique Floue dans le traitement d'une demande de calcul de scores de huit solutions proposées pour une cloudlet client (Tableau IV.3).

**Tableau IV. 3 Cloudlet demandée dans l'exemple illustratif**

| | | |
|---|---|---|
| userID | | 1 |
| ClouletId | | 1 |
| Ram () | minRam | 4 |
| | maxRam | 30 |
| Mips | minMips | 4 |
| | maxMips | 25 |
| Bw | | 999 |
| storage | | 1024 |
| Length | | 20 |
| Budget | max paiement | 28 |

On suppose que nous avons le scénario de déroulement qui est expliqué dans les étapes suivantes :

**Étape 1:** les agents **HA** proposent leurs solutions dans le Tableau IV.4. Ces solutions doivent répondre aux exigences du client.

**Tableau IV. 4 Solutions proposées par les agents HA**

| Solutions | Ram | Mips | paiement |
|---|---|---|---|
| **S1** | 8 | 14 | 4.5 |
| **S2** | 18 | 7 | 4.8 |
| **S3** | 4 | 4 | 2 |
| **S4** | 25 | 15 | 15 |
| **S5** | 22 | 30 | 25 |
| **S6** | 18 | 14 | 4.9 |
| **S7** | 9 | 35 | 14.5 |
| **S8** | 18 | 26 | 16 |

**Étape 2**: la représentation des variables du module de logique floue en deux groupes :

**a) Variables d'entrée**

-Ram = **low**, **fair**, **medium, high.**

-MIPS = **low**, **fair**, **good, very_good**.

-payement =**cheap**, **medium**, **expensive**.

**b) b) variables de sortie:**

-Score: **low, fair, good**, **very_good**.

Ces variables sont représentées par la méthode des nombres flous triangulaires comme le montre la Figure IV.13 (a, b, c et d).

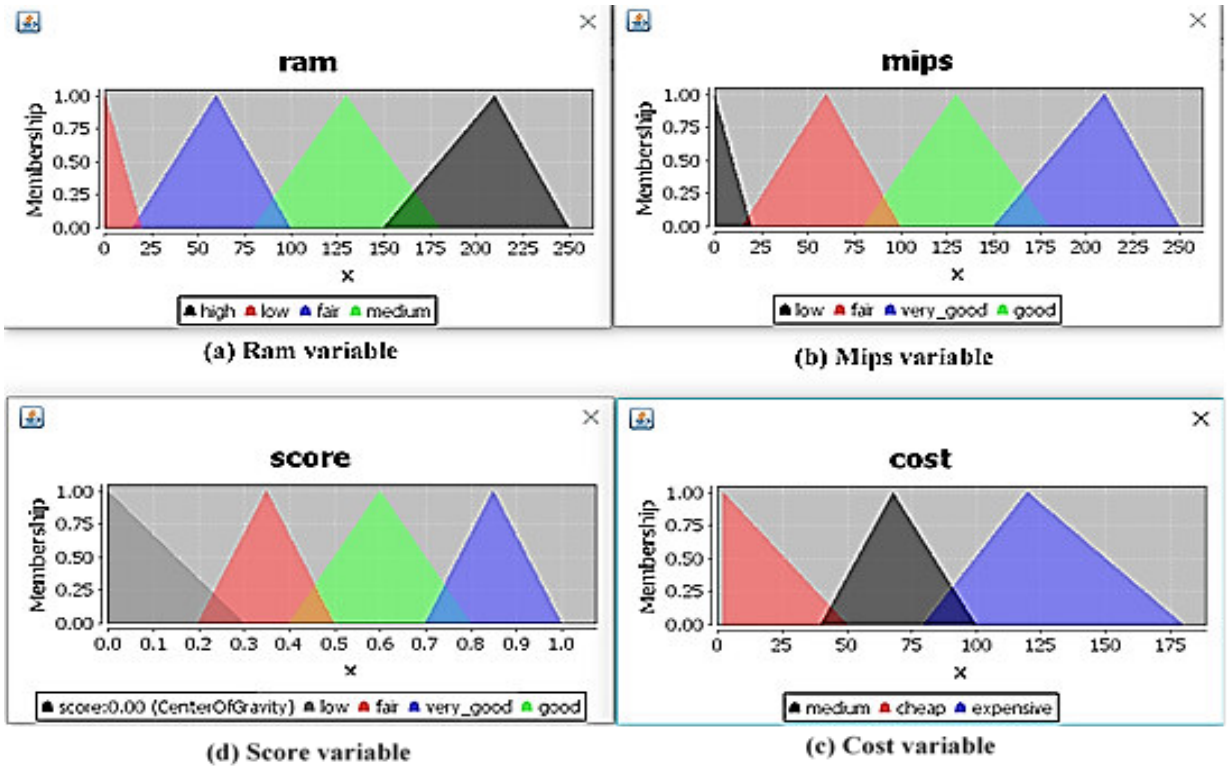



**Figure IV. 13 Variables d'entrée et de sortie de la Logique Floue**

**Étape 3** : (Fuzzification) la représentation en termes flous des solutions HA est illustrée dans le Tableau IV.5.

**Tableau IV. 5 Évaluations linguistiques des solutions**

| Solutions | Ram | Mips | paiement |
|---|---|---|---|
| **S1** | fair or medium | fair or good | cheap or medium |
| **S2** | medium or high | low or fair | cheap or medium |
| **S3** | low or fair | low | cheap |
| **S4** | High | fair or good | medium or expensive |
| **S5** | High | very_good | expensive |
| **S6** | medium or high | fair or good | cheap or medium |
| **S7** | fair or medium | very_good | medium or expensive |
| **S8** | medium or high | good or very_good | medium or expensive |

**Étape 4**: Après avoir défini les fonctions d'appartenance, nous suggérons d'utiliser un ensemble de règles floues (environ 48 règles) dans la défuzzification des valeurs floues pour obtenir les valeurs de score en utilisant deux méthodes: **GOG** (Tableau IV.6) et **MOM** (Tableau IV.7).

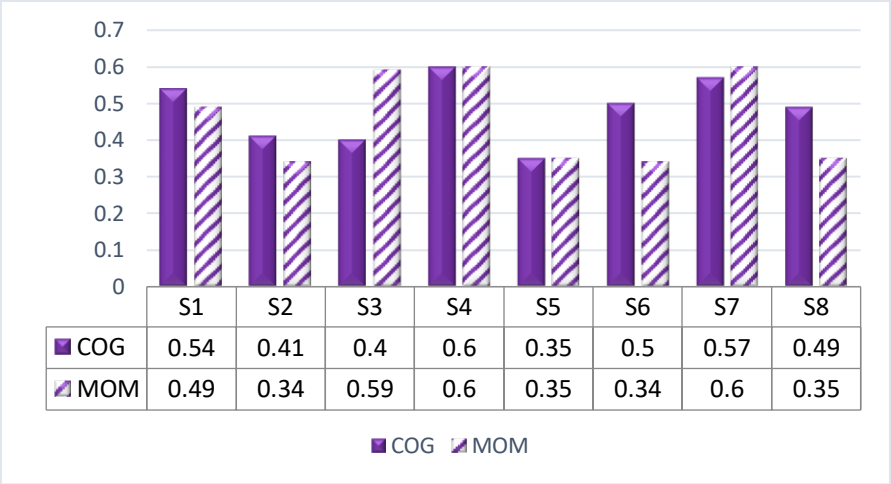
**Tableau IV. 6 Résultats obtenus avec la méthode COG**

| Solutions | Ram | Mips | paiement | score |
|---|---|---|---|---|
| **S1** | 8 | 14 | 4.5 | 0.54 |
| **S2** | 18 | 7 | 4.8 | 0.41 |
| **S3** | 4 | 4 | 2 | 0.40 |
| **S4** | 25 | 15 | 15 | **0.60** |
| **S5** | 22 | 30 | 25 | 0.35 |
| **S6** | 18 | 14 | 4.9 | 0.50 |
| **S7** | 9 | 35 | 14.5 | 0.57 |
| **S8** | 18 | 26 | 16 | 0.49 |

