الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة عين تموشنت بلحاج بوشعيب
Université –Ain Temouchent- Belhadj Bouchaib
Faculté des Sciences et de Technologie
Département des Mathématiques et Informatique

Projet de Fin d'Etudes
Pour l'obtention du diplôme de Master en : Informatique
Domaine : Mathématique et Informatique
Filière : Informatique
Spécialité : Réseaux et Ingénierie des Données

Thème :

Développement d'un Système Multi-agents pour l'Ordonnancement des Tâches dans le Cloud Computing

Présenté par :

1) Melle Berrached Fatima Zohra.

2) Melle Boumedol Hadjar .

Devant le jury composé de :

M. A. BENZERBADJ	MCB	UAT.B.B (Ain Temouchent)	Président
Mme F. Z. BERRAKEM	MAA	UAT.B.B (Ain Temouchent)	Examinatrice
M. Z. BOUAFIA	MAA	UAT.B.B (Ain Temouchent)	Encadrant

Année Universitaire : 2023/2024

Remerciements

En premier lieu, nous remercions « Allah » de nous avoir donné la force et la patience nécessaire pour achever ce mémoire. Nous tenons à remercier notre Encadrant Monsieur BOUAFIA ZOUHEYR , pour son aide à réaliser notre mémoire ainsi que pour sa disponibilité et son soutien.

Nous tenons également à adresser nos remerciements aux membres du jury M.A.BENZERBADJ et Mme F.Z.BERRAKEM,de nous avoir fait l'honneur d'accepter de participer à notre jury de mémoire.

Nous tenons aussi à saluer toute notre promotion de Master 2 Réseaux et ingénierie des données et tous nos amis. Enfin, que tous ceux qui ont participé de près ou de loin à la réalisation de ce travail, trouvent ici le témoignage de notre profonde reconnaissance.

Dédicaces

-Hadjar-

Je dédie ce travail :

À ma mère, qui sans cesse me conseille et me soutient moralement et spirituellement.

À mon père, pour son assistance, lui qui n'a ménagé aucun effort pour assurer mon éducation .

À mes sœurs et à mon frère pour leur appui et leur soutien multiforme.

À toute ma famille pour l'amour et le respect qu'ils m'ont toujours accordé.

À mon binôme pour la sœur agréable.

À tous mes amis.

Dédicaces

-Fatima-

Je dédie ce modeste travail :

À mes chers parents qui n'ont jamais cessé de m'encourager à poursuivre mes études et à atteindre mes objectifs.

À mes chères sœurs.

À tous les membres de ma famille, tantes, oncles, cousins.

À mon binôme, qui était une de mes très proches collègues.

À tous mes camarades de classe.

Sommaire

Liste des tableaux

Liste des abréviations

AMA ..... Algorithme Multi-Agents

BDI ..... Believe Desire Intention

CPU ..... Central Processing Unit

CIS ..... Cloud Information Services

IaaS ..... Infrastructure as a Service

Makespan ..... Le temps d'exécution

MIPS ..... Million Instructions Per Second

NIST ..... National Institute of Standard and Technology

PaaS ..... Platform as a Service

PE ..... Processeur Element

RAM ..... Mémoire vive (Random Access Memory)

RR ..... Round Robin

SaaS ..... Software as a Service

SMA ..... Les Systèmes Multi-agents

SLA ..... Service-Level-agreement

SJF ..... Shortest Job First

VM ..... Machine Virtuelle

QOS ..... Quality of Service

Introduction Générale

Introduction Générale

La combinaison entre le Cloud Computing et les Systèmes Multi-agents(SMA) permet de développer des techniques innovantes. Une nouvelle discipline, connue sous le nom de cloud computing à base d'agents, a émergé pour proposer des solutions axées sur les agents. Grâce aux agents, le cloud devient plus intelligent dans ses interactions avec les utilisateurs et plus efficace dans l'allocation des ressources de traitement et de stockage. Actuellement, des recherches sont en cours pour mettre au point des solutions à base d'agents efficaces pour le cloud computing.

Dans un environnement de Cloud Computing, l'ordonnancement des tâches est d'une importance cruciale pour optimiser l'utilisation des ressources disponibles. Cela implique d'ordonnancer et d'exécuter efficacement différentes tâches ou travaux sur des serveurs cloud, en tenant compte des contraintes de temps. Cependant, les méthodes traditionnelles d'ordonnancement des tâches peuvent rencontrer des difficultés lorsqu'un hôte de gestion tombe en panne, provoquant des perturbations dans l'ensemble du système.

C'est là que les Systèmes Multi-agents peuvent jouer un rôle crucial. En offrant à chaque agent une autonomie de décision et en lui permettant d'interagir de manière décentralisée, ces systèmes peuvent fournir une solution flexible et adaptable pour l'ordonnancement des tâches dans le Cloud Computing. En permettant à chaque agent d'évaluer indépendamment les ressources disponibles, de coordonner ses actions avec d'autres agents et de prendre des décisions en temps réel, SMA peut contribuer à améliorer l'efficacité opérationnelle et à réduire les effets des pannes matérielles ou des goulots d'étranglement sur les performances du système.

Notre objectif principal, dans le cadre du projet de fin d'études, est de proposer un algorithme d'ordonnancement de tâches basé sur des Systèmes Multi-agents, le but étant notamment d'optimiser le temps d'exécution des tâches.

Pour ce faire, nous avons organisé notre rapport selon les quatre chapitres suivants :

— Chapitre 1 : Nous examinerons certains concepts du Cloud Computing pour bien maitriser notre domaine de travail, notamment sa structure, ses principes de base, ses fonctionnalités de base et d'autres aspects clés pour bien le comprendre.

— Chapitre 2 : Dans ce chapitre, nous introduisons différents concepts de Systèmes Multi-agents et explorons leurs concepts de base et son utilisation dans le Cloud Computing.

— Chapitre 3 : Dans ce chapitre, nous améliorerons notre compréhension des algorithmes de base utilisés pour l'ordonnancement des tâches dans le Cloud Computing,avec un accent particulier sur ceux liés aux Systèmes Multi-agents,en soulignant les avantages et les limites de chaque approche.

— Chapitre 4 : Notre attention se tournera vers l'environnement de développement de Systèmes Multi-agents. Nous présentons les outils, langages de programmation et plates-formes utilisés dans la conception et la mise en œuvre de ces systèmes. Ensuite, nous commencerons à créer notre propre algorithme basé sur des Systèmes Multi-agents pour l'ordonnacement des tâches, en nous appuyant sur les connaissances acquises dans les chapitres précédents.

Nous procéderons à une comparaison entre les algorithmes de base et notre algorithme proposé.

Nous terminons ce rapport en fournissant une Conclusion Générale.

Chapitre 1 : Introduction au Cloud Computing

Chapitre 1

Introduction au Cloud Computing

1.1 Introduction

Le Cloud Computing a révolutionné le paysage informatique. Il offre une multitude d'avantages,notamment en termes de flexibilité, d'évolutivité et de réduction des coûts. Le Cloud Computing consiste en une technologie innovante offrant aux entreprises la possibilité d'externaliser le stockage de leurs données et de bénéficier d'une puissance de calcul supplémentaire pour le traitement de grandes quantités d'informations. Cette solution permet non seulement d'améliorer l'efficacité opérationnelle, mais également de réduire les coûts liés à la gestion des infrastructures informatiques. Dans ce chapitre, nous allons présenter quelques généralités sur le Cloud Computing, à savoir sa définition, ses caractéristiques, les services qu'il offre, ses éléments constitutifs et enfin on termine avec ses avantages et inconvénients.

1.2 Historique

Le concept du Cloud Computing remonte aux années 1950, lorsque les utilisateurs accédaient à des applications via des terminaux connectés à des systèmes centraux (mainframes).

En 1961, John McCarthy proposa d'accéder aux services sans passer par ces systèmes centraux.

En 1962, J.C.R. Licklider, impliqué dans le développement d'ARPANET, esquissa les premières idées de réseau informatique mondial, préfigurant des concepts similaires à Internet et au cloud.

Avec l'avènement d'Internet dans les années 1980, le stockage cloud devint progressivement standard.

En 1993, Internet se popularisa grâce aux navigateurs, et des cadres de Compaq, ainsi que Sean O'Sullivan, proposèrent le concept de cloud. Ramnath Chellappa utilisa le terme "cloud" en 1997 pour décrire un nouveau paradigme informatique.

En 1999, Salesforce introduisit la notion de "SaaS",fournissant des applications via le Web.

En 2006, Google et Amazon popularisèrent le terme "cloud computing", déployant des services comme le courrier électronique et des outils collaboratifs.

Amazon avait déjà commencé à louer ses serveurs sur demande dès 2000.

Le cloud évolua avec Internet, surtout après l'apparition de navigateurs comme Mosaic et Netscape en 1993-1994, et de sites comme eBay et Amazon.

Le Cloud Computing s'est accéléré dans les années 2000 avec la montée en puissance d'Internet et l'arrivée de Google. Bien que certains analystes considèrent le cloud comme une avancée

majeure, d'autres le voient comme une mode à motivation commerciale. La pandémie de Covid-19 a provoqué une forte hausse de l'utilisation du cloud, avec une augmentation de 35 % du marché entre fin 2019 et début 2020 [Astran, 2022].

1.3 Définition

Le Cloud signifie « nuage » et Computing « informatique », le Cloud Computing est donc l'informatique en nuage. Bien que nous avons trouvé de nombreuses définitions disponibles, nous retiendrons cependant la définition la plus utile, la plus complète et la plus populaire provient de l'Institut national des normes et de la technologie (NIST) qui définit le Cloud Computing comme étant un modèle qui offre aux utilisateurs du réseau un accès à la demande, à un ensemble de ressources informatiques partagées et configurables (par exemple, réseaux serveurs, stockage, applications et services) qui peuvent être rapidement mises à la disposition du client et publiées avec un effort de gestion minimal et sans interaction directe avec le prestataire de service [Mell and Grance, 2011].

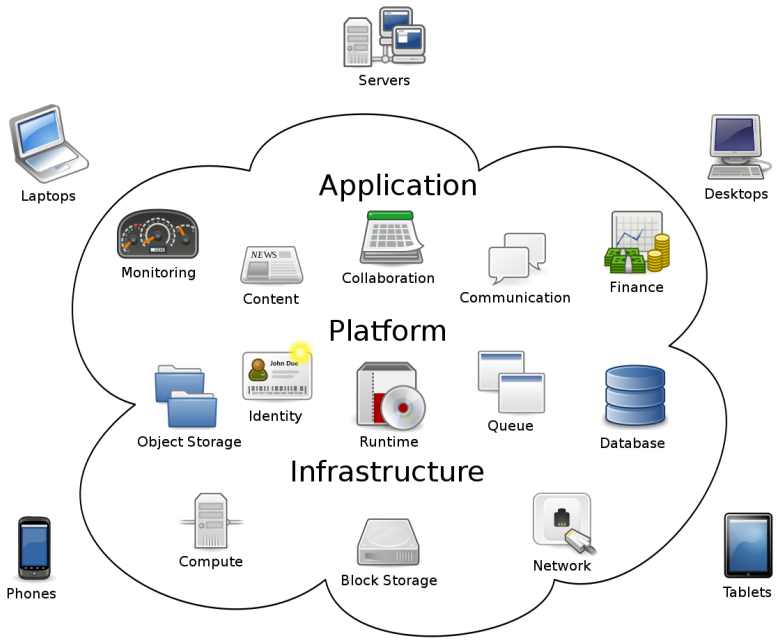

FIGURE 1.1 – Cloud Computing
[Bouzara, 2018]

1.4 Notions de base

— Le Centre de Données « Data Center » :Un cloud a besoin de serveurs sur un réseau, et ils ont besoin d'une maison (station). Cette maison physique et tout le matériel un centre de données. Le centre de données est un site hébergeant l'ensemble des systèmes

nécessaires au fonctionnement des applications informatiques. Il est toujours constitué de trois composants élémentaires :

- L'infrastructure, c'est à dire l'espace et les équipements nécessaires au support des opérations du centre de données. Cela comprend les transformateurs électriques, les alimentations, les générateurs, les armoires de climatisation, les systèmes de distribution électrique, etc.

- Les équipements informatiques comprenant les serveurs, le stockage, le câblage ainsi que les outils de gestion des systèmes et des équipements réseaux.

- Les espaces d'exploitation, c'est-à-dire le personnel d'exploitation qui pilote,entretient et répare les systèmes lorsque cela est nécessaire [Maaref, 2012].

— Machine Virtuelle : Une machine virtuelle est un ordinateur logiciel qui, à l'instar d'un ordinateur physique, exécute un système d'exploitation et des applications. La machine virtuelle se compose d'un ensemble de fichiers de spécification et de configuration ;elle est secondée par les ressources physiques d'un hôte [Khelifi and Bellil, 2014].
— Machine Physique :Une machine physique (Hôte, Host) est un matériel physique doté d'une puissance de calcul, capacité de stockage que les machines virtuelles se servent de ses ressources (CPU, RAM, stockage...),pour exécuter leurs tâches.
— Virtualisation :La virtualisation constitue le coeur du cloud computing. Elle consiste à ajouter une couche logicielle qui permet de faire l'abstraction entre le matériel et le système d'exploitation dans le but de faire fonctionner plusieurs systèmes d'exploitation sur la même machine physique [Hennion et al., 2012].
— Broker :agissant pour le compte d'un client cloud. Il masque la gestion des VM telles que la création de VM, la soumission de cloudlets aux VM et la destruction des VM.
— Tâche (cloudlet) :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'opérations qui requièrent, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif [Patel and Patel, 2013].
— Makespan :représente le temps de finition maximum parmi toutes les tâches reçues par heure [Mohialdeen, 2013].
— Coût :Le coût total engendré par l'exécution d'une application distribuée peut comprendre de nombreux éléments de coût, tels que le coût de calcul et le coût de transfert de données [Zeng et al., 2015].

1.5 Les caractéristiques du Cloud Computing

Bien qu'il n'y ait pas de description unique pour tous les environnements Cloud, presque tous ont une série de propriétés communes qui sont à la base d'un produit de qualité. Comme l'indique la définition fournie par le NIST [Mell and Grance, 2011], il existe cinq caractéristiques principales du Cloud Computing parmi lesquelles :

— Le self-service à la demande :qui permet au client de fournir des aptitudes informatiques en fonction de ses besoins, sans l'intervention d'un individu avec le prestataire de service.
— L'accessibilité en réseau :où les compétences sont présentes sur le réseau et peuvent être obtenues via des protocoles standard pour une utilisation sur différentes plateformes telles que Smartphones, tablettes, PC et stations de travail.
— L'agrégation de ressources :où les outils informatiques du prestataire sont rassemblés pour

assister plusieurs clients dans un spécimen multilocataire, avec des outils matériels et virtuels alloués et réalloués selon la requête. Il y a une perception d'autonomie géographique, car l'acquéreur ne connaît généralement pas le lieu exact des ressources, mais peut les spécifier à un niveau de conception supérieur.
— L'élasticité rapide :où les aptitudes sont ravitaillées rapidement pour s'adapter à la demande, parfois automatiquement. Les dispositions semblent souvent absolues pour le client.
— La mesure du service :où les systèmes en nuage contrôlent et maximalisent l'exploitation des moyens en utilisant une capacité de cadence (généralement sur une assise d'emploi ou de facturation) pour différents types de services tels que le stockage, le traitement, la bande passante et les comptes d'utilisateurs actifs. L'exploitation des outils est examinée, vérifiée et rapportée, proposant ainsi une fiabilité pour le prestataire du service et le client.

1.6 Modèles de service

On distingue plusieurs types de services cloud, nous n'aborderons que les principaux types dans ce mémoire :

- SaaS :Software as-a-Service.
- PaaS :Platform as-a-Service.
- IaaS :Infrastructure as-a-Service.

1.6.1 Software as-a-Service

Ce modèle désigne des applications prêtes à l'utilisation et offertes à distance par un fournisseur de services cloud. Dans ce modèle, toutes les procédures d'installation, de mise à jour et de maintenance de l'application et de son environnement d'exécution sont effectuées par le fournisseur. Parmi les applications SaaS les plus populaires, nous citons la suite logicielle de collaboration Google Apps,l'application de gestion de la relation client d'Oracle (Sage CRM), etc. Les solutions SaaS sont généralement soit disponibles pour une utilisation gratuite, soit facturées sur la base d'un abonnement [Voorsluys et al., 2011].

1.6.2 Platform as-a-Service

Les services Cloud de type PaaS sont généralement conçus pour les développeurs de logiciels. Ils fournissent aux utilisateurs des plates-formes pour développer, tester et déployer leurs applications cloud. Les plates-formes PaaS sont des environnements intégrés de haut niveau (systèmes d'exploitation, langages de programmation, librairies, bases de données, etc.). prenant en charge le cycle de vie complet du logiciel [Singh and Chana, 2016].

1.6.3 Infrastructure as-a-Service

IaaS fait référence à l'approvisionnement à la demande en ressources informatiques de base (calcul, mémoire, stockage et réseau).Les ressources de ce modèle sont utilisées comme des ressources virtualisées. Dans ce modèle, les clients IaaS ont un meilleur contrôle sur leurs ressources par rapport aux modèles SaaS et PaaS. Ils sont responsables de la gestion du système d'exploitation, des applications et des données déployées, tandis que les fournisseurs IaaS gèrent toujours le matériel sous-jacent et les couches de virtualisation [Voorsluys et al., 2011].

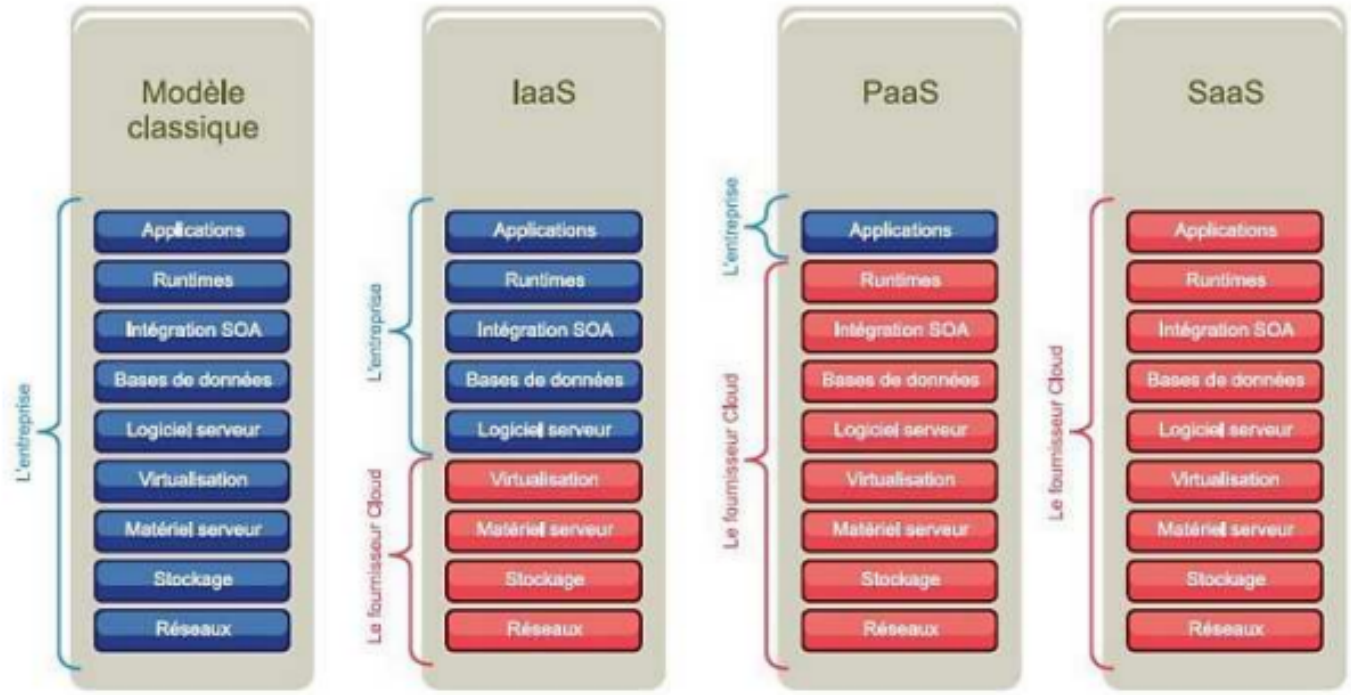

FIGURE 1.2 – IaaS SaaS PaaS
[Hamdani et al., 2019]

1.7 Modèles de déploiement

1.7.1 Cloud privé

L'infrastructure Cloud est utilisée par une seule organisation. Elle peut être gérée par l'organisation ou par une tierce partie. L'infrastructure peut être placée dans les locaux de l'organisation ou à l'extérieur [Figer, 2012].

1.7.2 Cloud public

L'infrastructure Cloud est ouverte au public ou à de grands groupes industriels. Cette infrastructure est possédée par une organisation qui vend des services Cloud. C'est le cas le plus courant. C'est celui de la plate-forme Amazon Web Services [Figer, 2012].

1.7.3 Cloud hybride

L'infrastructure Cloud est composée d'un ou plusieurs modèles ci-dessus qui restent des entités séparées. Ces infrastructures sont liées entre elles par la même technologie qui autorise la portabilité des applications et des données.C'est une excellente solution pour répartir ses moyens en fonction des avantages recherchés [Figer, 2012].

1.7.4 Cloud communautaire

L'infrastructure Cloud est partagée par plusieurs organisations pour les besoins d'une communauté qui souhaite mettre en commun des moyens (sécurité, conformité, etc..).Elle peut être gérée par les organisations ou par une tierce partie et peut être placée dans les locaux ou à l'extérieur [Figer, 2012].

1.8 Avantages et inconvénients

— Avantages :

-L'agilité pourl'entreprise :Résolution des problèmes de gestion informatique simplement sans avoir à vous engager à Long terme.

-Un développement plus rapide des produits :Réduisons le temps de recherche pour les développeurs sur le paramétrage des Applications.

-Pas de dépenses de capital : Plus besoin des locaux pour élargir vos infrastructures informatiques.

— Inconvénients :

-La bande passante peut faire exploser votre budget : La bande passante qui serait nécessaire pour mettre cela dans le Cloud est gigantesque, et les coûts seraient tellement importants qu'il est plus avantageux d'acheter le stockage nous-mêmes plutôt que de payer quelqu'un d'autre pour s'en charger.

-Les performances des applications peuvent être amoindries : Un Cloud public n'améliorera définitivement pas les performances des applications.

-La fiabilité du Cloud : Un grand risque lorsqu'on met une application qui donne des avantages compétitifs ou qui contient des informations clients dans le Cloud.

-Taille de l'entreprise : Si votre entreprise est grande alors vos ressources sont grandes, ce qui inclut une grande consommation du cloud. Vous trouverez peut-être plus d'intérêt à mettre au point votre propre Cloud plutôt que d'en utiliser un externalisé. Les gains sont bien plus importants quand on passe d'une petite consommation de ressources à une consommation plus importante [GmbH, 2009].

1.9 Conclusion

Le Cloud Computing émerge comme un catalyseur majeur dans l'évolution du paysage informatique contemporain. En offrant une flexibilité inégalée, une évolutivité sans limites et des économies substantielles de coûts.

Dans ce chapitre, nous avons exposé les concepts fondamentaux ainsi que les caractéristiques essentielles du Cloud Computing offrant un aperçu complet de son impact sur le monde de l'informatique.

Dans le chapitre suivant nous allons présenter quelques concepts sur Les Systèmes Multi-agents. Nous allons aussi mettre l'accent sur la relation entre Systèmes Multi-agents et Cloud Computing.

Chapitre 2 : Les Systèmes Multi-agents

Chapitre 2

Les Systèmes Multi-agents

2.1 Introduction

Les Systèmes Multi-agents se sont progressivement imposés comme une composante majeure de l'informatique ces dernières années, intervenant dans divers domaines dont les systèmes distribués.

Ces systèmes sont aujourd'hui confrontés à des exigences croissantes d'interactivité, de réactivité et de mobilité dans leurs usages les plus courants. L'introduction des Systèmes Multi-agents (SMA) apporte ainsi une nouvelle perspective à la modélisation des applications du monde réel, en offrant une représentation qui s'adapte à leur complexité et à leur dynamisme, constituée d'agents logiciels interagissant pour atteindre un objectif commun.

Dans ce chapitre, nous discuterons de quelques généralités sur les Systèmes Multi-agents, à savoir leur définition, leurs caractéristiques, leurs domaines d'application et leurs types.

2.2 Définition d'un agent

Dans le domaine informatique, le terme « agent » a plusieurs définitions et il n'y a pas de définition commune. Cependant, l'auteur « Ferber »définit un agent comme étant une entité physique ou virtuelle

a) qui est capable d'agir dans un environnement,

b) qui peut communiquer directement avec d'autres agents,

c) qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),

d) qui possède des ressources propres,

e) qui est capable de percevoir (mais de manière limitée) son environnement,

f) qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),

g) qui possède des compétences et offre des services,

h) qui peut éventuellement se reproduire,

i) dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit [Ferber, 1997].

2.3 Caractéristiques d'un agent

— Situé :l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.

— Autonome :l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.

— Proactif :l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.

— Capable de répondre à temps :l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis.

— Social :l'agent doit être capable d'interagir avec des autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs.

— Intentionnalité :un agent intentionnel est un agent guidé par ses buts, une intention est la déclaration explicite des buts et des moyens d'y parvenir. Elle exprime donc la volonté d'un agent d'atteindre un but ou d'effectuer une action.

— Rationalité :un agent rationnel est un agent qui suit le principe suivant "Si un agent sait qu'une de ses actions lui permet d'atteindre un de ses buts, il la sélectionne". La notion de rationalité se rapporte au comportement cognitif de l'agent. Ce terme qualifie l'utilisation efficace des ressources par l'agent.

— Engagement :la notion d'engagement est une qualité essentielle des agents coopératifs.Un agent coopératif planifie ses actions par coordination et négociation avec les autres agents. En construisant un plan pour atteindre un but, l'agent se donne les moyens d'y parvenir et donc s'engage à accomplir les actions qui satisfont ce but :l'agent croit qu'il est en mesure d'exécuter tout le plan qu'il a élaboré, ce qui le conduit (ainsi que les autres agents) à agir en conséquence.

— Adaptabilité :un agent adaptatif est un agent capable de contrôler ses aptitudes (communicationnelles, comportementales) selon l'agent avec qui il interagit. Un agent adaptatif est un agent d'un haut niveau de flexibilité.

— Intelligence :un agent intelligent est un agent cognitif, rationnel, intentionnel et adaptatif [Touaf, 2005].

2.4 Les types d'agents

2.4.1 Les agents Réactifs

Comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent ne fait ni délibération ni planification, il se contente simplement d'acquérir des perceptions et de réagir à celles-ci (Figure 2.1). Étant donné qu'il n'y a pratiquement pas de raisonnement, ces agents peuvent agir et réagir très rapidement.

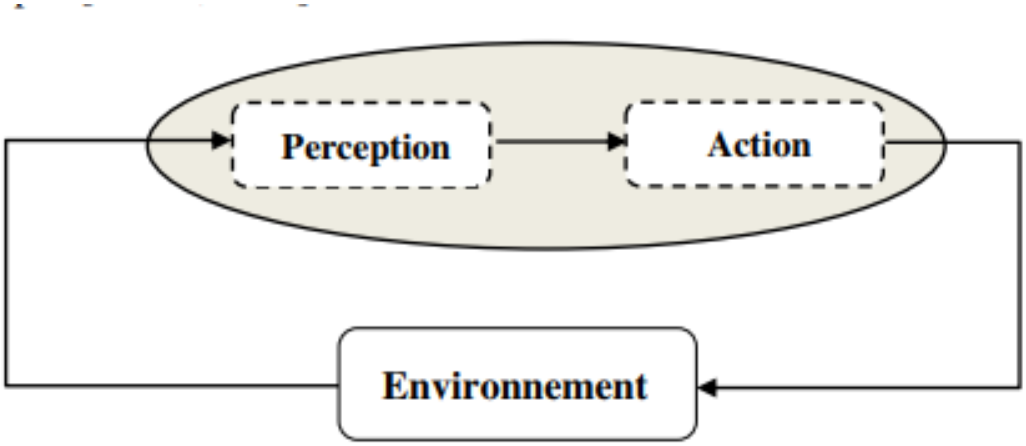

FIGURE 2.1 – Les Agents Réactifs
[Benhamza, 2016]

L'agent réactif (Figure 2.1) présente les caractéristiques suivantes :
-Pas de mémoire ;
-Pas de représentation explicite ;
-Prise de décision se basant sur le fait du Stimulus/Réponse ;
-Simple à mettre en œuvre.