**Tableau IV. 7 Résultats obtenus avec la méthode MOM**

| Solutions | Ram | Mips | Paiement | score |
|---|---|---|---|---|
| **S1** | 8 | 14 | 4.5 | 0.49 |
| **S2** | 18 | 7 | 4.8 | 0.34 |
| **S3** | 4 | 4 | 2 | 0.59 |
| **S4** | 25 | 15 | 15 | 0.60 |
| **S5** | 22 | 30 | 25 | 0.35 |
| **S6** | 18 | 14 | 4.9 | 0.34 |
| **S7** | 9 | 35 | 14.5 | 0.60 |
| **S8** | 18 | 26 | 16 | 0.35 |

**Étape 5** Discussion : Comme il est illustré dans le Tableau IV.7, le meilleur score obtenu avec l'utilisation de la méthode COG est (0,60) ce qui correspond à (S4). Par conséquent, il sera retourné au client. De plus, les scores des deux méthodes sont très proches l'un de l'autre, comme le montre la Figure IV.14.




| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|
| COG | 0.54 | 0.41 | 0.4 | 0.6 | 0.35 | 0.5 | 0.57 | 0.49 |
| MOM | 0.49 | 0.34 | 0.59 | 0.6 | 0.35 | 0.34 | 0.6 | 0.35 |

**Figure IV. 14 Comparaison entre les méthodes COG et MOM**

### IV.4.4 Métriques de performance utilisées

Dans cette étude nous intéressons à quatre métriques dans le processus d'allocation des ressources dans le cloud computing qui sont :

### IV.4.5 Temps d'exécution Moyen (TAvg)

$$T_{Avg} = \frac{\sum_i^n T_i}{n} \qquad (38)$$

Où :

$\mathbf{T_i}$ est le temps d'exécution d'une cloudlet $\boldsymbol{i}$.

$\boldsymbol{n}$ est le nombre total de cloudlets dans le système .

### IV.4.6 Charge Moyenne (LAvg)

$$L_{Avg} = \frac{\sum_d^m \sum_h^k L_h}{m*k} * 100 \qquad (39)$$

Où,

$\boldsymbol{L_h}$ charge de l'hôte $\boldsymbol{h}$ dans le datacenter $\boldsymbol{d}$.

$\boldsymbol{k}$ nombre des hôtes dans le datacenter $\boldsymbol{d}$.

$\boldsymbol{m}$ nombre des datacenters,

### IV.4.7 Coût global de consommation d'énergie (C)

$$C = \sum_d^m \sum_h^k Cost_h \qquad (40)$$

Où

$\boldsymbol{Cost_h}$ Coût de consommation d'énergie de l'hôte $\boldsymbol{h}$ dans datacenter $\boldsymbol{d}$.

### IV.4.8 Taux moyen de gain de paiement des clients (G)

$$G = \frac{\sum_d^m \sum_h^k \sum_c^w G_c}{n} \quad (41)$$

$$G_c = \frac{B_c - P_c}{B_c} \quad (42)$$

$$P_c = \sum_r^s U_r * Price_r * t_c \quad (43)$$

Où,

$t_c$ le temps d'exécution d'une cloudlet $c$.

$Price_r$ Prix défini pour le type de ressource $r$ par l'hôte $h$.

$U_r$ Unités requises d'une ressource demandée $r$.

$s$ Nombre de ressources demandées par la cloudlet $c$.

$r$ Type de ressource demandée (Mips, RAM, Storage, …) par la cloudlet $c$.

$P_c$ Valeur calculée du paiement d'une cloudlet $c$.

$B_c$ Budget de la cloudlet $c$. (défini par le client)

$G_c$ Gain dans le payement d'une cloudlet $c$.

$c$ Cloudlet soumise dans (affectée à) le hôte $h$

$w$ nombre de cloudlets soumises dans le hôte $h$.

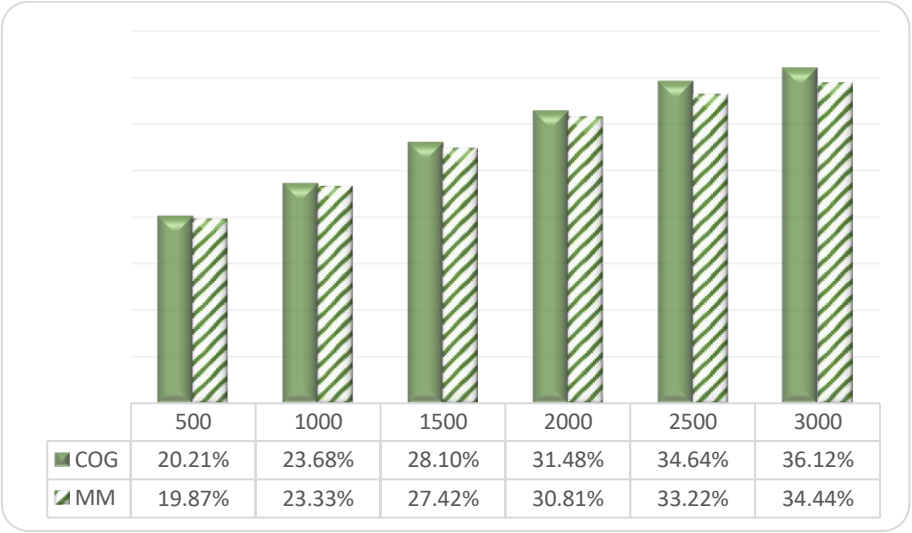
### IV.4.9 Résultats de simulations

L'implémentation se compose de deux modules. Le premier module est la simulation de l'environnement de cloud qui inclut le système SMA et les contraintes DCSP. Le deuxième module est Fuzzy Logic qui est utilisé pour déterminer les scores des solutions arrivées du premier module.

Les résultats de l'évaluation expérimentale sont obtenus à partir de 06 paquets de cloudlets. Ainsi, chaque paquet a un nombre différent de cloudlets tels que : le premier paquet comporte 300 cloudlets et le dernier paquet inclut 3000 cloudlets. Ces paquets ont été examinés sur plusieurs paramètres de performance (TAvg, LAvg, C et G) avec l'utilisation de deux méthodes de défuzzification : Center of Gravity (COG) et Mean of Maxima (MM). Les résultats obtenus des paramètres définis sont discutés dans les points suivants (a, b, c, d) et illustrés dans leurs figures correspondantes (Figure IV.15, Figure IV.16, Figure IV.17 et Figure IV.18).

a) *Charge moyenne, (L_Avg)* est une métrique importante car elle représente la consommation d'énergie. Par conséquent, notre approche vise à réduire la consommation d'énergie dans les datacenters (voir Figure IV. 15). Pendant le temps expérimental, la charge est réduite et sa valeur se situe entre 20% et 37%. En d'autres termes, notre approche fournit un bon équilibrage de charge entre les hôtes dans les différents