2.4.2 Les agents cognitifs

L'agent cognitif est un agent qui dispose d'une base de connaissances comprenant l'ensemble des informations et de savoir-faire nécessaires à la réalisation de sa tâche et la gestion des interactions avec les autres agents et avec son environnement .Ainsi, ces agents possèdent une représentation explicite de leur environnement, des autres agents et d'eux-mêmes. Ils sont aussi dotés de capacités de raisonnement et de planification ainsi que de communication. Le travail le plus représentatif de cette famille d'agent porte sur le modèle BDI (Believe Desire Intention). Ce modèle se fonde sur trois attitudes qui définissent la rationalité d'un agent intelligent :

-Croyance (Belief B) : ce sont les informations que possède l'agent sur son environnement et sur les autres agents agissant sur le même environnement. Ceci constitue les connaissances supposées vraies de l'agent.

-Désir (Desire D) : ce sont les états de l'environnement et parfois de lui-même, qu'un agent aimerait voir réaliser. Ce sont les objectifs que se fixe un agent.

-Intention (Intention I) : ce sont les actions qu'un agent a décidé de faire pour accomplir ses désirs. Ils forment des ensembles de plans qui sont exécutés tant que l'objectif correspondant n'est pas atteint. Le modèle BDI a inspiré beaucoup d'architectures d'agents cognitifs.

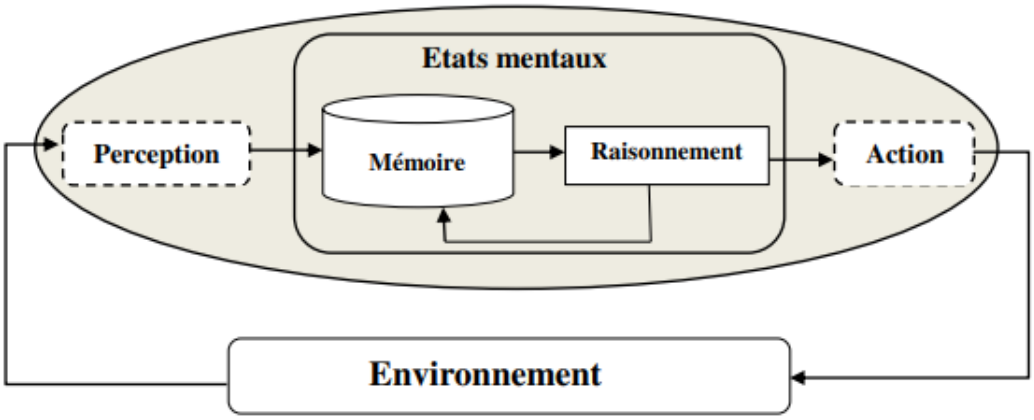

FIGURE 2.2 – Les Agents cognitifs
[Benhamza, 2016]

Ce type d'agent (Figure 2.2) se caractérise par : - Une représentation explicite de l'environnement et du monde auquel ils appartiennent ;

-Une réaction planifiée ;

- Une base de connaissances comprenant des informations et du savoir-faire ;

- Une mémoire pour mémoriser les anciens états. Il est possible de combiner ces deux architectures pour obtenir une troisième architecture qui se base sur des agents cognitifs exhibant des capacités de réactions aux événements, on parlera alors d'agents hybrides.

2.4.3 Les agents hybrides

L'agent hybride est conçu pour combiner les capacités réactives à des capacités cognitives, ce qui leur permet d'adapter son comportement en temps réel à l'évolution de l'environnement [Benhamza, 2016].

2.5 Différence entre un objet et un agent

Il est intéressant de remarquer qu'un agent n'est pas un objet. L'auteur « Wooldridge» présente trois différences fondamentales [Wooldridge, 1999] :

-La première différence est qu'une entité extérieure peut manipuler directement l'état interne d'un objet (avec des attributs publics par exemple) alors qu'on ne peut pas manipuler l'état interne d'un agent ni l'obliger à effectuer une fonction.

-La deuxième différence est le caractère autonome du comportement d'un agent contrairement à un objet qui reçoit des ordres.

-La troisième différence concerne l'exécution des comportements. Dans le cas d'un agent, l'exécution ne dépend que de lui. Alors que dans le cas d'un objet, cela dépend du système (plateforme logicielle par exemple) auquel il appartient et qui planifie ses temps et durées d'exécution.

2.6 Définition d'un Système Multi-agents

Un Système Multi-agents est un ensemble d'agents qui évoluent dans un environnement commun. Dans [Weiss, 1999], Gerhard Weiss définit l'intelligence artificielle distribuée comme étant l'étude, la conception et la réalisation de Systèmes Multi-agents qu'il présente comme étant des systèmes dans lesquels des agents intelligents interagissent et poursuivent un ensemble de buts ou réalisent un ensemble d'actions. - On appelle Système Multi-agents, un système composé des éléments suivants : un ensemble d'agents, un ensemble de tâches à réaliser et un ensemble d'objets associés à l'environnement.

2.7 Interactions entre les agents

La notion d'interaction constitue l'essence d'un Système Multi-agents puisque c'est grâce à elle que les agents vont pouvoir produire des comportements collectifs complexes et dépendants les uns des autres. En effet, la fonction interactionnelle d'un agent porte sur l'ensemble des mécanismes lui permettant de faire le lien avec ce qui l'entoure (son environnement ainsi que l'ensemble des autres agents). L'interaction peut être vue comme une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques [Thomas, 2005]. On distingue différents types d'interaction que les agents peuvent adopter tels que : la coopération, la négociation et la coordination [ElWessabi, 2014].

2.7.1 La coopération

Chaque agent possède un ensemble de compétences qui lui permettent de résoudre les différents problèmes, mais il existe des situations où ses capacités et ses compétences ne suffisent pas à accomplir certaines tâches (ou bien il ne dispose pas des moyens nécessaires) donc il aura besoin de l'intervention d'un autre agent du système qui va l'aider à résoudre le problème, c'est-à-dire qu'il y'a une coopération pour faire évoluer le système vers ses objectifs. La coopération consiste, donc, à faire participer plusieurs agents pour satisfaire un but individuel ou commun.

2.7.2 La négociation

On définit la négociation comme le processus d'améliorer les accords (en réduisant les inconsistances et l'incertitude) sur des points de vue communs ou des plans d'action grâce à l'échange structuré d'informations pertinentes.

2.7.3 La coordination

La coordination entre les agents est présente lorsque les agents utilisent des ressources communes ou résolvent des problèmes qui ne sont pas complètement indépendants mais liés et com-

plémentaires. Les agents du système doivent accomplir en plus de leurs tâches de résolution des problèmes individuels, des tâches supplémentaires (appelées tâches de coordination) qui améliorent le fonctionnement du système.

2.8 La communication entre les agents

La communication est la base de la résolution coopérative des problèmes. Elle permet de synchroniser les actions des agents et résoudre les conflits de ressources et de buts par la négociation. Les deux principaux modes de communications dans les Systèmes Multi agents sont [Asnoune, 2016] :

a) Communication par envoi de messages : Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire : « Si un agent A connait l'agent B, alors il peut entrer en communication avec lui »

-Mode point à point :l'agent émetteur du message connaît et précise l'adresse de ou des agent(s) destinataire(s). Ce type de communication est généralement le plus employé par les agents cognitif.

-Mode par diffusion :le message est envoyé à tous les agents du système. Ce type de transmission est très utilisé dans les systèmes dynamiques ainsi que les systèmes d'agents réactifs. En fait, ceci suppose en général une messagerie : un agent spécialisé gère autant de files d'attente que de destinataires, chaque agent peut traiter le premier message de sa file.

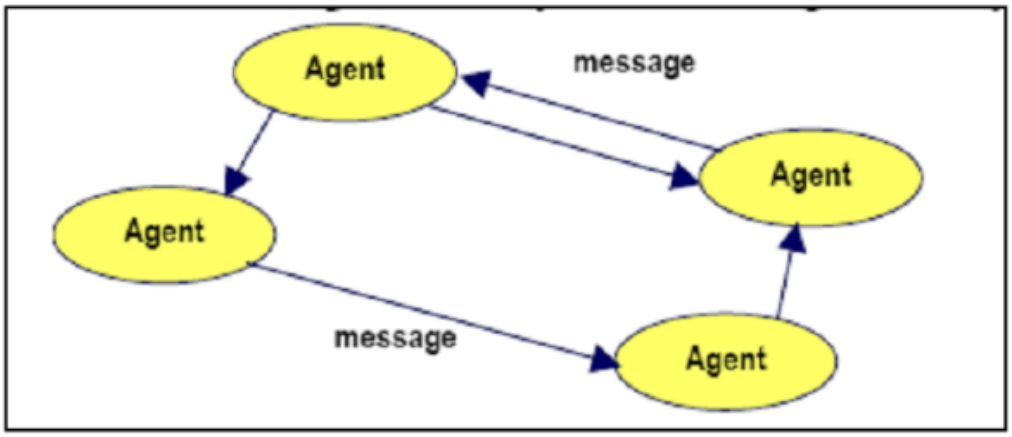

FIGURE 2.3 – Communication par envoi de messages
[Asnoune, 2016]

b) Communication par partage d'information : Les agents ne sont pas en liaison directe mais communiquent via une structure de données partagée, ou on trouve les connaissances relatives à la résolution (état courant du problème) qui évoluent durant les processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des Systèmes Multi-agents. Le meilleur exemple qui utilise ce type de communication est les systèmes à tableau noir.

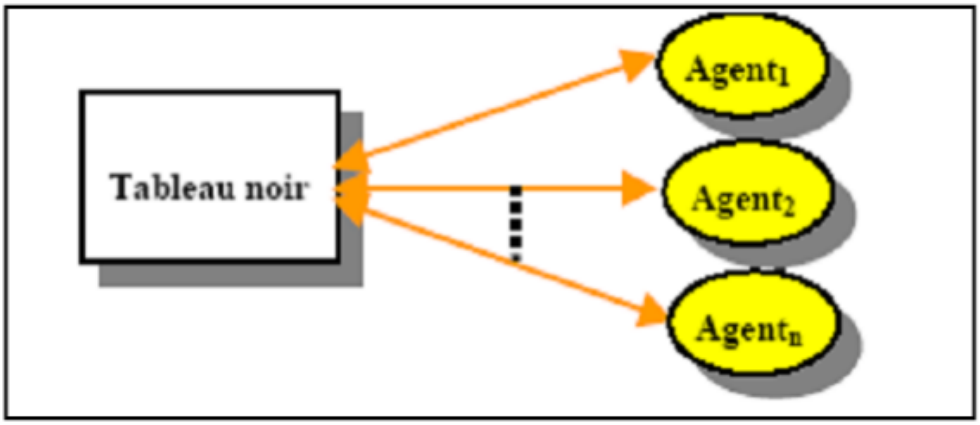

FIGURE 2.4 – Communication par partage d'information
[Asnoune, 2016]

2.9 Domaine d'application des Systèmes Multi-agents

Les différentes applications des SMA relèvent de plusieurs domaines [Marzougui, 2014] :

-La résolution de problèmes par émergence :faire émerger une solution à un problème Complexe à partir de comportements simples tels que la gestion des réseaux de télécommunication, système de transport.

-Le contrôle de systèmes complexes :les SMA sont aussi utilisés pour le contrôle ou le Pilotage d'outils ou de composants immergés dans un environnement dynamique, telles les chaînes de production ou encore les robots tels que les robots automates mobiles.

-La simulation des systèmes complexes :c'est là que les caractéristiques d'autonomie, de Proactivité, des agents interviennent. elle est utilisée pour pouvoir jouer sur des paramètres difficilement modifiables, voir non modifiables, dans les cas réels pour observer leurs influences, tel que la simulation individu-centrée.

-Le travail collaboratif assisté par ordinateur :comme les agents assistants, agents Médiateurs.

-La télématique (internet) :comme les agents "intelligents", agents d'interface.

-Les systèmes multi-capteurs.

-Les jeux vidéo (intelligence des caractères).

2.10 Le Cloud et systèmes Multi-agents

Le cloud computing et les agents sont deux technologies qui peuvent être combinées pour produire des techniques innovantes. Une nouvelle discipline, appelée Cloud Computing à base d'agents est apparait pour fournir des solutions à base d'agents pour améliorer l'utilisation des ressources de Cloud, la gestion des services, la négociation SLA et la composition de services. Les agents autonomes peuvent créer un Cloud plus intelligent dans l'interaction avec les utilisateurs et plus efficace dans l'allocation traitement et stockage.

Les activités de recherche sont effectuées pour mettre en œuvre des solutions à base d'agents efficaces pour le cloud computing. Dans ce cas, il existe trois classes possibles :

-Dans les SaaS (Software as a Service), les agents peuvent être utilisés pour aider à l'approvisionnement intelligent des ressources de base aux applications de l'utilisateur.

-Dans les PaaS (Plateforme as a Service), l'agent peut jouer un rôle dans le déploiement et l'exécution d'environnements de programmation que les développeurs utilisent pour la mise en ceuvre d'application efficace.

-Dans les laaS (Infrastructures as a Service), les agents peuvent être programmés pour optimiser l'utilisation des applications fournies comme des services et de la gestion du sous-jacent infrastructure matériel logiciel en prenant soin de son utilisation efficace et. en même temps, pour le maintien de la qualité de service déclarée [Meroufel, 2016].

2.11 Conclusion

Notre exploration du domaine des Systèmes Multi-agents met en lumière leur ascension remarquable en tant que pilier de l'informatique. De l'intelligence artificielle à la robotique en passant par les systèmes distribués, ces systèmes se sont imposés comme une solution incontournable, répondant aux besoins croissants en interactivité, réactivité et mobilité dans de nombreux domaines d'application.

Dans ce chapitre, nous avons introduit quelques concepts sur les Systèmes Multi-Agents, offrant ainsi un aperçu complet de cette technologie émergente et de son potentiel transformateur dans le paysage informatique.

Dans le chapitre suivant, nous aborderons les concepts relatifs à l'ordonnancement des tâches, en explorant en détail quelques algorithmes utilisés dans ce domaine.

En plus des méthodes classiques comme Min-Min, Round Robin, Max-Min et SJF, nous présenterons également les algorithmes d'ordonnancement basés sur les Systèmes Multi-agents.

Chapitre 3 : Les Algorithmes d'Ordonnancement des Tâches dans un Environnement de Cloud Computing

Chapitre 3

Les Algorithmes d'Ordonnancement des Tâches dans un environnement de Cloud Computing

3.1 Introduction

L'ordonnancement des tâches pour le Cloud Computing représente un défi crucial et complexe dans le domaine de l'informatique moderne. Avec l'essor rapide des technologies de cloud, qui offrent une capacité de traitement quasi illimitée et une flexibilité sans précédent, la gestion efficace des ressources devient essentielle pour maximiser la performance, minimiser les coûts et assurer une qualité de service optimale.

Dans ce chapitre, nous explorerons les principes fondamentaux de l'ordonnancement des tâches, en mettant l'accent sur les algorithmes classiques utilisées pour résoudre les problèmes d'ordonnancement, telles que l'algorithme de Max-Min, l'algorithme de Min-Min, l'algorithme de Round Robin et SJF (Shortest Job First).Ainsi que sur les approches plus avancées basées sur les Systèmes Multi-agents où plusieurs agents autonomes collaborent pour résoudre les problèmes d'ordonnancement. Chaque agent peut représenter une machine ou une tâche, et interagir avec les autres pour optimiser l'ordonnancement global. Nous explorerons des techniques telles que l'algorithme jumper firefly, algorithme de combinaison d'optimisation Adaptative des Semences d'Arbres (ATSO). . ... Enfin, nous procéderons à une comparaison entre quelques algorithmes basés sur les Systèmes Multi-agents, mettant en évidence les avantages et les inconvénients.

3.2 Définition de l'ordonnancement des tâches

L'ordonnancement des tâches dans le Cloud Computing est le processus d'affectation des tâches aux machines physiques (plus précisément les machines virtuelles).Cette attribution joue un rôle très important sur les performances du Datacenter, ainsi que d'autres exigences de l'utilisateur décrites dans le SLA [Baccouche, 1995].

3.3 Les algorithmes de bases

3.3.1 Algorithme Max-Min

Cet algorithme se déroule en deux phases :

Phase 1 : Calcul d'abord le temps d'exécution de chaque tâche sur chaque machine, puis pour chacun d'elle choisir la machine qui traite les tâches en un minimum de temps.

Phase 2 : Parmi toutes les tâches de la méta- tâche, la tâche avec un temps d'exécution maximal est sélectionnée et est affectée à la machine. La tâche est supprimée de la liste de Meta tâches et la procédure continue jusqu'à ce que cette liste devienne vide [Elzeki et al., 2012].

Algorithme 1 : Max-Min

Les Entrées :cloudletlist.size : la liste des tâches, VML : la liste de VM ;
Les Sorties : Mapper chaque tâche à une machine virtuelle ;
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow 1$ **to** N **do**
    $C_{ij} \leftarrow E_{ij} + R_j$;
  end
end
while *toutes les tâches non planifiées ne sont pas épuisées* **do**
  for *chaque tâche non planifiée* **do**
    trouver le temps d'achèvement maximum de la tâche et la machine virtuelle qui l'obtient;
  end
  trouver la tâche t_p avec un temps d'exécution maximum;
  attribuer la tâche t_p à la machine virtuelle qui donne le temps de réalisation minimum;
  supprimer la tâche t_p de l'extraction des tâches non ordonnancées;
  mettre à jour le temps de préparation de la machine qui donne le temps de réalisation maximum;
end

3.3.2 Algorithme Min-Min

C'est le même principe que l'algorithme Max-Min sauf :

Phase 1 : Calculer les temps d'exécution de chaque tâche sur chaque machine, puis la machine qui traite les tâches en un minimum temps d'exécution sera sélectionnée.

Phase2 : Au milieu de toutes les tâches du méta tâche, la tâche qui a un temps d'exécution minimal est sélectionnée et est affectée à la machine. La tâche est supprimée de la liste Meta tâche et la procédure continue jusqu'à ce que cette liste devienne vide [Priyadarsini and Arockiam, 2014].

Algorithme 2 : Min-Min

Les Entrées :cloudletlist : la liste des tâches, VML : la liste de VM ;
Les Sorties : Mapper chaque tâche à une machine virtuelle ;
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow 1$ **to** N **do**
    $C_{ij} \leftarrow E_{ij} + R_j$;
  end
end
while *toutes les tâches non planifiées soient épuisées* **do**
  for *chaque tâche non planifiée* **do**
    trouver le temps minimum d'achèvement de la tâche et la machine virtuelle qui l'obtient;
  end
  trouver la tâche t_p avec l'heure d'achèvement la plus précoce;
  attribuer la tâche t_p à la machine virtuelle qui donne le temps de réalisation minimum;
  supprimer la tâche t_p de l'extraction des tâches non ordonnancées;
  mettre à jour le temps de préparation de la machine qui donne le temps minimum de réalisation;
end

3.3.3 Algorithme Round Robin

Le principe de l'algorithme est simple qui consiste à distribuer de façon équitable les tâches sur les machines virtuelles disponibles, autrement dit le nombre de tâches pour chaque machine virtuelle est le même. L'algorithme Round Robin est implémenté par défaut dans le simulateur CloudSim [Yeboah et al., 2015].

Algorithme 3 : Round-Robin

Les Entrées : cloudletlist : la liste des tâches, VML : la liste de VM ;
Les Sorties : Mapper chaque tâche à une machine virtuelle ;
Nbcl ← cloudletlist.size();
NbVM ← VML.size();
index ← 0;
for $j \leftarrow 0$ **to** *Nbcl* **do**
    cl ← cloudletlist.get(j);
    index ← $(index + 1) \mod NbVM$;
    v ← VML.get(index);
    stagein ← TempsDeTransfert(cl, v, in);
    stageout ← TempsDeTransfert(cl, v, out);
    exec ← TempsDeExcution(cl, v);
    if $cl.AT + stagein + stageout + exec + v.RT \leq cl.DL$ **then**
        EnvoyerLeTravail(cl, v);
        Mise-a-Jour(v);
    end
    else
        Déposer(cl);
        ÉchouerLeTravail;
    end
end

3.3.4 Algorithme Shortest Job First

Le principe de l'algorithme SJF ressemble au FIFO, on choisit d'exécuter la tâche qui sera plus courte, au lieu d'exécuter dans l'ordre d'arrivée. Le problème se pose dans la détermination du temps d'exécution d'une tâche avant de l'exécuter et pour cela il faut se baser sur une estimation [Yeboah et al., 2015].

Algorithme 4 : Shortest Job First

Les Entrées : Liste des tâches : [tâche1, tâche2,..., tâcheN], Longueurs des tâches : [longueur1, longueur2,..., longueurN],Liste des VMS : [VM1, VM2,..., VMN] ;
Les Sorties : Affectations de tâches ;
Trier les tâches par ordre croissant de longueur;
foreach *tâche i dans la liste des tâches* **do**
    Attribuer tâche_i à VM;
end

3.4 Les Algorithmes d'Ordonnancement des Tâches basés sur les Systèmes Multi-agents

3.4.1 Algorithme de Jumper Firefly

Dans cet article, les auteurs présentent une nouvelle méthode basée sur l'Algorithme Jumper Firefly, qui maximise l'Algorithme Jumper Firefly standard en partageant des informations à l'aide d'une table d'état. Il étudie également la situation des fireflies en termes de recherche de la solution appropriée et identifie l'agent le plus inefficace en utilisant l'option de saut.

De plus, l'Algorithme de Jumper Firefly est utilisé pour réduire le temps d'exécution (makespan).

Trois agents sont considérés pour ce projet : agent de demande des tâches permettant de trouver des fournisseurs appropriés disposant des ressources informatiques appropriées pour exécuter les tâches, via agent mappeur. L'agent ordonnanceur assure l'ordonnancement des tâches, pour les demandes de l'gent mappeur et conserve les informations auprès des fournisseurs de services pour une meilleure cartographie. L'agent mappeur connecte les agents et facilite la mise en correspondance des demandes de tâche des utilisateurs avec les ressources informatiques appropriées des fournisseurs. Il peut également être utilisé pour faire face à des événements inattendus [Nithya and Jayapratha, 2014].

Algorithme 5 : Jumper Firefly

Les Entrées : $F(X)$, $X = (X1, X2, X3, ..., Xn)$, m, y, a ;// constantes définies par l'utilisateur
Les Sorties : $I = F(X)$;
for $i \leftarrow 1$ **to** m **do**
  $X_i \leftarrow \text{solution}_i initiale()$ **end**
  Créer une table d'état;
  while *les conditions de résiliation ne sont pas remplies* **do**
    if *une luciole est en danger* **then**
      Mettre la luciole dans une nouvelle position de manière stochastique;
      Tableau d'état de mise à jour;
    end
    $X_{\min} \leftarrow \arg\min_i\{1, ..., m\}(F(X_i))$;
    for $i \leftarrow 1$ **to** m **do**
      for $j \leftarrow 1$ **to** m **do**
        if $F(X_i) < F(X_j)$ **then**
          $d_{ij} \leftarrow \text{Distance}(X_i, X_j)$;
          $\beta \leftarrow \text{attractivité}(I_0, d_{ij})$;
          $X_i \leftarrow (1 - \beta)X_i + \beta X_j + a(\text{aléatoire}() - 1/2)$;
          Tableau d'état de mise à jour;
        end
      end
    end
    $X_{\min} \leftarrow X_{\min} + a(\text{aléatoire}() - 1/2)$;
  end

3.4.2 Mécanisme adaptatif basé sur des agents

Cet article fournit un nouveau mécanisme adaptatif basé sur des agents pour un ordonnancement efficace des tâches. Un nouveau type d'agent appelé agent hôte est introduit pour coordonner les comportements intéressés des agents sans participer à la prise de décision des agents lors de l'ordonnancement des tâches.

- Trois agents sont considérés pour ce projet : Une fois les tâches arrivent ou les ressources disponibles, les agents de tâche ou les agents de ressources enverront des demandes d'adhésion qui incluent les caractéristiques de tâches et des ressources (par exemple, les dates limites des tâches) à l'agent hôte. Lorsque l'agent hôte reçoit des demandes de jointure provenant d'agents des tâches et d'agents de ressources, il ajoute les informations de ces agents expéditeurs à la liste externe de travaux. L'agent hôte classe les agents des tâches et les agents ressources dans les deux listes externes en tenant compte respectivement des caractéristiques des tâches et des ressources. L'agent hôte ajoute de manière adaptative les informations sur les agents des tâches et les agents de ressources à la liste interne. Les agents de la liste interne sont autorisés à entrer dans le groupe interactif. L'agent hôte informe les agents des tâches et les agents de ressources, qui figurent dans la liste interne, d'entrer dans le groupe interactif. Après avoir reçu les informations de l'agent hôte, les agents des tâches et les agents ressources entreront dans le groupe interactif.

Dans le groupe interactif, les agents d'emploi et les agents ressources peuvent interagir les uns avec les autres de manière décentralisée. L'agent de tâche /ressource qui a réussi à parvenir à un accord sur l'allocation des ressources enverra une Demande de terminer à l'agent hôte. Ensuite, l'agent hôte supprime les informations de cet agent De la liste interne et de la liste externe des tâches/ressources.

-Trois algorithmes sont présentés respectivement pour les opérations des agents des tâches, des agents ressources et le processus de coordination de l'agent hôte.

-Opérations d'un agent de tâche : Cet algorithme décrit comment un agent de tâches interagit avec des agents ressources sous la coordination de l'agent hôte.

-Opérations de l'agent ressource : Cet algorithme décrit comment un agent ressource interagit et parvient à un accord avec un agent de tâche sous la coordination de l'agent hôte.

-Le processus de coordination de l'agent hôte : L'algorithme décrit formellement le processus de coordination de l'agent hôte dans le mécanisme [Yang et al., 2021].

Algorithme 6 : Agent de tâche

Les Entrées : Attributs de la ressource r ;
Les Sorties : Accord avec un agent de tâche ;
while *la ressource r est disponible* **do**
  envoyer une demande de jointure à l'agent hôte;
  attendre une réponse de l'agent hôte pour entrer dans le groupe interactif;
  while *entré dans le groupe interactif* **do**
    interagir avec les agents dans le groupe interactif;
    if *l'accord avec un agent de tâche est conclu* **then**
      envoyer une demande d'arrêt à l'agent hôte et quitter le groupe interactif;
      la ressource r devient indisponible;
      return succès;
    end
    else
      aller à l'étape précédente pour interagir avec les agents;
    end
  end
end
while *la ressource r est indisponible* **do**
  utilisée par la tâche assignée;
  if *la tâche assignée est terminée* **then**
    le statut de la ressource r redevient disponible;
  end
end

Algorithme 7 : Agent de ressource

Les Entrées : Attributs de la tâche j ;
Les Sorties : Accord avec un agent de ressources ;
while *la tâche j est publiée* **do**
  Envoyer une demande de rejoindre à l'agent hôte;
  if *le temps actuel* $\leq t_d$ **then**
    Attendre une réponse de l'agent hôte pour entrer dans le groupe interactif;
  end
  else
    Envoyer une demande de départ à l'agent hôte;
    return un échec;
  end
  while *entré dans le groupe interactif* **do**
    Rechercher des agents de ressources dans le groupe interactif et interagir avec eux;
    if *l'accord avec un agent de ressources est complété* **then**
      Envoyer une demande de départ à l'agent hôte et quitter le groupe interactif;
      return un succès;
    end
    else if *le temps actuel* $\geq t_d$ **then**
      Envoyer une demande de départ à l'agent hôte et quitter le groupe interactif;
      return un échec;
    end
  end
end

Algorithme 8 :Le processus de coordination de l'agent hôte

Les Entrées : Demandes d'adhésion et demandes de départ des agents des tâches et des agents de ressources ;
Les Sorties : Adaptation du groupe interactif ;
while *recevoir une demande d'adhésion d'un agent de tâche ou d'un agent de ressource* **do**
  ajouter les informations de l'agent expéditeur à la liste externe des tâches `elj` ou à la liste externe des ressources `elr`;
end
while *la liste externe des tâches* `elj` *n'est pas vide* **do**
  trier les agents de tâche dans l'ordre;
end
while *la liste externe des ressources* `elr` *n'est pas vide* **do**
  trier les agents de ressources dans l'ordre;
end
while *le nombre d'agents de tâche dans le groupe interactif est inférieur à* m_j **do**
  ajouter à la liste interne l'agent de tâche ayant la priorité la plus élevée dans `elj`;
  informer l'agent de tâche qu'il peut entrer dans le groupe interactif;
end
prendre en compte la demande des agents de tâches dans le groupe interactif pour chaque type de ressources `dmr_t`;
while *le nombre d'un certain type d'agents de ressources dans le groupe interactif* nr_{rt} *est inférieur à la demande des agents de tâches* `dmr_t` **do**
  ajouter l'agent de ressource de ce type ayant la priorité la plus élevée dans `elr` à la liste interne;
  informer l'agent de ressource d'entrer dans le groupe interactif;
end
while *recevoir une demande d'arrêt d'un agent de tâche ou d'un agent de ressource* **do**
  supprimer les informations de l'agent expéditeur de la liste externe des tâches/ressources et de la liste interne;
end

3.4.3 Algorithmes d'ordonnancement dynamiques basées sur des agents nommés ANGEL

Dans cet article les auteurs propose un algorithme d'ordonnancement dynamique en temps réel basé sur des agents nommé ANGEL. Cet algorithme repose sur des agents, qui sont des entités autonomes capables de prendre des décisions et d'interagir avec leur environnement pour atteindre des objectifs spécifiques.