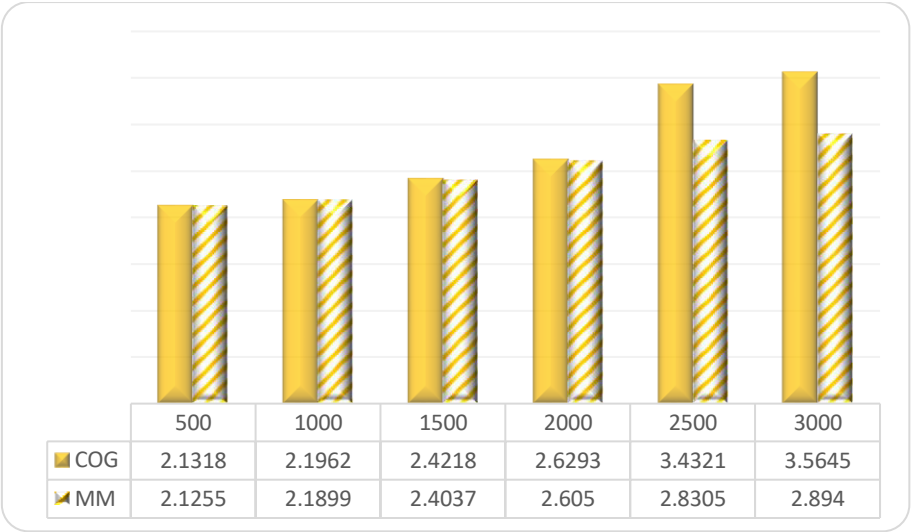
datacenters. Cela est dû à l'efficacité de son architecture qui est basée sur des agents distribués et leurs contraintes DCSP. De plus, quand on a le nombre de cloudlets augmente, la méthode **MM** est meilleure que la méthode **COG** en termes de vitesse d'augmentation de la charge, comme le montre la (Figure IV.15).



| | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
|---|---|---|---|---|---|---|
| COG | 20.21% | 23.68% | 28.10% | 31.48% | 34.64% | 36.12% |
| MM | 19.87% | 23.33% | 27.42% | 30.81% | 33.22% | 34.44% |

**Figure IV. 15 Charge Moyenne**

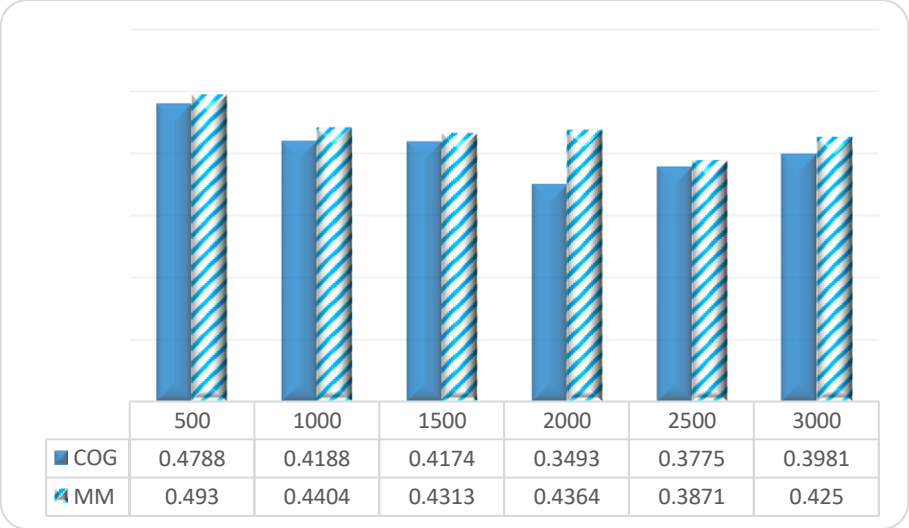
b) *Coût global de la consommation d'énergie (C),* il a un rôle déterminant dans l'algorithme d'hébergement de VM dans les différents hôtes. Cependant, les hôtes élus sont ceux qui ont le meilleur rapport qualité-prix. Dans cette étude, l'optimisation du coût global est directement liée à la réduction de la consommation d'énergie. Cela montre la rentabilité de la conservation de l'équilibrage de charge, comme il est montré dans les résultats de (Figure IV.16). Dans la figure IV. 16 nous observons que les résultats des deux méthodes sont différents dans les deux derniers paquets. Ceci en raison de la différence de charge moyenne dans ces packages. On constate que la méthode MM est meilleure que la méthode COG en terme de calcul des coûts.



| | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
|---|---|---|---|---|---|---|
| COG | 2.1318 | 2.1962 | 2.4218 | 2.6293 | 3.4321 | 3.5645 |
| MM | 2.1255 | 2.1899 | 2.4037 | 2.605 | 2.8305 | 2.894 |

**Figure IV. 16 Coût global de la consommation d'énergie**

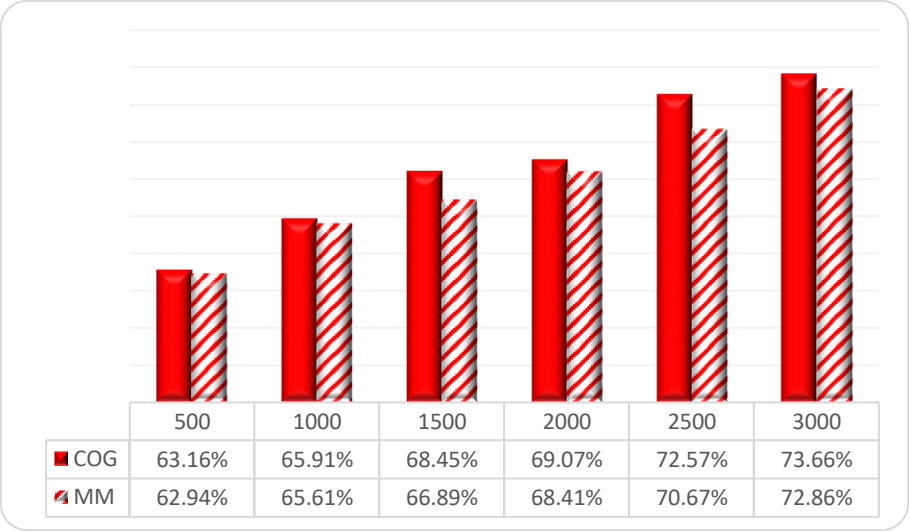
c) *Temps d'exécution Moyenne (TAvg),* il représente la moyenne des temps d'exécution des cloudlets par paquet. Le temps d'exécution d'un cloudlet soumis dans une VM est lié à la charge de travail (work load) de cette VM. Par conséquent, le temps d'exécution moyen de chaque paquet augmente. La disparité (haute et basse) dans les valeurs TAvg dans les paquets est causée par l'augmentation du nombre de cloudlets comme indiqué dans la (Figure IV.17). Les valeurs faibles de TAvg apparaissent lorsque les nouveaux cloudlets sont soumises dans de nouvelles VM vides. Cependant, les valeurs élevées apparaissent lorsque les nouveaux cloudlets sont soumis dans des VM anciennes (sont déjà hébergées). De plus, la technique COG est meilleure que MM en aspect de temps d'exécution moyen des cloudlets.



| | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
|---|---|---|---|---|---|---|
| COG | 0.4788 | 0.4188 | 0.4174 | 0.3493 | 0.3775 | 0.3981 |
| MM | 0.493 | 0.4404 | 0.4313 | 0.4364 | 0.3871 | 0.425 |

**Figure IV. 17 Temps d'exécution Moyenne (TAvg)**

*d) Taux de gain de paiement (G),* il représente le taux de gain total dans les paiements des cloudlets par le total de leurs budgets. La valeur du paiement de cloudlet est un facteur déterminant dans le calcul du score à l'intérieur du module flou, cette valeur est représentée sur des intervalles. En d'autres termes, l'importance de ce facteur réside dans le fait d'éviter les solutions coûteuses. Donc il améliore le gain de paiement comme le montre Figure IV.18. On observe également que la valeur de G atteint (73%). Cela reflète l'efficacité de l'utilisation de la logique floue dans notre approche. La disparité des valeurs de G est causée par les valeurs élevées des ressources demandées par les clients dans les cloudlets et par l'augmentation des prix des ressources dans les hôtes. De plus, la technique COG est meilleure que MM en termes de gain de paiement des clients.



| | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
|---|---|---|---|---|---|---|
| COG | 63.16% | 65.91% | 68.45% | 69.07% | 72.57% | 73.66% |
| MM | 62.94% | 65.61% | 66.89% | 68.41% | 70.67% | 72.86% |