Trois agents concerné pour ce projet : l'agent de gestion (Gestionnaire) joue un rôle central en initiant le contact entre les tâches à réaliser et les machines virtuelles disponibles. Il établit des contrats avec les agents de machines virtuelles pour l'attribution des tâches. Une fois les contrats conclus, il informe les agents concernés et calcule les valeurs d'enchères pour chaque tâche. Par la suite, l'agent de gestion procède à la création des agents de tâches et assure la transmission

des informations nécessaires aux agents de machines virtuelles. Il coordonne également les offres et les réponses entre ces différents acteurs, en ajustant les correspondances entre les tâches et les machines virtuelles en fonction des offres reçues. Une fois les affectations finales déterminées, il notifie les agents de machines virtuelles et finalise les contrats relatifs aux tâches attribuées. Ils assurent également la communication des informations sur les tâches assignées à l'agent de gestion, facilitant ainsi le processus de coordination. Quant aux agents de machines virtuelles, ils reçoivent les informations sur les tâches à exécuter de la part de l'agent de gestion. Une fois les tâches attribuées, ils informent à leur tour l'agent de gestion de leur disponibilité et envoient les détails des tâches assignées.

Cinq algorithmes sont présentés respectivement pour les agents gestionnaires, les agents de tâches, les agents VM,et quelques fonctions

- Algorithme de l'agent gestionnaire : permettre de coordonner la planification dynamique des tâches et des machines virtuelles (Vms) dans un environnement où les tâches arrivent simultanément.(voir algorithme 9).

- Algorithme les agents de tâches : permettre à un agent de tâche (Task Agent) de choisir une machine virtuelle (VM) appropriée pour exécuter la tâche associée, en suivant un processus d'annonce et d'enchères.

- Algorithme les agents VM : permettre à un agent de machine virtuelle (VM Agent) de sélectionner efficacement une tâche appropriée pour exécuter sur la VM associée.

- Algorithme La fonction scaleUpResource () : permettre l'extension des ressources disponibles en créant de nouvelles machines virtuelles (Vms) avec une puissance de traitement supplémentaire.

- Algorithme d'ordonnancement Greedy : permettre de fournir une méthode de prise de décision pour l'allocation de tâches aux machines virtuelles (VMs) disponibles [Zhu et al., 2015].

-Nous présentons seulement le pseudo-code de l'agent gestionnaire

Algorithme 9 : Agent gestionnaire

$T_{waiting} \leftarrow$ les tâches qui arrivent au même instant;
foreach t_i *dans* $T_{waiting}$ **do**
  $V_i \leftarrow$ les agents de VM qui satisfont les exigences de base de t_i;
  Envoyer les V_i à l'agent de tâche t_i;
end
while $T_{waiting} \neq \emptyset$ **do**
  Les agents de tâche commencent la phase d'annonce d'enchère avant;
  Les agents de VM commencent la phase d'annonce d'enchère arrière;
end

3.4.4 Algorithmes d'optimisation adaptative basées sur des semences d'arbres

L'algorithme est une combinaison d'optimisation adaptative des semences d'arbres (ATSO) multi-agents, cet algorithme créer un système de planification et d'allocation de ressources hautement efficace et adaptable.

- Quatre agents sont considérés pour ce projet : l'agent utilisateur récupère les requêtes des différents utilisateurs du Cloud. La tâche contient des informations comme (taux de requêtes, type, taille, etc.). Ensuite, les requêtes collectées sont directement transférées à l'agent de surveillance.

L'agent de surveillance doit récupérer les requêtes de l'agent utilisateur et les informations sur les ressources du centre de données.tel que le taux de charge des ressources (charge CPU et mémoire). L'agent de surveillance transmet les informations collectées sur les demandes d'un utilisateur et les informations de charge des ressources à l'agent d'ordonnancement. L'agent ordonnanceur alloue de manière optimale la tâche aux ressources tout en tenant compte du temps, du coût et de l'utilisation des ressources. Après le processus d'ordonnancement, les tâches planifiées sont confiées à l'agent d'exécution. L'exécuteur alloue la tâche à chaque VM [Arravinth and Manjula, 2022].

Algorithme 10 : Agent d'interface

Les Entrées : Recevoir une demande de l'utilisateur ;
Les Sorties : Demande complète de l'utilisateur :
if *Générer un identifiant de demande pour chaque tâche* **then**
  Appeler l'agent de surveillance des ressources (α_i);
end

Algorithme 11 : Agent de surveillance des ressources

Les Entrées :Tâche des utilisateurs et liste des ressources ;
Les Sorties :Détails de la tâche et des ressources ;
if *Générer une table de ressources* **then**
  Appeler l'agent d'ordonnancement (γ_i);
end

Algorithme 12 :Agent d'ordonnancement

Les Entrées : Table des ressources ;
Les Sorties :Tâche planifiée ;
if *Planifier la tâche appropriée avec les ressources disponibles* **then**
  Appeler l'agent exécuteur (E_i);
end

3.4.5 Algorithmes basés sur des BDI agents

Dans cet article les auteurs ont proposé deux algorithmes basés sur des agents pour mettre en œuvre les processus d'Ordonnancement et de Ré-Ordonnancement des tâches, l'objectif est de fournir une approche flexible, adaptable et efficace pour gérer les ressources et les charges de tâche dans un environnement dynamique.

- Trois agents proposés pour ce projet : Les agents utilisateurs recueillent les exigences des tâches auprès des utilisateurs, tandis que les agents hôtes collectent des informations VM en temps réel. Les agents hôtes communiquent la disponibilité des VM à l'agent de supervision, qui les priorise en fonction de leur disponibilité. Lorsque les utilisateurs demandent des ressources, les agents utilisateurs transmettent les détails de la tâche et les délais à l'agent de supervision, qui recommande des VM disponibles à l'aide d'un algorithme de recommandation asynchrone (ARA). Les agents utilisateurs interagissent ensuite avec les agents hôtes pour demander l'utilisation des VM, recevant des propositions pour les temps d'exécution des tâches. Les agents utilisateurs sélec-

tionnent la meilleure proposition en fonction de l'heuristique du Temps de Terminaison Minimum, l'acceptent et mettent à jour l'agent hôte. S'ils ne reçoivent aucune proposition, les agents utilisateurs réessaient périodiquement. Une fois les accords conclus, les agents hôtes mettent à jour les informations VM auprès de l'agent de supervision [Yang et al., 2024].

Algorithme 13 : d'Ordonnancement

Entrée : $U = \{u_1, \ldots, u_n\}, DC = \{H_1, \ldots, H_i\}$
Sortie : Correspondance entre utilisateurs et VMs
Initialisation :

$B_n \leftarrow \{u_1, \ldots, u_n\}$; `// uan définit les informations du` n`-ième utilisateur comme sa croyance`
$B_i \leftarrow \{H_1 = \mathrm{vm}_{1i}, \ldots, H_k = \mathrm{vm}_{ki}\}$; `// hai définit les VMs dans` H_i `comme sa croyance`
Toutes les VMs définissent les états $\mathrm{state}_{ki}(\tau) =$ ÉTAT_PRÊT
$B \leftarrow \{H_1, \ldots, H_i\}$; `// sa définit les VMs dans les hôtes comme sa croyance`

foreach vm_i^k **do**
  if *les informations de* vm_i^k *changent* **then**
    hai déclenche $i \in I_i$ pour se synchroniser avec sa;
    sa trie les VMs en temps opportun, plus tôt est at_i^k, plus la priorité est élevée;
  end
end

while *uan détecte des tâches non attribuées* **do**
  uan déclenche $d \in D_n$ pour correspondre à une VM appropriée;
  uan envoie T_n et D_n à *sa*;
  sa recherche des VMs qui répondent à T_n, sous réserve de : $\mathrm{ram}_n^p \geq \max(T_n(\mathrm{ram}_n^p)) \wedge \mathrm{sc}_n^p \geq \max(T_n(\mathrm{sc}_n^p)) \wedge \mathrm{bw}_n^p \geq \max(T_n(\mathrm{bw}_n^p)) \wedge \mathrm{at}_i^k(\tau) + \frac{\sum_1^p \mathrm{wl}_n^p}{\mathrm{cpu}_i^k} \leq D_n$;
  repeat
    foreach vm_i^k **do**
      sa vérifie l'état de chaque vm_i^k;
      if *l'état de* $vm_i^k(\tau) == \acute{E}TAT\_PR\hat{E}T$ **then**
        sa ajoute vm_i^k au message msg;
        Définir l'état de $\mathrm{vm}_i^k(\tau)$ comme ÉTAT_OCCUPÉ;
      end
    end
  until *la taille de la proposition* $\geq \theta \in \mathbb{N}^+$ *ou plus de VMs*;
  sa envoie la proposition à uan;
  if *uan ne reçoit pas de proposition* **then**
    uan envoie périodiquement T_n et D_n à *sa*;
  end
  else if *uan parvient à un accord avec un hai* **then**
    uan et *hai* synchronisent les informations avec *sa*;
    Définir l'état des VMs comme ÉTAT_PRÊT;
  end
end

Algorithme 14 : Ré-Ordonnancement

Entrée : $U, DC, E = \{e_1, \ldots, e_j\}$
Sortie : Résolution des événements incertains E
if $\exists e_j \in U$ **then**
  uan détecte les changements dans B_n, $B_n = \{u_n \rightarrow u'_n = [T'_n, D'_n]\}$;
  if *l'horaire actuel est invalide* **then**
    uan déclenche D_n pour maximiser le taux de réussite des tâches;
    uan délibère sur les intentions I_n, $I_n = [i_1 \preceq i_2 \preceq i_3]$;
    repeat
      uan traite les intentions par priorité;
      // Que i_1 désigne le changement des créneaux horaires dans le même vm_i^k
      // Que i_2 désigne le changement d'une VM vm_i^k dans le même hôte H_i
      // Que i_3 désigne le changement d'une VM $\mathrm{vm}_i'^{k'}$ dans un autre H'_i
    until *résoudre le* e_j;
  end
end
else if $\exists e_j \in vm_i^k$ **then**
  hai détecte les changements dans vm_i^k, $B_i = \{n\ \mathrm{vm}_i^k \rightarrow (\mathrm{vm}_i^k)'\}$;
  hai déclenche D_i pour maximiser le taux de réussite des tâches;
  forall u_n *sur* vm_i^k **do**
    if $(vm_i^k)'$ *ne peut pas satisfaire* $T_n \parallel D_n$ **then**
      hai recherche une VM disponible pour u_n dans H_i;
      if $\exists vm_i^{(k^*)}$ *satisfait* u_n **then**
        hai génère une proposition et informe uan;
        uan établit le nouveau contrat avec hai;
      end
      else if $\nexists vm_i^{(k^*)}$ *satisfait* u_n **then**
        hai informe uan;
        repeat
          uan interagit avec *sa* pour une nouvelle VM;
        until *résoudre le* e_j;
      end
    end
  end
end

3.5 Comparaison entre les différents algorithmes d'ordonnancement des tâches basés sur systèmes multi-agents

Après avoir étudié les algorithmes d'ordonnancement des tâches basés sur les systèmes multi-agents, nous proposons une comparaison détaillée de ces algorithmes sous forme de tableau. Ce tableau met en lumière l'objectif de chaque algorithme et les avantages, inconvénients :

Algorithmes	Objectif	Paramètres	outil	Avantages	Inconvénients
Algorithme de jumper firefly.	-Minimisele temps de makespan. -Trouver les solutions optimales globales.	- Makespan. - Population size.	Cloudsim.	-Optimise efficacement la planification des tâches. -Réduisant les délais d'exécution des tâches -Améliorant les performances globale de système.	-La conception, l'implémentation et la gestion de cette approche hybride nécessitent une expertise considérable pour assurer une performance optimale.
Algorithme d'optimisation adaptative basé sur des semences d'arbres (ATSO).	-Attribuez la tâche de manière optimale aux ressources.	- Makespan. -Coût. - Utilisation des ressources.	Cloudsim.	-L'ATSO offre une approche puissante et adaptable pour la planification efficace des tâches dans les environnements de cloud computing.	-L'ATSO peut nécessiter une expertise approfondie pour sa conception et son paramétrage. - L'ATSO peut demander des ressources computationnelles importantes, ce qui peut entraîner des coûts élevés dans les environnements de cloud computing
Algorithme d'ordonnancement dynamique basé sur des agents nommé ANGEL.	- Solution d'ordonnancement dynamique en temps réel. -Maximiser l'efficacité de la distribution des ressources dans un environnement informatique dynamique et évolutif.	-Taille du bain Inférieur, intervalTimeLower, thêta.	Cloudsim.	- Optimiser plusieurs objectifs simultanément, tels que la minimisation du coût, la maximisation de l'efficacité et la garantie des délais de réalisation dans un environnement temps réel.	- Entraîner une augmentation de la complexité du système et des coûts de développement.

Algorithmes basés sur des BDI agents.	-Fournir une approche flexible, adaptable et efficace pour gérer les ressources et les charges de travail dans un environnement dynamique.	- Makespan. -Deadline (Dn).	JADEX Cloudsim.	-Offrent une approche flexible et intelligente pour l'ordonnancement et la gestion des ressources dans le cloud computing, en permettant une adaptation dynamique aux conditions changeantes et une optimisation des performances du système.	-L'algorithme basé sur des agents peuvent être complexes en raison de la coordination nécessaire entre les agents et des interactions multiples dans un environnement distribué. -Le système plus difficile à comprendre, à déboguer et à maintenir.
Mécanisme adaptatif basé sur des agents.	- Définir de manière précise et complète les interactions entre les différents agents impliqués dans la gestion des ressources et l'exécution des tâches.	- Deadline, la capacité de la ressource.	-Eclipse IDE using the JADE.	- Offrent une adaptabilité dynamique, une optimisation des performances et une haute disponibilité du système.	- Entraîner des retards dans l'exécution des tâches. -Affecter la performance globale du système de manière significative.

TABLE 3.1 – Comparaison entre les différents algorithmes d'ordonnancement des tâches basés sur systèmes multi-agents.

3.6 Conclusion

L'ordonnancement des tâches se révèle être un pilier essentiel, permettant d'optimiser la productivité dans le cloud computing. Ce chapitre a mis en lumière les principes fondamentaux ainsi que les algorithmes d'ordonnancement des tâches, allant des techniques classiques telles que Max-Min et Round Robin, aux approches plus sophistiquées comme les systèmes multi-agents avec des algorithmes tels que Jumper Firefly et ATSO.

Dans le chapitre suivant, nous allons introduit notre propre algorithme d'ordonnancement des tâches basé sur un système multi-agent, visant à résoudre les problèmes inhérents des algorithmes centralisés.

Chapitre 4 : Implémentation et Simulations

Chapitre 4

Implémentation et Simulations

4.1 Introduction

Dans le contexte des Clouds, la plupart des algorithmes d'ordonnancement des tâches adoptent une approche centralisée. Bien qu'ils soient efficaces dans de nombreux cas, ils peuvent ne pas réagir de manière optimale sur des plates-formes à grande échelle, ce qui peut compromettre les niveaux de qualité de service. Une solution à ce défi consiste à décentraliser l'ordonnancement.

Diverses méthodes ont été avancées pour décentraliser le processus d'ordonnancement, parmi lesquelles l'approche multi-agents. Cette dernière implique le déploiement d'agents logiciels sur l'ensemble des nœuds de l'infrastructure à gérer, permettant ainsi la répartition du travail d'ordonnancement entre ces agents.

Nous présentons dans ce travail, une approche distribuée pour l'ordonnancement des tâches, fondée sur une architecture multi-agents. Nous sommes particulièrement concentrés sur la phase cruciale de prise de décision inhérente à l'ordonnancement des tâches, qui implique la nécessité de faire plusieurs choix. Cette phase décisionnelle est conçue en suivant une architecture dédiée aux agents intelligents, ce qui permet une gestion proactive et adaptative des processus d'ordonnancement.

Notre travail se concentre principalement sur la conception, réalisation et simulation d'un algorithme d'ordonnancement de tâches dans le Cloud computing, dans le but de le comparer avec des algorithmes de base. L'objectif principal de cet algorithme est d'optimiser le temps d'exécution des tâches.

4.2 Outils et environnement de développement

Le travail proposé dans ce mémoire a été implémenté et testé dans un environnement possédant les caractéristiques suivantes :

-Une machine avec un processeur Intel (R) Core (TM) i5-5300U CPU@ 2.30GHz, une vitesse de 2.30 GHz et une capacité mémoire de 4GB.

-Le simulateur CloudSim est sous Windows 8 de 64 bits.

-Le simulateur CloudSim version 3.0.3.

-Le langage de programmation Java.

-L'IDE NetBeans.

4.2.1 JDK (Java Développement Kit)

Le Java Développement Kit, communément appelé JDK, est le kit de développement de base que propose gratuitement la société Oracle. Le Kit de développement comprend plusieurs outils, parmi lesquels :

-Javac :le compilateur Java
-Java :un interpréteur d'applications (machine virtuelle)
-Applet viewer :un interpréteur d'applets
-Jdb :un débogueur
-Javap :un décompilateur, pour revenir du bytecode au code source
-Javadoc :un générateur de documentation
-Jar :l'éditeur d'archives Java [Bouteldja and Bakdi, 2019].

4.2.2 Langage de programmation JAVA

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par Gosling et Patrick Naughton, employés de Sun Microsystems. Il permet une programmation orientée objet, modulaire et reprend une syntaxe très proche de celle du langage C.

Outre son orientation objet, le langage Java a l'avantage d'être modulaire (on peut écrire des portions de code génériques, c'est-à-dire utilisables par plusieurs applications), rigoureux (la plupart des erreurs se produisent à la compilation et non à l'exécution) et portable (un même programme compilé peut s'exécuter sur différents environnements à savoir sur plusieurs systèmes d'exploitation tels qu'UNIX, Windows, Mac OS ou Linux). En contrepartie, les applications Java ont le défaut d'être plus lentes à l'exécution que des applications programmées en C par exemple [Kamilia and Chahinas, 2015].

4.2.3 NetBeans

Est un projet open source fondé par Sun Microsystems. L'IDE NetBeans est un environnement de développement permettant d'écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'IDE NetBeans. L'IDE NetBeans est un produit gratuit, sans aucune restriction quant à son usage [Benamar and Ayachi, 2017].

4.2.4 La Plateforme JADE

JADE (Java Agent Développement Framework) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie), il fournit un environnement de développement et d'exécution des systèmes multi-agents compatibles avec les standards FIPA [Benmerzoug, 2010].

4.2.5 Modelio

C'est un environnement de modélisation open source, il offre plusieurs fonctionnalités pour les développeurs de logiciels, les analystes et les concepteurs.

4.2.6 CloudSim

C'est un outil de simulation extensible complet pour la modélisation et simulation du Cloud Computing. Il permet d'étendre et de définir des politiques pour tous les systèmes Composants. Il prend en charge la modélisation du système et du comportement comme les centres de données, les machines virtuelles et l'approvisionnement des ressources. Il est considéré comme l'outil de simulation Cloud le plus populaire [Khalil et al., 2017].

4.3 Classes de CloudSim

Le simulateur Cloud Sim est constitué de plusieurs classes, comme illustré dans la figure 4.1 Parmi les classes fondamentales qui constituent les blocs constitutifs du simulateur CloudSim, nous pouvons citer [Marlene and Ernest, 2017] :

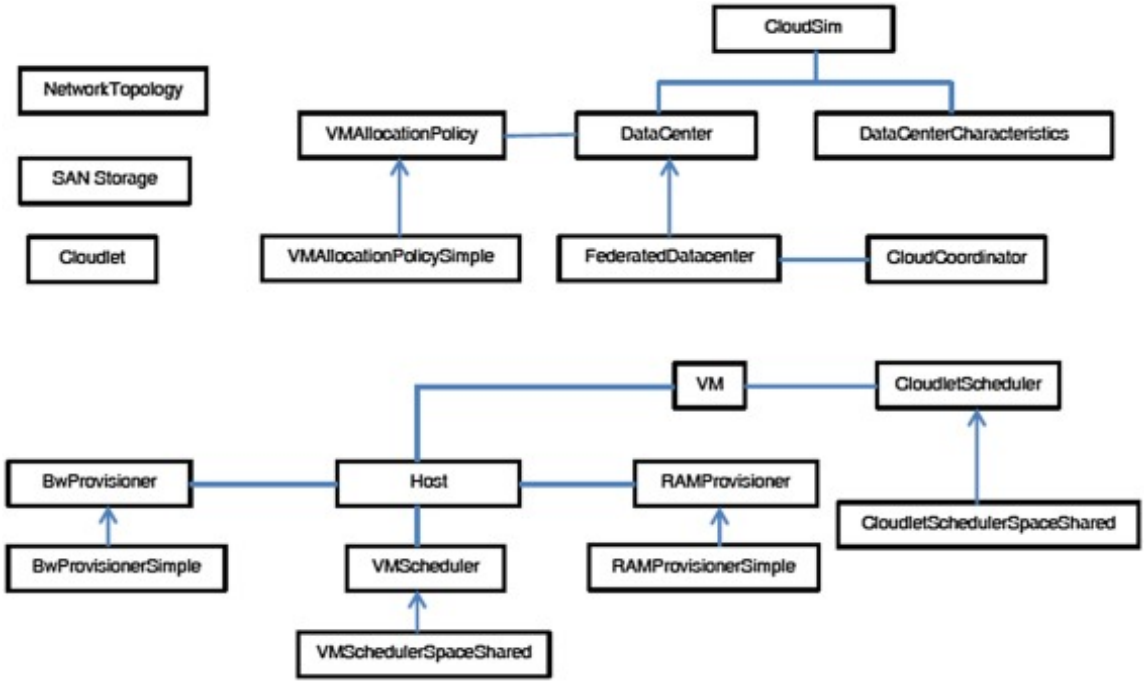

FIGURE 4.1 – Diagramme de classe toolkit CloudSim
[Rahmani and Otmani, 2021]

4.3.1 Cloudlet

C'est une classe qui représente une tâche. Elle modélise les services d'application du Cloud (comme la livraison, réseaux sociaux et les sites d'affaires). Cette classe peut aussi être étendue pour supporter la modélisation des performances et d'autres paramètres de composition pour les applications telles que les transactions dans les applications orientées bases de données (Oracle, SQL).

4.3.2 Datacenter

Cette classe modélise l'infrastructure du noyau (matériel, logiciel) offert par des fournisseurs de service dans un environnement de Cloud Computing. Il encapsule un ensemble de machines de calcul qui peuvent être homogènes ou hétérogènes en ce qui concerne leur configurations de ressources (mémoire , noyau ,capacité et stockage) .En outre ,chaque composant du Datacenter instancie un composant généralisé d'approvisionnement en ressources qui implémente un ensemble de politiques d'allocation de bande passante, de mémoire et des dispositif de stockage.

4.3.3 DataCentreBroker

Cette classe modéliser le courtier, qui est responsable de la médiation entre les utilisateurs et les prestataires de service selon les conditions de QoS des utilisateurs et il déploie les tâche de service à travers les Clouds. Le Broker agissant au nom des utilisateurs identifie les prestataires de service appropriés du cloud par le service d'information du cloud CIS (cloud information services) en négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs.

4.3.4 Machine virtuelle (VM)

Cette classe modéliser une instance de machine virtuelle, qui est géré et hébergé pendant son cycle de vie par le composant Cloud Host. Chaque composant de VM a accès à un composant qui stocke les caractéristiques liées à elle telles que : l'accès mémoire, le processeur, la capacité de stockage et les politiques de provisionnement interne de la machine virtuelle.

4.3.5 Host

Cette classe représente un serveur informatique physique dans un Cloud. Le Host exécute des actions liées à la gestion des machines virtuelles et a une politique définie pour l'approvisionnement mémoire et bande passante, ainsi que d'une politique de répartition des PE (Processeur Element) à des machines virtuelles. Un hôte est associé à un Datacenter. Il peut héberger un ou plusieurs VMs.

4.4 Architecture globale de notre proposition

Notre contribution dans le cadre de ce PFE est la proposition d'un algorithme d'ordonnancement des tâches indépandantes sans priorité basé sur les systèmes multi agents.

Notre proposition (AMA) est constituée de deux types d'agents (agent Globale et des agents hôtes) qui coopèrent entre eux pour ordonnancer les tâches VMs, chaque agent hôte est hébergé sur une machine physique .

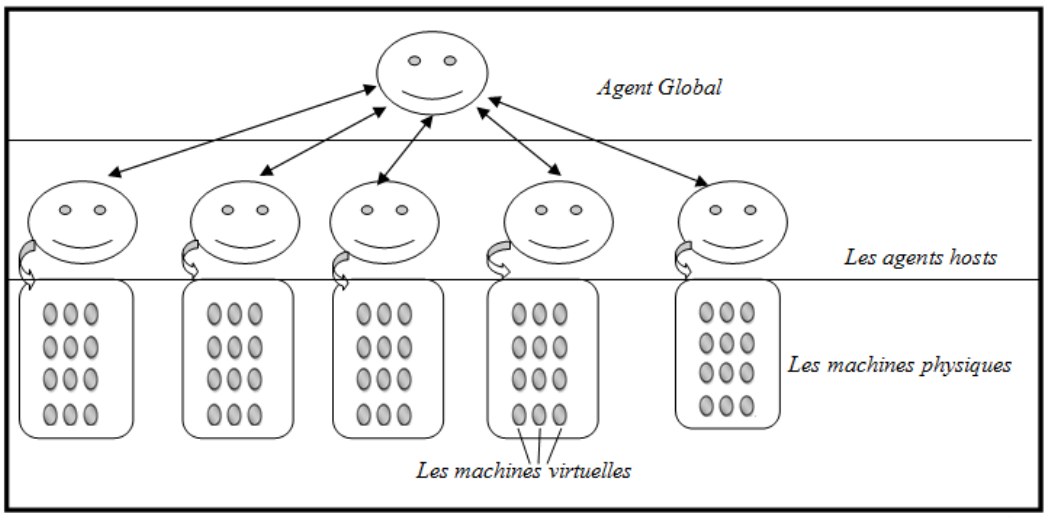

FIGURE 4.2 – Architecture globale de notre proposition

Agent Global :

-Lire le fichier de caractéristiques des cloudletes (tâches) (générer au moment de la 1 ère simulation)

-créer les cloudletes en fonction des caractéristiques.

-créer les agents hôtes.

-Diviser la liste des cloudlets en sous-listes, chaque sous-liste correspondant à un hôte spécifique.

-Distribuer les sous-listes aux agents d'hôtes.

-Recevoir les temps d'exécution maximum des agents hôtes.

-Afficher le temps d'exécution maximum reçu.

Agent hôte :

- Lire son propre fichier de caractéristiques des vms (générer au moment de la 1 ère simulation)
- Créer les vms a partir de fichier.
- Reçoit une liste de cloudlets à traiter provenant de l'agent globale.
- Ordonnancer les tâches par l'algorithme de base (Max-Min).
- Calcule et afficher le temps d'exécution : Représente le temps nécessaire pour une tâche pour qu'elle s'exécute au niveau de la machine virtuelle.Nous utilisons cette formule :

Le temps d'exécution=Cloudletlength/(VM MIPS * VM PEsNumber) Où :

— Cloudletlength : signifie la taille de la tâche.

— VM PEsNumber : signifie le nombre de coeur de CPU occupé par la machine virtuelle.

— VM MIPS : est la vitesse du processeur de la machine virtuelle [Calheiros et al., 2011].