**Figure IV. 18 Taux de gain de paiement**

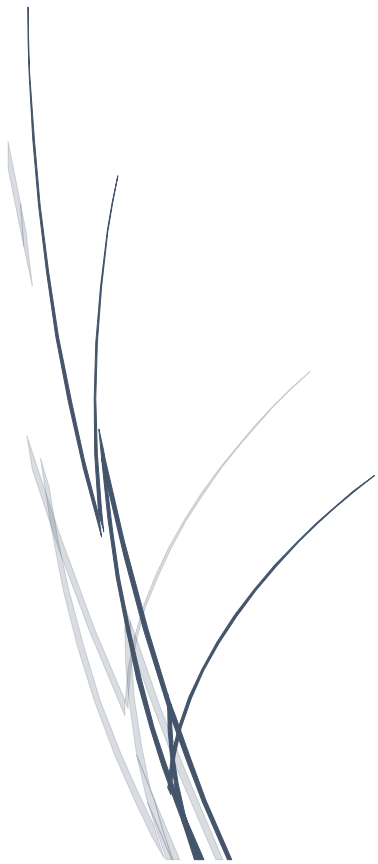
## IV.5 Conclusion

Dans ce chapitre, nous avons essayé de mettre en œuvre l'ensemble des idées qui caractérise l'approche proposée en se concentrant sur l'implémentation de système multi agents ainsi que l'intégration entre différents agents.

Notre architecture est bien implémentée en utilisant un environnement de développement JAVA qui comporte le simulateur CloudSim pour simuler l'environnement cloud computing et la plateforme Jade qui permet de visualiser l'interaction entre l'ensemble des agents implémentés.

Les résultats obtenus à partir de notre simulation, nous confirment que l'utilisation d'un système multi-agents pour l'allocation des ressources dans le cloud computing assure la satisfaction du client qui consiste à diminuer le cout des ressources consommées afin d'essayer d'utiliser le maximum de service a un cout minimal et équilibrer la charge dans les datacenters du cloud.

# Conclusion Générale

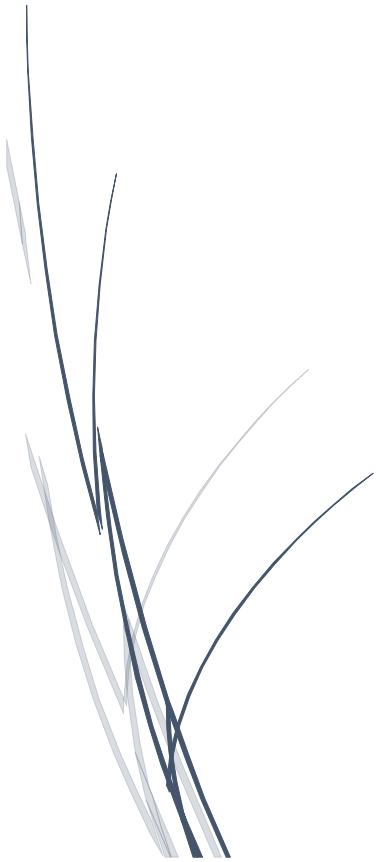


Le gaspillage de ressources et les factures astronomiques dont souffrent les cloud computing actuels, et dont ils continueront d'en souffrir à l'avenir, appelle des techniques sophistiquées pour l'allocation des ressources. Des travaux de recherches nombreux, ont examinés la question et ont débouchés sur des approches multiples. Il semble qu'aucune approche ne suffit, à elle seule, pour relever tous les défis qui se posent aux concepteurs, tant le problème est complexe et ses aspects divers. Nous sommes arrivés à conclure que l'implication de plusieurs méthodes spécialisées est une bonne piste, pourvu qu'il y ait des mécanismes efficaces qui permettent l'intégration et la coopération entre ces composants dans un système distribué comme le cloud computing.

Le travail qui vient d'être présenté tente d'envisager une telle approche. Nous y avons discuté une approche de redistribution le processus d'allocation des ressources dans des dataceners hétérogènes avec la mise en œuvre une politique d'allocation des ressources distribuée, conséquence de la diversité virtuelle et temporelle, des taux d'utilisation de ces ressources dans les divers datacenters du cloud. Nous avons aussi exploité l'hétérogénéité de ces centres et les chevauchements dans l'allocation des ressources virtuelles pour dresser une sorte d'un système multi agents, utilisé en tant que système de gestion des entités réparties (hôtes physiques et machines virtuelles). Nous pensons que notre approche doit être analysé en tant que point d'articulation des divers composants impliqués, notamment la logique floue. Dans la conception de cette approche elle-même nous avons (à dessein) choisi un fonctionnement distribué et parallèle, où les entités impliquées (agents DCA et HA) peuvent se contenter d'un minimum de connaissances mutuelles. Nous pensons avoir pris le parti de la simplicité, tout comme nous garantissons l'équité de la distribution et sa réalisation d'un équilibre de charge selon les expérimentations.

Une étude préliminaire nous laisse penser que notre approche est adaptée à la gestion des demandes floues des ressources et la prise en charge de certains critères de QoS. Cette question pourrait faire l'objet de travaux de recherches futurs en vue de proposer une architecture complète d'un système d'allocation des ressources mettant en œuvre cette approche. Des questions connexes pourraient aussi faire l'objet de recherches plus poussées telles que la faisabilité de l'approche de prévision proposé, le fonctionnement dans les conditions de fortes charges (contrôle de congestion), et la prise en charge de la QoS.

# Bibliographie



[1] E. I. DJEBBAR, “Optimisation D’Ordonnancement Et D’Allocation De Ressources Dans Les Cloud Computing,” Oran, 2016.
[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds,” *ACM SIGCOMM Comput. Commun. Rev.*, 2008, doi: 10.1145/1496091.1496100.
[3] G. Reese, *Cloud Application Architectures*, 1st ed. O’Reilly Media, 2009.
[4] P. Mell and T. Grance, “The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology,” in *Public Cloud Computing: Security and Privacy Guidelines*, 2012, pp. 97–101.
[5] B. Wiggins, “Cloud Computing and SOA Convergence in Your Enterprise—A Step-by-Step Guide,” *Int. J. Inf. Manage.*, vol. 30, no. 5, pp. 470–471, 2010, doi: 10.1016/j.ijinfomgt.2010.06.002.
[6] B. Abdelbasset, “Composition de service web dans le cloud computing,” Mohamed KHIDER - BISKRA Faculté, 2018.
[7] “Iinfrastrcuter as a service. - Wikipedia.” https://en.wikipedia.org/wiki/Iinfrastrcuter_as_a_service. (accessed Aug. 26, 2021).
[8] N. M. Lucas, V. Kherbache, M. Mohamed, K. Yannick, and L. Allan, “Cloud Computing ;Licence Professionnelle : Administration de systèmes, réseaux et applications à base de logiciels libres,” *IUT Charlemagne*. IUT Nancy Charlemagne, p. 39.
[9] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010, doi: 10.1007/s13174-010-0007-6.
[10] Xenserver, “Citrix Hypervisor | Open Source Server Virtualization,” 2019. https://xenserver.org/ (accessed Aug. 26, 2021).
[11] “KVM.” https://www.linux-kvm.org/page/Main_Page (accessed Aug. 26, 2021).
[12] “What is ESXI? | Bare Metal Hypervisor | ESX | VMware.” Accessed: Aug. 26, 2021. [Online]. Available: https://www.vmware.com/products/esxi-and-esx.html.
[13] H. Geng, *Data Center Handbook*. 2014.
[14] Cisco, “Data Center Architecture Overview - Cisco,” *cisco.com*, 2008. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCInfra_1.html (accessed Aug. 26, 2021).
[15] D. DAD, “OPTIMISATION DES PERFORMANCES DES DATA CENTERS DES CLOUD SOUS CONTRAINTE D’ÉNERGIE CONSOMMÉE,” Université d’Oran 1, Ahmed Ben Bella, 2016.
[16] K. Bessai, “Gestion optimale de l’allocation des ressources pour l’execution des processus dans le cadre du Cloud,” Université Paris1 Panthéon-Sorbonne.Français., 2014.
[17] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, “Agent-based load balancing in Cloud data centers,” *Cluster Comput.*, 2015, doi: 10.1007/s10586-015-0460-x.
[18] M. Xu, W. Tian, and R. Buyya, “A survey on load balancing algorithms for virtual machines placement in cloud computing,” *Concurr. Comput. Pract. Exp.*, vol. 29, no. 12, p. e4123, Jun. 2017, doi: 10.1002/cpe.4123.
[19] S. M. Parikh, N. M. Patel, and H. B. Prajapati, “Resource Management in Cloud Computing: Classification and Taxonomy,” *arXiv Prepr. arXiv1703.00374*, Feb. 2017, [Online]. Available: http://arxiv.org/abs/1703.00374.
[20] M. R. HABES, “ALLOCATION DE RESSOURCES DANS LE CLOUD COMPUTING,” http://biblio.univ-annaba.dz/wp-content/uploads/2016/12/These-Habes-Mohamed-Raouf.pdf, 2016.
[21] D. Talia, “Clouds Meet Agents: Toward Intelligent Cloud Services,” *IEEE Internet Comput.*, vol. 16, no. 2, pp. 78–81, Mar. 2012, doi: 10.1109/MIC.2012.28.
[22] N. Krishnaveni, “Survey on Dynamic Resource Allocation Strategy in Cloud Computing Environment,” *Int. J. Comput. Appl. Technol. Res.*, vol. 2, no. 6, pp. 731–732, Nov. 2013, doi: 10.7753/IJCATR0206.1019.
[23] N. Asha and G. Rao, “A Review on Various Resource Allocation Strategies in Cloud Computing,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 7, pp. 177–183, 2013.
[24] P. Ronak and S. Patel, “Survey on Resource Allocation Strategies in Cloud Computing,” *Int. J. Eng. Res. Technol.*, vol. 2, no. 2, pp. 1–5, 2013.
[25] S. L. V.Vinothina, D. R.Sridaran, and G. Padmavathi, “A survey on resource allocation strategies in cloud,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 3, no. 6, pp. 97–104, 2012, doi:

10.1504/IJRIS.2018.096229.

[26] R. Shimonski, *Windows Server 2003 Clustering & Load Balancing*, 1st ed. McGraw-Hill Osborne Media, 2003.

[27] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, "Availability and Load Balancing in Cloud Computing," in *International Conference on Computer and Software Modeling*, 2011, vol. 14, pp. 134–140.

[28] V. K. Prasad, A. Nair, and S. Tanwar, "Resource Allocation in Cloud Computing," in *Instant Guide to Cloud Computing*, no. August, BPB Publications, 2019, pp. 343–376.

[29] H. Zhou, S. Deng, H. Huang, and Y. Wu, "Resource allocation in cloud computing based on clustering method," in *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, Apr. 2015, pp. 489–494, doi: 10.1109/SYSCON.2015.7116799.

[30] Y. O. Yazir *et al.*, "Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis," in *2010 IEEE 3rd International Conference on Cloud Computing*, Jul. 2010, pp. 91–98, doi: 10.1109/CLOUD.2010.66.

[31] A. Kochut, K. Beaty, H. Shaikh, and D. G. Shea, "Desktop workload study with implications for desktop cloud resource optimization," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, Apr. 2010, pp. 1–8, doi: 10.1109/IPDPSW.2010.5470723.

[32] D. Kumar and A. S. Singh, "A survey on resource allocation techniques in cloud computing," in *International Conference on Computing, Communication & Automation*, May 2015, pp. 655–660, doi: 10.1109/CCAA.2015.7148454.

[33] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009, doi: 10.1016/j.future.2008.12.001.

[34] V. Awasare and S. Deshmukh, "Survey and Comparative Study on Resource Allocation Strategies in Cloud Computing Environment," *IOSR J. Comput. Eng.*, vol. 16, no. 2, pp. 94–101, 2014, doi: 10.9790/0661-162194101.

[35] G. E. Gonçalves *et al.*, "Resource allocation in clouds: concepts, tools and research challenges," in *Minicursos do SBRC 2011*, SBC, 2011, pp. 197–240.

[36] F. I. Popovici and J. Wilkes, "Profitable services in an uncertain world," in *ACM/IEEE SC 2005 Conference (SC'05)*, 2005, pp. 36–36, doi: 10.1109/SC.2005.58.

[37] L. Wu, S. K. Garg, and R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2011, pp. 195–204, doi: 10.1109/CCGrid.2011.51.

[38] K. S. Guruprakash and S. Siva Sathya, "Log based automated SMI parameter identification and resource recommendations in cloud," *Indian J. Sci. Technol.*, vol. 9, no. 30, pp. 1–7, 2016, doi: 10.17485/ijst/2016/v9i30/99011.

[39] Y. Sun, J. White, S. Eade, and D. C. Schmidt, "ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization," *J. Syst. Softw.*, vol. 116, pp. 146–161, Jun. 2016, doi: 10.1016/j.jss.2015.08.006.

[40] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Apr. 2012, doi: 10.1109/TSC.2011.7.

[41] K. H. K. Reddy, G. Mudali, and D. Sinha Roy, "A novel coordinated resource provisioning approach for cooperative cloud market," *J. Cloud Comput.*, vol. 6, no. 1, p. 8, Dec. 2017, doi: 10.1186/s13677-017-0078-z.

[42] A. Singh, D. Juneja, and M. Malhotra, "A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 1, pp. 19–28, 2017, doi: 10.1016/j.jksuci.2015.09.001.

[43] J. Li, M. Qiu, J.-W. Niu, Y. Chen, and Z. Ming, "Adaptive resource allocation for preemptable jobs in cloud systems," in *2010 10th International Conference on Intelligent Systems Design and Applications*, Nov. 2010, pp. 31–36, doi: 10.1109/ISDA.2010.5687294.

[44] B. P. Rimal, E. Choi, and I. Lumb, “A Taxonomy and Survey of Cloud Computing Systems,” in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 44–51, doi: 10.1109/NCM.2009.218.

[45] P. Seematai S. and B. Koganti, “Dynamic resource allocation using virtual machines for cloud computing environment,” *Int. J. Eng. Adv. Technol.*, vol. 3, no. 6, pp. 218–221, Jun. 2014.

[46] S. P. Abirami and R. Shalini, “Linear Scheduling Strategy for Resource Allocation in Cloud Environment,” *Int. J. Cloud Comput. Serv. Archit.*, vol. 2, no. 1, pp. 9–17, 2012, doi: 10.5121/ijccsa.2012.2102.

[47] “Nimbus.” https://www.nimbusproject.org/ (accessed Oct. 11, 2021).

[48] D. M. Quan *et al.*, “Energy efficient resource allocation strategy for cloud data centres,” in *Computer and Information Sciences II - 26th International Symposium on Computer and Information Sciences, ISCIS 2011*, 2012, no. February 2014, pp. 133–141, doi: 10.1007/978-1-4471-2155-8-16.

[49] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.

[50] W. Shu, W. Wang, and Y. Wang, “A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2014, no. 1, p. 64, Dec. 2014, doi: 10.1186/1687-1499-2014-64.

[51] X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. eds Karamanoglu, *Swarm intelligence and bio-inspired computation: theory and applications*. Elsevier, 2013.

[52] A. V. Karthick, E. Ramaraj, and R. G. Subramanian, “An Efficient Multi Queue Job Scheduling for Cloud Computing,” in *2014 World Congress on Computing and Communication Technologies*, Feb. 2014, pp. 164–166, doi: 10.1109/WCCCT.2014.8.