- Prendre le temps maximum d'exécution parmi toutes les Vms et envoie ce résultat à l'agent global.

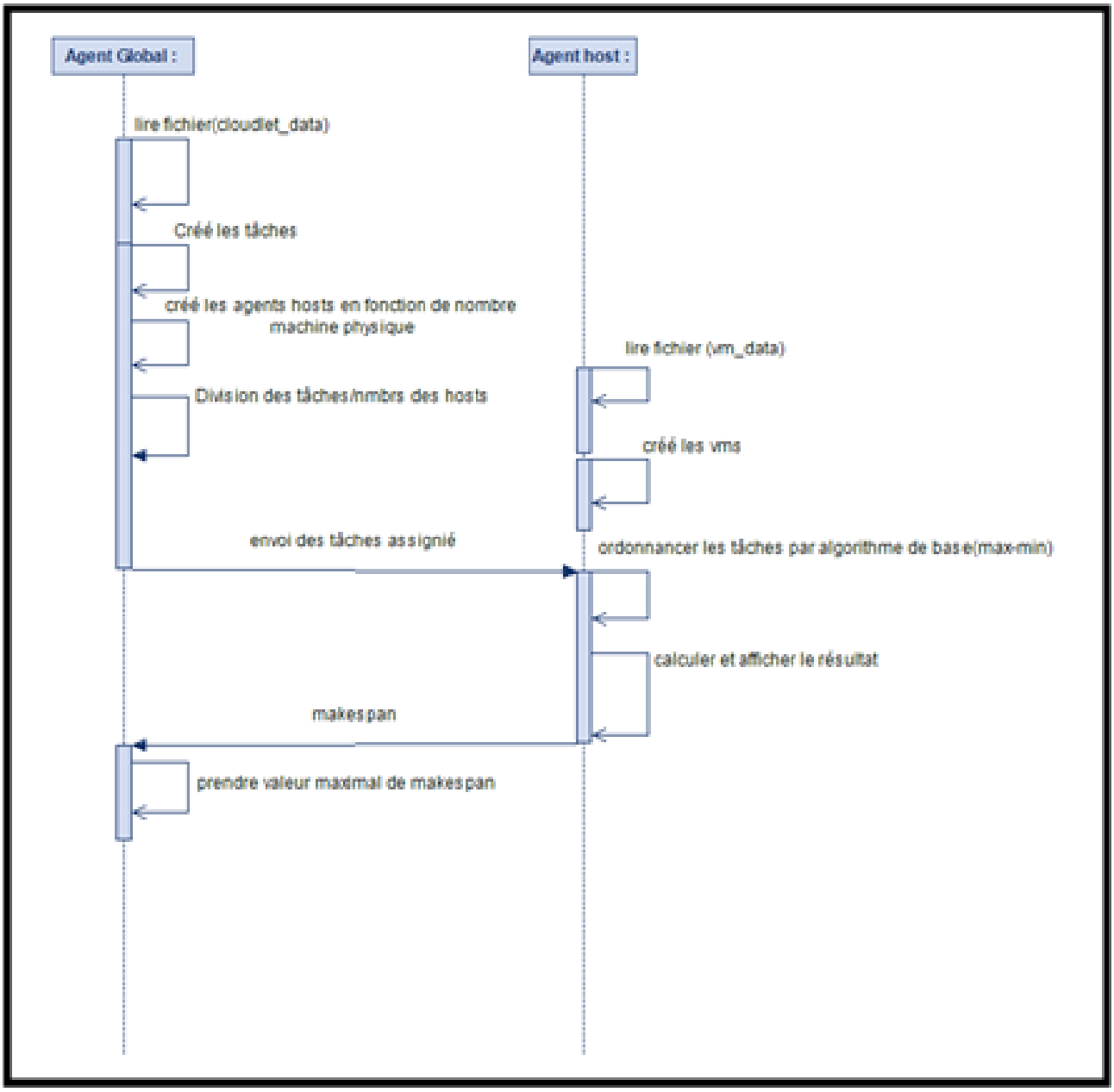

FIGURE 4.3 – Diagramme de communication entre les agents

Enregistrement et la lecture de fichier :

Après la simulation de l'algorithme Round Robin, nous avons sauvegardé des fichiers contenant les caractéristiques des machines virtuelles (VM) et des cloudlets, afin que nous puissions l'utiliser dans d'autres simulations d'algorithmes(Min-Min,SJF,AMA. . ..).

L'objectif de cette sauvegarde c'est pour travailler dans le même environnement de simulation et par conséquent la possibilité de comparaison des résultats entre les algorithmes.

De plus, on a divisé le fichier de VM en 10 sous-listes afin que chaque agent hôte puisse lire son propre fichier.

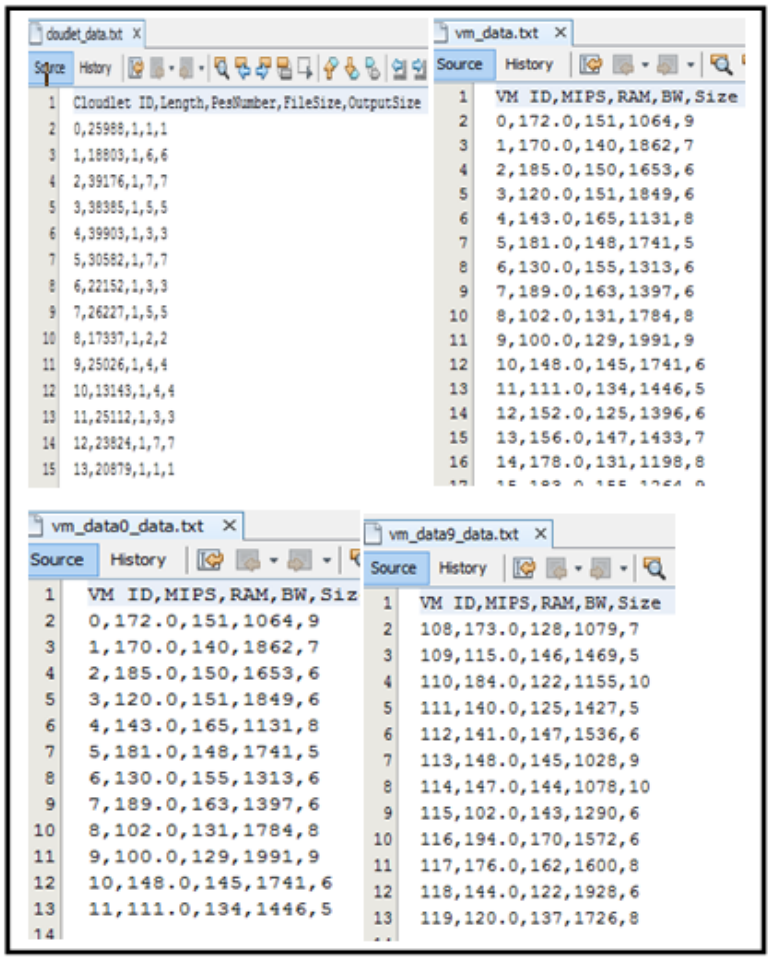

FIGURE 4.4 – Fichiers des caractéristiques des tâches et VM.

Pseudo code de l'approche proposée :

l'algorithme ci-dessous présente notre travail :

Algorithme 15 :Agent Global

Les Entrées : Liste de cloudlets ;`cloudletList`, Liste d'hôtes `hostList`
Les Sorties :`maxExecutionTimeReceived;`
Début
Créer le centre de données;
Créer les hôtes;
Créer le broker;
Lire(cloudlets-data);
Créer les cloudlets à partir du fichier;
Obtenir le nombre total de tâches à partir du fichier cloudlets-data;
for $i \leftarrow 0$ **to** *HOSTS_NUMBER* **do**
  Diviser(totalTasks / HOSTS_NUMBER) ;; `// Diviser en sous-listes`
end
for $i \leftarrow 0$ **to** *HOSTS_NUMBER* **do**
  Créer un nouvel agent ("AgentHôte" + i) ;
  Object[] args = new Object[]subLists.get(i) ;
  Démarrer AgentHôte ;;
end
Ajouter CyclicBehaviour;
while *Recevoir maxExecutionTime de chaque AgentHôte* **do**
  Afficher maxExecutionTimeReceived =
   Math.max(maxExecutionTimeReceived, maxExecutionTime);
end
Fin

Algorithme 16 :Agent Hôte

Les Entrées : Liste de VMs `vmList`, Liste de cloudlets `cloudletList` ;
Les Sorties : Mapper chaque tâche à une machine virtuelle ;
Début
Lire(vm-data);
Créer les VMs à partir du fichier;
Recevoir la sous-liste de cloudlets;
Ordonnancer les cloudlets en ordre décroissant;
Ordonnancer les VMs en ordre décroissant;
for $i \leftarrow 0$ **to** *cloudletList.size()* **do**
  cloudlet.setVmId(vm.getId());
  vm.getCloudletScheduler().cloudletSubmit(cloudlet);
end
Afficher les résultats des hôtes;
Envoyer `maxExecutionTime` à l'agent global;
Fin

4.5 Résultats et analyses

Tableaux des Paramètres de simulation :

Paramètres	Valeurs
Workload(MI)	[10000-40000]
RAM (M)	[800-1200]
Storage (G)	[1-8]
Bandwidth (Mb/s)	[100-500]

TABLE 4.1 – Configuration des VMs

Paramètres	Valeurs
Workload(MI)	200000
RAM (M)	10000
Storage (G)	1000000
Bandwidth (Mb/s)	25000

TABLE 4.2 – Configuration des hôtes

Paramètres	Valeurs
Workload(MI)	[10000-40000]
RAM (M)	[800-1200]
Storage (G)	[1-8]
Bandwidth (Mb/s)	[100-500]

TABLE 4.3 – Configuration des Tâches

Paramètres	Valeurs
Number of Hosts	10
Number of Users	100
Number of VMs in a Host	12
Number of Tasks in a User	6
Number of data center	1

TABLE 4.4 – Configuration de base

Simulation :
-Interface utilisateur graphique (GUI) jade :

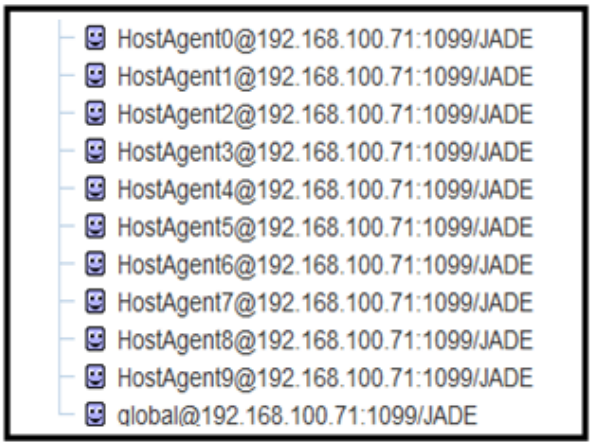

FIGURE 4.5 – Interface utilisateur graphique (GUI) jade.

-Aprés chaque simulation, le résultat est affiché dans la console qui contient les informations suivantes : host , l'ID de la tâche , ID de Vm , le statut de la tâche, le temps d'exécution de la tâche et le temps de début et la fin de la tâche.

HOST	Cldt ID	VM ID	STATUS	cldt Length	VM MIPS	Time	Start Time	Finish Time
HostAgent1@192.168.100.71:1099/JADE	117	5	SUCCESS	39690	194.0	204,59	0,1	204,69
HostAgent1@192.168.100.71:1099/JADE	66	7	SUCCESS	39374	189.0	208,33	0,1	208,43
HostAgent1@192.168.100.71:1099/JADE	81	3	SUCCESS	39111	183.0	213,72	0,1	213,82
HostAgent1@192.168.100.71:1099/JADE	94	2	SUCCESS	38315	178.0	215,25	0,1	215,35
HostAgent1@192.168.100.71:1099/JADE	67	4	SUCCESS	37167	158.0	235,23	0,1	235,33
HostAgent1@192.168.100.71:1099/JADE	97	1	SUCCESS	36853	156.0	236,24	0,1	236,34
HostAgent1@192.168.100.71:1099/JADE	112	0	SUCCESS	36539	152.0	240,38	0,1	240,48
HostAgent1@192.168.100.71:1099/JADE	72	11	SUCCESS	36295	150.0	241,97	0,1	242,07
HostAgent1@192.168.100.71:1099/JADE	109	6	SUCCESS	36082	143.0	252.32	0.1	252.42

FIGURE 4.6 – Simulation cloudsim

-Après simulation chaque agent envoi (maxExecutionTime) à l'agent global :

```
global@192.168.1.6:1099/JADE: Received max execution time 1045.1792079207921 seconds from HostAgent7
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1175.4942307692309 seconds from HostAgent4
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1039.4162393162392 seconds from HostAgent1
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1006.18547008547 seconds from HostAgent2
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1063.3347826086956 seconds from HostAgent6
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1079.7605504587154 seconds from HostAgent3
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 1053.570588235294 seconds from HostAgent9
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
global@192.168.1.6:1099/JADE: Received max execution time 989.4039215686274 seconds from HostAgent0
global@192.168.1.6:1099/JADE: The maximum execution time received is 1199.786274509804 seconds.
```

FIGURE 4.7 – Communication des agents

Résultat du Makespan :

Dans cette section, nous allons élaborer une comparaison en terme du makespan entre notre algorithme Multi-agent (AMA) et les algorithmes RR, SJF, Max-Min, Min-Min. Le tableau suivant indique les résultats obtenus par variation du nombre de tâches de 100 à 600.