[53] Qiang Li and Yike Guo, “Optimization of Resource Scheduling in Cloud Computing,” in *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Sep. 2010, pp. 315–320, doi: 10.1109/SYNASC.2010.8.

[54] P. Singh, M. Dutta, and N. Aggarwal, “A review of task scheduling based on meta-heuristics approach in cloud computing,” *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 1–51, Jul. 2017, doi: 10.1007/s10115-017-1044-2.

[55] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” *Egypt. Informatics J.*, vol. 16, no. 3, pp. 275–295, Nov. 2015, doi: 10.1016/j.eij.2015.07.001.

[56] D. Nayak, S. Kumar Malla, and D. Debadarshini, “Improved Round Robin Scheduling using Dynamic Time Quantum,” *Int. J. Comput. Appl.*, vol. 38, no. 5, pp. 34–38, Jan. 2012, doi: 10.5120/4607-6816.

[57] P. Sangwan, M. Sharma, and A. Kumar, “Improved round robin scheduling in cloud computing,” *Adv. Comput. Sci. Technol.*, vol. 10, no. 4, pp. 639–644, 2017.

[58] J. Li, T. Ma, M. Tang, W. Shen, and Y. Jin, “Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing,” *Information*, vol. 8, no. 1, p. 25, Feb. 2017, doi: 10.3390/info8010025.

[59] S. Elmougy, S. Sarhan, and M. Joundy, “A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique,” *J. Cloud Comput.*, vol. 6, no. 1, p. 12, Dec. 2017, doi: 10.1186/s13677-017-0085-0.

[60] P. Thakur and M. Mahajan, “Different Scheduling Algorithm in Cloud Computing: A Survey,” *Int. J. Mod. Comput. Sci.*, vol. 5, no. 1, 2017.

[61] N. Sharma, S. Tyagi, and S. Atri, “HYMM: A New Heuristic in Cloud Computing,” *Int. Res. J. Eng. Technol.*, vol. 04, no. 05, pp. 3520–3526, 2017, [Online]. Available: https://1library.net/document/q2mkdn6y-hymm-a-new-heuristic-in-cloud-computing.html.

[62] R. J. S. Raj and S. V. M. Prasad, “Survey on variants of heuristic algorithms for scheduling workflow of tasks,” in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, Mar. 2016, pp. 1–4, doi: 10.1109/ICCPCT.2016.7530288.

[63] U. Bhoi and P. Ramanuj, “Enhanced Max-min Task Scheduling Algorithm in Cloud Computing,” *Int. J. Appl. or Innov. ...*, vol. 2, no. 4, pp. 259–264, 2013, [Online]. Available: http://ijaiem.org/Volume2Issue4/IJAIEM-2013-04-30-130.pdf.

[64] Huankai Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in *2013 National Conference on Parallel Computing Technologies (PARCOMPTECH)*, Feb. 2013, pp. 1–8, doi: 10.1109/ParCompTech.2013.6621389.

[65] R. Al-Arasi and A. Saif, "Task scheduling in cloud computing based on metaheuristic techniques: A review paper," *EAI Endorsed Trans. Cloud Syst.*, vol. 6, no. 17, pp. 1–19, May 2020, doi: 10.4108/eai.13-7-2018.162829.

[66] X. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Luniver Press, 2010.

[67] S. Kenneth and F. Glover, "Metaheuristics," *Encyclopedia of operations research and management science*, vol. 62. pp. 960–970, 2013.

[68] A. Al-maamari and F. A. Omara, "Task Scheduling using Hybrid Algorithm in Cloud Computing Environments," *IOSR J. Comput. Eng.*, vol. 17, no. 3, pp. 96–106, Oct. 2015, doi: 10.9790/0661-173696106.

[69] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic Scheduling for Cloud: A Survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, Mar. 2014, doi: 10.1109/JSYST.2013.2256731.

[70] Z. Pooranian, M. Shojafar, J. H. Abawajy, and A. Abraham, "An efficient meta-heuristic algorithm for grid computing," *J. Comb. Optim.*, vol. 30, no. 3, pp. 413–434, Oct. 2015, doi: 10.1007/s10878-013-9644-6.

[71] C. T. Joseph, K. Chandrasekaran, and R. Cyriac, "A Novel Family Genetic Approach for Virtual Machine Allocation," *Procedia Comput. Sci.*, vol. 46, pp. 558–565, 2015, doi: 10.1016/j.procs.2015.02.090.

[72] S. Rouhi and E. B. Nejad, "CSO-GA : A New Scheduling Technique for Cloud Computing Systems Based on Cat Swarm Optimization and Genetic Algorithm," *Cumhur. Univ. Fac. Sci. J.*, vol. 36, no. 4, pp. 1672–1685, 2015.

[73] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "An Ant Colony Based Load Balancing Strategy in Cloud Computing," in *Smart Innovation, Systems and Technologies*, 2014, pp. 403–413.

[74] X. Chen, J. Zhang, J. Li, and X. Li, "Resource virtualization methodology for on-demand allocation in cloud computing systems," *Serv. Oriented Comput. Appl.*, vol. 7, no. 2, pp. 77–100, Jun. 2013, doi: 10.1007/s11761-011-0092-9.

[75] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds, "Cloud computing: A new business paradigm for biomedical information sharing," *J. Biomed. Inform.*, vol. 43, no. 2, pp. 342–353, Apr. 2010, doi: 10.1016/j.jbi.2009.08.014.

[76] S. Fu, "Failure-aware resource management for high-availability computing clusters with distributed virtual machines," *J. Parallel Distrib. Comput.*, vol. 70, no. 4, pp. 384–393, Apr. 2010, doi: 10.1016/j.jpdc.2010.01.002.

[77] R. Alonso-Calvo, J. Crespo, M. Garc'ia-Remesal, A. Anguita, and V. Maojo, "On distributing load in cloud computing: A real application for very-large image datasets," *Procedia Comput. Sci.*, vol. 1, no. 1, pp. 2669–2677, May 2010, doi: 10.1016/j.procs.2010.04.300.

[78] Z. Miljanic and P. Spasojevic, "Resource Virtualization with Programmable Radio Processing Platform," in *Proceedings of the 4th International ICST Conference on Wireless Internet*, 2008, doi: 10.4108/ICST.WICON2008.4887.

[79] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Comput.*, vol. 11, no. 3, pp. 213–227, 2008, doi: 10.1007/s10586-008-0060-0.

[80] W. Wang, Y. Jiang, and W. Wu, "Multiagent-Based Resource Allocation for Energy Minimization in Cloud Computing Systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 2, pp. 205–220, 2017, doi: 10.1109/TSMC.2016.2523910.

[81] S. Estivie, "Allocation de Ressources Multi-Agents : Théorie et Pratique," Université Paris Dauphine, 2006.

[82] A. Mazrekaj, D. Minarolli, and B. Freisleben, "Distributed resource allocation in cloud computing using multi-agent systems," *Telfor J.*, vol. 9, no. 2, pp. 110–115, 2017, doi: 10.5937/telfor1702110M.

[83] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965, doi: 10.1016/S0019-

9958(65)90241-X.
[84] R. Fullér, *Introduction to Neuro-Fuzzy Systems*, vol. 2. Heidelberg: Physica-Verlag HD, 2000.
[85] "Fuzzy Logic - Introduction." https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_introduction.htm (accessed Aug. 29, 2021).
[86] S. Foresti, V. Piuri, and G. A. Soares, "On the use of fuzzy logic in dependable cloud management," in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sep. 2015, pp. 767–768, doi: 10.1109/CNS.2015.7346926.
[87] S. Frey, C. Lüthje, V. Huwwa, and C. Reich, "Fuzzy Controled QoS for Scalable Cloud Computing Services," in *The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization Fuzzy*, 2013, pp. 150–155, [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2013_6_30_20107.
[88] D. Minarolli and B. Freisleben, "Virtual Machine Resource Allocation in Cloud Computing via Multi-Agent Fuzzy Control," in *2013 International Conference on Cloud and Green Computing*, Sep. 2013, pp. 188–194, doi: 10.1109/CGC.2013.35.
[89] F. Ramezani, J. Lu, and F. Hussain, "An online fuzzy Decision Support System for Resource Management in cloud environments," in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, Jun. 2013, pp. 754–759, doi: 10.1109/IFSA-NAFIPS.2013.6608495.
[90] P. Pandey and S. Singh, "Fuzzy logic based job scheduling algorithm in cloud environment," *Comput. Model. NEW Technol.*, vol. 21, no. 3, pp. 25–30, 2017.
[91] J. Ejarque, J. Álvarez, R. Sirvent, and R. M. Badia, "Resource allocation for cloud computing: A semantic approach," in *Open Source Cloud Computing Systems: Practices and Paradigms*, 2012, pp. 90–112.
[92] A. Singh, K. Dutta, and A. Singh, "Resource Allocation in Cloud Computing Environment using AHP Technique," *Int. J. Cloud Appl. Comput.*, vol. 4, no. 1, pp. 33–44, 2014, doi: 10.4018/ijcac.2014010103.
[93] D. Lu, J. Ma, and N. Xi, "A universal fairness evaluation framework for resource allocation in cloud computing," *China Commun.*, vol. 12, no. 5, pp. 113–122, May 2015, doi: 10.1109/CC.2015.7112034.
[94] A. S. and K. Chandrasekaran, "Interoperability Based Resource Management in Cloud Computing by Adaptive Dimensional Search," in *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Nov. 2017, pp. 77–84, doi: 10.1109/CCEM.2017.23.
[95] A. Naseri and N. Jafari Navimipour, "A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm," *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 5, pp. 1851–1864, May 2019, doi: 10.1007/s12652-018-0773-8.
[96] H. M. Khan, G. Y. Chan, and F. F. Chua, "A fuzzy model for detecting and predicting cloud quality of service violation," *J. Eng. Sci. Technol.*, vol. 13, no. Special Issue on ICCSIT 2018, pp. 58–77, 2018.
[97] M. Artan, D. Minarolli, and F. Bernd, "Distributed Resource Allocation in Cloud Computing Using Multi-Agent Systems," *Telfor*, vol. 9, no. 2, pp. 110–115, 2017, [Online]. Available: http://journal.telfor.rs/Published/Vol9No2/Vol9No2_A7.pdf.
[98] Q. Li and Y. Guo, "Optimization of resource scheduling in cloud computing," in *12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC*, 2010, pp. 315–320, doi: 10.1109/SYNASC.2010.8.
[99] M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," *Cluster Comput.*, vol. 18, no. 2, pp. 829–844, 2015, doi: 10.1007/s10586-015-0435-y.
[100] N. Baliyan and S. Kumar, "Quality Assessment of Software as a Service on Cloud Using Fuzzy Logic," in *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Oct. 2013, pp. 1–6, doi: 10.1109/CCEM.2013.6684439.
[101] P. Chopra and R. Bedi, "Applications Of Fuzzy Logic in Cloud Computing: A Review," *Int. J. Sci. Res. Eng. Technol.*, vol. 6, no. 11, pp. 1083–1086, 2017, [Online]. Available: www.ijsret.org.

[102] T. F. Mabrouk, “MASCM: Multi Agent System for Cloud Management,” in *Advances in Intelligent Systems and Computing*, vol. 349, no. August, R. Silhavy, R. Senkerik, Z. K. Oplatkova, Z. Prokopova, and P. Silhavy, Eds. Cham: Springer International Publishing, 2015, pp. 371–377.

[103] H. N. Van, F. D. Tran, and J.-M. Menaud, “Performance and Power Management for Cloud Infrastructures,” in *2010 IEEE 3rd International Conference on Cloud Computing*, Jul. 2010, pp. 329–336, doi: 10.1109/CLOUD.2010.25.

[104] H. Aghamohammadi, M. Saadi Mesgari, D. Molaei, and H. Aghamohammadi, “Development a heuristic method to locate and allocate the medical centers to minimize the earthquake relief operation time,” *Iran. J. Public Health*, vol. 42, no. 1, pp. 63–71, 2013.

[105] R. R. Kumar, S. Mishra, and C. Kumar, “Prioritizing the solution of cloud service selection using integrated MCDM methods under Fuzzy environment,” *J. Supercomput.*, vol. 73, no. 11, pp. 4652–4682, Nov. 2017, doi: 10.1007/s11227-017-2039-1.

[106] F. N. N. for R. T. C. A. Kayacan and M. A. Khanesar, “Sliding Mode Control Theory-Based Parameter Adaptation Rules for Fuzzy Neural Networks,” in *Fuzzy Neural Networks for Real Time Control Applications Concepts, Modeling and Algorithms for Fast Learning*, Elsevier Inc., 2016, pp. 85–131.

[107] C. Radhika and R. Parvathi, “Defuzzification of intuitionistic fuzzy sets,” *Notes Intuitionistic Fuzzy Sets*, vol. 22, no. 5, pp. 19–26, 2016, [Online]. Available: http://ifigenia.org/mediawiki/images/1/13/NIFS-22-5-19-26.pdf.

[108] O. A. M. Ali, A. Y. Ali, and B. S. Sumait, “Comparison between the Effects of Different Types of Membership Functions on Fuzzy Logic Controller Performance,” *Int. J. Emerg. Eng. Res. Technol.*, vol. 3, no. 3, pp. 76–83, 2015, [Online]. Available: https://pdfs.semanticscholar.org/ba50/d3d49ed072528c449b14527825abaa5ea2ac.pdf.

[109] B. Y. Ooi, H. Y. Chan, and Y. N. Cheah, “Resource selection using fuzzy logic for dynamic service placement and replication,” in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2011, pp. 128–132, doi: 10.1109/TENCON.2011.6129077.

[110] M. Mayilvaganan and K. Rajeswari, “Health Care Analysis Based On Fuzzy Logic Control System,” *Int. J. Comput. Sci. Trends Technol.*, vol. 2, no. 4, pp. 119–122, 2014.

[111] J. J. Saade and H. B. Diab, “Defuzzification methods and new techniques for fuzzy controllers,” *Iran. J. Electr. Comput. Eng.*, vol. 3, no. 2, pp. 161–174, 2004.

[112] D. Sonali and A. Gayatri, “The modern Weighted Fuzzy First Maxima Defuzzification Technique for Fuzzy Control of VSCHVDC Converters,” *Electr. Syst.*, vol. 3, no. 7–3, pp. 332–342, 2011.

[113] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.

[114] P. Cingolani and J. Alcala-Fdez, “jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation,” in *2012 IEEE International Conference on Fuzzy Systems*, Jun. 2012, pp. 1–8, doi: 10.1109/FUZZ-IEEE.2012.6251215.

# Annexe :
# Publications Scientifiques

## PUBLICATIONS

1. **Abdallah Almutawakel**, Okba Kazar, Mouadh Bali, Belouaar Houcine and Barkat Abdelbasset, "*Smart and fuzzy approach based on CSP for cloud resources allocation*", International Journal of Computers and Applications (December 2019),DOI:10.1080/1206212X.2019.1701241.
2. **Abdallah Almutawakel**, Okba Kazar, and Mouadh Bali, "*A New Design for Smart and Decentralized Approach for Resource Allocation in Cloud Computing using CSP Model*", Journal of Digital Information Management(August 2019). DOI: 10.6025/jdim/2019/17/4/201-213.
3. Mouadh Bali, Abdelkamel Tari, **Abdallah Almutawakel** and Okba Kazar,"*Smart Design for Resources Allocation in IoT Application Service Based on Multi-agent System and DCSP*", Informatica 44(3):373–386, (October 2020),DOI: 10.31449/inf.v44i3.2962.
4. Alwesabi Ali, **Abdallah Almutawakel** and Okba Kazar, "Implementation of Cloud Computing Approach Based on Mobile Agents", (January 2013).
5. Ebrahim Ahmed Ali Alkebsi, Hacene Ameddah, Outtas Toufik and **Abdallah Almutawakel**, "Design of graded lattice structures in turbine blades using topology optimization", International Journal of Computer Integrated Manufacturing, Volume 34, Issue 4 (February 2021),DOI: 10.1080/0951192X.2021.1872106.
6. Ebrahim Ahmed Ali Alkebsi, Outtas Toufik, **Abdallah Almutawakel**, Hacene Ameddah and Toufik Kanit, "Design of Mechanically Compatible Lattice Structures Cancellous Bone Fabricated by Fused Filament Fabrication of Z-ABS Material", International Journal of Mechanics of Advanced Materials and Structures, Volume xx, Issue xx, 2022.
7. E. Alkebsi, O Toufik, H Ameddah, **A. Almutawakel**, Al. Alsamawi "Comparison of Two Materials of a Gas Turbine Blade by Thermomechanical Analysis" Journal of Energy and Thermofluids Engineering, Volume xx, Issue xx, 2022.
8. Al. Alsamawi, N. Boumechra, **A. Almutawakel**, E. Alkebsi, H. Basri, H.Charrak. "Numerical investigation of fully encased composite columns with web and flange shear connectors under cyclic loading" Journal of Civil and Structural Engineering, Volume xx, Issue xx, 2022.

## PUBLICATIONS IN PROGRESS

1. Houcine BELOUAAR, **Abdallah Almutawakel**, Kazar.Okba "**Using fuzzy logic to rank e-commerce websites**" International Journal of Computer Applications, Volume xx, Issue xx, 202x.
2. Ebrahim Ahmed Ali Alkebsi, **Abdallah Almutawakel,** Outtas Toufik, Hacene Ameddah and Toufik Kanit, "Design Mechanical Compatibility of Graded Lattice Structures in Matched Anatomical Ti64 Implant ", Advanced Materials Journal, Volume xx, Issue xx, 202x.

## LIST PAPERS THAT ARE SOON TO BE SUBMITTED

1. dates-Info: Cloud Based Intelligent System for Enhancing the Agricultural Production of dates Using IoT Technology and fuzzy logice.

## نهج تعاوني لا مركزي قائم على الوكيل ومشكلة تلبية القيود المفروضة لتخصيص الموارد في الحوسبة السحابية


**ملخص:**

يجلب تطور الحوسبة السحابية تحديات جديدة تتعلق بتشغيل خدمات الحوسبة السحابية عند الطلب مثل: الحوسبة والتخزين والشبكة. في الواقع، تم اقتراح العديد من الأساليب التجريبية للحفاظ على أنظمة تخصيص موارد الحوسبة السحابية والاستجابة لهذه التحديات بطريقة شفافة وفعالة. في هذا السياق، نعالج مشكلة تخصيص الموارد في السحابة. نقترح نهج تخصيص الموارد الذي يهدف إلى استكشاف هدفين لتحسين تخصيص الموارد. أولاً، يوازن بين الخصائص المختلفة للبنية التحتية السحابية، بما في ذلك موازنة الحمل، مما يحسن أداء البنية التحتية. ثانيًا، يوفر نهجنا حلاً لاحتياجات العميل من خلال تقليل وقت الاستجابة إلى الحد الأدنى وتقليل المدفوعات للموارد المطلوبة ذات الطبيعة الديناميكية.

في هذه الأطروحة، نقترح نهجًا مختلطًا لتخصيص الموارد يعتمد على ثلاث طرق: النظام متعدد العوامل (MAS)، ومشكلة رضا القيد الموزع (PSCD) والمنطق الضبابي (LF). الذي يمثل SMA البنية التحتية المادية للسحابة ويسمح بالإدارة الفعالة للموارد في توزيع هذه البنية التحتية وعدم تجانسها. من ناحية أخرى، يعمل PSCD جنبًا إلى جنب مع SMA للحفاظ على سياسات تخصيص الموارد في مراكز البيانات، بينما يتم استخدام LF لتسهيل تمثيل قيم الموارد الديناميكية من الناحية اللغوية (منخفضة،متوسطة، عالية ...) ويساعد النظام على تحديد الحل الأفضل وفقًا للمعايير في طلبات العملاء.



**كلمات مفتاحية**

تخصيص الموارد ، الحوسبة السحابية ، النظام متعدد العوامل ، مشكلة تلبية القيود الموزعة ، المنطق الضبابي.


## A decentralized, agent-based and CSP-based cooperative approach for resource allocation in cloud computing


**Abstract**

The evolution of Cloud Computing brings new challenges relating to the operation of on-demand cloud computing services such as: computing, storage, network. Indeed, several heuristics are proposed to maintain cloud computing resource allocation systems and respond to these challenges in a transparent and efficient manner. In this context, we address the problem of resource allocation in the cloud. We propose a resource allocation approach that aims to explore two objectives of resource allocation optimization. First, it balances the various peculiarities of the cloud infrastructure, including load balancing, which improves the performance of the infrastructure. Second, our approach provides a solution to the client's needs by minimizing turnaround time and reducing payments for requested resources which are dynamic in nature.

In this thesis, we propose a hybrid resource allocation approach based on three methods: the multi-agent system (MAS), the distributed constraint satisfaction problem (PSCD) and fuzzy logic (LF). Whose SMA represents the physical infrastructure of the cloud and allows efficient management of resources in the distribution and heterogeneity of this infrastructure. PSCD, on the other hand, works side by side with SMA to maintain resource allocation policies in data centers, while LF is used to facilitate the representation of dynamic resource values in linguistic terms (low, medium, high ...) and helps the system to determine the best solution according to the criteria in customer requests.



**Keywords**

Resource allocation, Cloud computing, Multi-agent system, Problem of satisfaction of distributed constraints, Fuzzy logic.


## Une approche coopérative décentralise basée agent et CSP pour l’allocation de ressource dans le cloud computing


**Résumé**

L’évolution de Cloud Computing permet d'apporter des nouveaux défis relatifs à l’exploitation des services à la demande du cloud Computing tels que : calcul, stockage, réseau. En effet, plusieurs heuristiques sont proposées pour maintenir les systèmes d’allocation des ressources de cloud computing et répondent aux ces défis d'une manière transparente et efficace. Dans ce contexte, nous abordons le problème d'allocation des ressources dans le cloud. Nous proposons une approche d'allocation des ressources qui vise à explorer deux objectifs d'optimisation d’allocation des ressources. Premièrement, il équilibre les différentes particularités de l'infrastructure de cloud, y compris l'équilibrage de charge, ce qui améliore les performances de l'infrastructure. Deuxièmement, notre approche fournit une solution aux besoins du client en minimisant le temps d'exécution et en réduisant les paiements des ressources demandées qui ont une nature dynamique.

Dans cette thèse, Nous proposons une approche d'allocation de ressources hybride basée sur trois méthodes : le système multi-agents (SMA), le problème de satisfaction de contraintes distribuées (PSCD) et la logique floue (LF). Dont le SMA représente l'infrastructure physique du cloud et permet une gestion efficace des ressources dans la distribution et l'hétérogénéité de cette infrastructure. PSCD, d'autre part, travaille côte à côte avec SMA pour maintenir les politiques d'allocation des ressources dans les centres de données, tandis que LF est utilisée pour faciliter la représentation des valeurs de ressources dynamiques en termes linguistiques (faible, moyen, élevé ...) et aide le système à déterminer la meilleure solution selon les critères dans les demandes des clients.



**Mots clés**

Allocation des ressources, Cloud computing, Système multi-agents, Problème de satisfaction des contraintes distribuées, Logique floue.