Algo/nmbr des taches	RR	Max-Min	Min-Min	sjf	AMA
100 Tâches	338,6	200,53	333,17	384,41	223,54
200 Tâches	661,96	321,97	468,95	631	367,69
300 Tâches	849,62	440,3	559,52	770,91	545,42
400 Tâches	1158,86	641,08	818,91	1035,6	671,46
500 Tâches	1313	869,15	1109,67	1191,04	949,3
600 Tâches	1540,76	1093,57	1389,25	1297,61	1114,82

TABLE 4.5 – Résultat de simulation du makespan (par nombre des tâches)

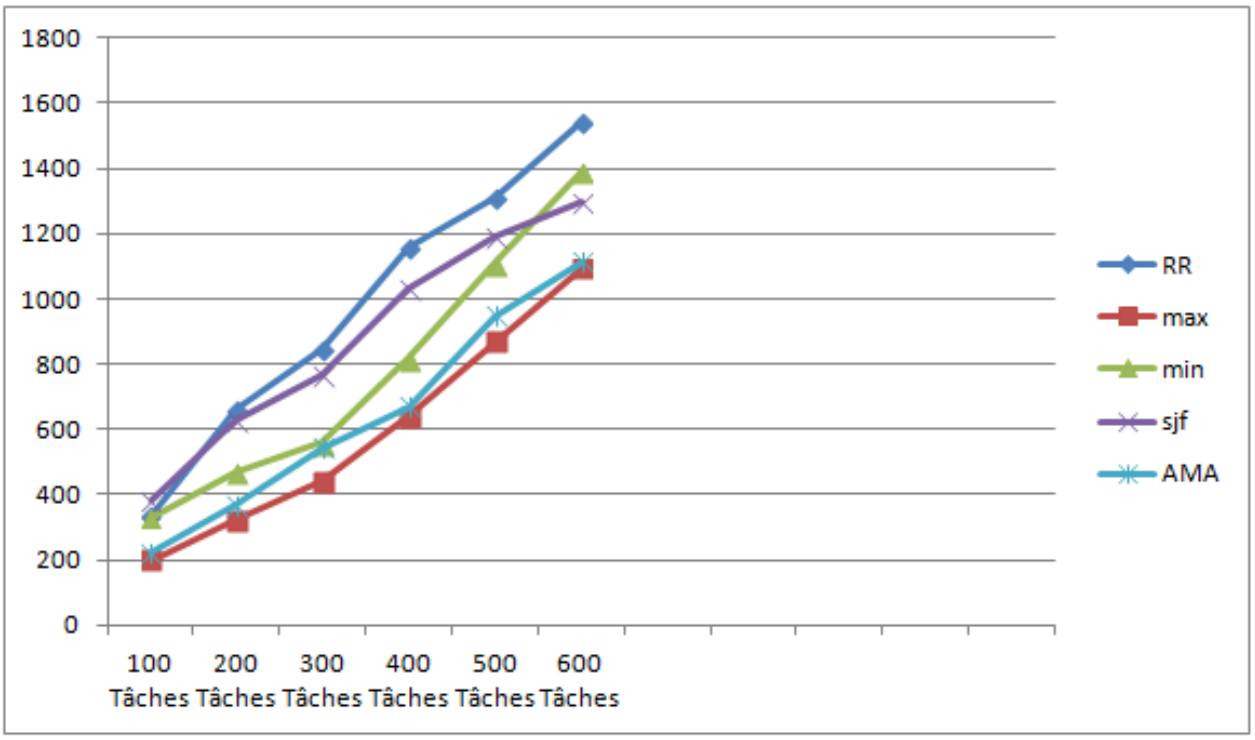

FIGURE 4.8 – Courbe comparative entre les algorithmes d'ordonnancemes des tâches par makespan/nb tâches

-Pour 600 tâches nous avons varié le nombre des VM de 40 à 160.

Algo/nmbr des Vms	RR	Max-Min	Min-Min	sjf	AMA
40 VMs	3817,92	3514,26	3799,09	3691,51	3631,82
80 VMs	2128,58	1694,43	1835,8	2038,21	1745,82
120 VMs	1635,23	1085,56	1378,25	1294,75	1142,2
140 VMs	1386,46	919,47	1114,88	1263,89	977,22
160 VMs	1191,04	743,43	920,61	1078,62	778,4

TABLE 4.6 – Résultat de simulation du makespan (par nombre des VMs)

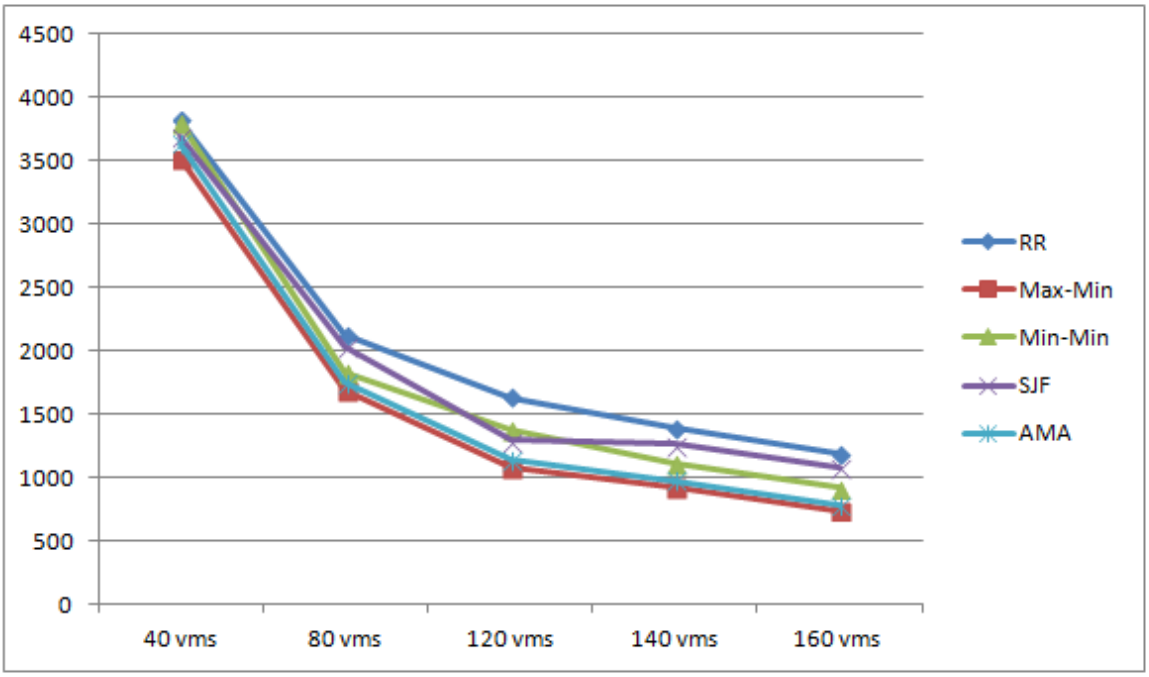

FIGURE 4.9 – Courbe comparative entre les algorithmes d'ordonnancement des tâches par makespan/nb VM

Discussions :

• Nous distinguons qu'à chaque fois le nombre des tâches augmente, le makespan augmente à son tour puisque la charge du travail s'amplifient.

• On observe une diminution générale du Makespan pour tous les algorithmes lorsque le nombre de VM augmente. Cela est logique car plus il y a de VM, plus il y a de ressources disponibles pour exécuter les tâches en parallèle, ce qui réduit le temps total d'exécution.

• On constate que l'algorithme Max-Min donne un meilleur résultat que les autres algorithmes en terme du makespan.

• la différence entre les méthodes en termes de temps n'est pas flagrante surtout pour un nombre réduit des tâches et des VM.

• Notre algorithme montre des performances compétitives en particulier lorsque le nombre de VMs est élevé.

• Notre algorithme montre aussi des performances compétitives avec les autres algorithmes d'ordonnancement lorsque le nombre de tâches augmente.

• Au début, la différence n'est pas aussi remarquable, mais lors du passage à l'échelle, nous décelons très bien la différence estimable dans le temps.

• Lors du passage à l'échelle, notre algorithme donne de bons résultats par apport aux algorithmes de base.

• Notre solution est tolérante aux fautes, car même si un nœud tombe en panne, les autres continueront à fonctionner.

4.6 Conclusion

Nous avons pu présenter dans ce chapitre l'essentiel de notre solution, basée sur l'implémentation des systèmes multi agents, où deux types d'agents collaborent pour garantir une prise de décision efficace pour l'ordonnancement des tâches. Le fonctionnement de celle-ci se déroule en deux phases distinctes : l'ordonnancement des tâches et la simulation de leurs exécutions.

Les résultats obtenus démontrent que notre algorithme donne de bons résultats en terme du makespan par apport aux algorithmes classiques d'ordonnancement RR, SJF, Min-Min et Max-Min.

L'utilisation des systèmes Multi-agent pourrait être vue comme une solution favorable pour le passage à l'échelle. Or, cette étude va nous pousser à élargir au maximum le nombre de tâches et machines mis en service pour une meilleure évaluation de ce passage à l'échelle.

Dans ce cas là, et comme perspective, nous pourrons être capables de parfaire notre solution en employant plusieurs Data centers et en s'appuyant sur d'autre algorithmes sophistiqués.

Conclusion Générale

Conclusion Générale

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

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches.

Dans ce travail, nous avons proposé une approche d'ordonnancement des tâches basée sur les Systèmes Multi-agents dans le but d'optimiser le makespan.Nous avons comparé les algorithmes d'ordonnancement RR (Round Robin), SJF (Shortest Job First), Min-Min, Max-Min et notre propre algorithme Multi-agents à l'aide du simulateur CloudSim, en terme du Makespan. Après plusieurs simulations, nous avons remarqué que notre algorithme a réussi à améliorer considérablement l'objectif par rapport à l'algorithme Round Robin et sjf ,Min-Min.

Ainsi, pour notre modeste travail, nous nous sommes centrés sur une solution Mult-agents qui pourrait apporter une meilleure fiabilité et un passage à l'échelle.

Dans nos futurs travaux, nous envisageons plusieurs perspectives, nous comptons perfectionner notre algorithme en ajoutant :

-d'autres objectifs comme l'équilibrage de charge, le cout et la fiabilité.

-des tâches dépendantes avec priorité.

-plusieurs datacenter afin d'évaluer le passage à l'échelle.

-d'autres algorithmes d'ordonnancement pour les agents hôtes.

Résumé

Le Cloud Computing est une technologie prometteuse qui facilite l'exécution des applications de divers domaines. Les algorithmes d'ordonnancement des tâches sont pour la plupart hautement centralisés. Cela limite leur capacité à passer à l'échelle, autrement dit à gérer de manière réactive des infrastructures de grande taille et une forte demande, qui sont de plus en plus courantes.

Au cours de notre travail, nous nous sommes intéressés aux façons d'améliorer les algorithmes d'ordonnancement des tâches ; l'une des techniques consiste à décentraliser le traitement par les systèmes multi-agents.

Nous avons mis en œuvre un algorithme d'ordonnancement des tâches basé sur un système d'agents intelligents, que nous avons validé au travers de simulations (notamment via l'outil CloudSim), Nous avons pu constater que notre algorithme se montrait particulièrement intéressant lorsque le nombre des tâches augmente.

Mots clés :Cloud Computing, Ordonnancement des Tâches, Système Multi-agents, CloudSim.

Abstract

Cloud computing is a promising technology that facilitates the execution of applications across various domains. However, most task scheduling algorithms are highly centralized, limiting their ability to scale, that is to effectively manage large-scale infrastructures and high demand, which are becoming increasingly common.

In our work, we have focused on ways to improve task scheduling algorithms, one of which involves decentralizing processing through multi-agent systems.

We have implemented a task scheduling algorithm based on intelligent agent systems, which we have validated through simulations (including using the CloudSim tool). We have observed that our algorithm shows particular promise when the number of tasks increases.

Keywords : Cloud Computing , Task scheduling, Multi-agent System, CloudSim.

الملخص

الحوسبة السحابية هي تقنية واعدة تسهل تنفيذ التطبيقات في مختلف المجالات. غالبًا ما تكون خوارزميات جدولة المهام مركزية للغاية. وهذا يحد من قدرتها على التوسع، أي إدارة البنية التحتية الضخمة والطلب المرتفع بشكل تفاعلي، وهو أمر شائع بشكل متزايد.

خلال عملنا، كنا مهتمين بطرق تحسين خوارزميات جدولة المهام؛ تتمثل إحدى التقنيات في المعالجة اللامركزية بواسطة أنظمة متعددة الوكلاء.

لقد قمنا بتنفيذ خوارزمية جدولة المهام استنادًا إلى نظام من الوكلاء الأذكياء، والتي قمنا بالتحقق من صحتها من خلال عمليات المحاكاة (على وجه الخصوص عبر أداة المحاكاة في تطبيقات و أبحاث الحوسبة السحابية). لقد تمكنا من رؤية أن الخوارزمية الخاصة بنا كانت مثيرة للاهتمام بشكل خاص عندما يزيد عدد المهام.

الكلمات مفتاحية : الحوسبة السحابية،جدولة المهام، نظام متعدد العملاء،أداة المحاكاة في تطبيقات و ابحاث الحوسبة السحابية.

Bibliographie

[Arravinth and Manjula, 2022] Arravinth, J. and Manjula, D. (2022). Multi-Agent with Multi Objective-Based Optimized Resource Allocation on Inter-Cloud. *Intelligent Automation & Soft Computing*, 34(1).

[Asnoune, 2016] Asnoune, M. E. A. (2016). Mise en place d'un système de communication multi-agents. Mémoire de master, Université Ibn Khaldoun - Tiaret.

[Astran, 2022] Astran (2022). L'histoire du cloud : quand, comment et par qui a-t-il été inventé ? https ://www.linkedin.com/pulse/lhistoire-du-cloud-quand-comment-et-par-qui-a-t-il-.

[Baccouche, 1995] Baccouche, L. (1995). *Un mécanisme d'ordonnancement distribué de tâches temps réel*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG).

[Benamar and Ayachi, 2017] Benamar, K. and Ayachi, C. (2017). *Conception et réalisation d'un éditeur graphique de réseau de pétri*. Thèse de doctorat, Université Ibn Khaldoun-Tiaret.

[Benhamza, 2016] Benhamza, K. (2016). *Conception d'un système multi-agents adaptatif pour la résolution de problème*. Thèse de doctorat, Université Badji Mokhtar, Annaba.

[Benmerzoug, 2010] Benmerzoug, D. (2010). Une architecture basée agent pour l'intégration d'applications d'entreprises. In *7ème Séminaire National en Informatique, SNIB'2010*.

[Bouteldja and Bakdi, 2019] Bouteldja, A. and Bakdi, K. (2019). Gestion des Dossiers des Œuvres Sociales de La Direction de L'éducation. Mémoire de master, Université Ibn Khaldoun-Tiaret.

[Bouzara, 2018] Bouzara, A. (2018). Sécurité dans le Cloud Computing. Thèse de doctorat, Université Ibn Khaldoun – Tiaret.

[Calheiros et al., 2011] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software : Practice and experience*, 41(1) :23–50.

[ElWessabi, 2014] ElWessabi, A. A. Y. (2014). Une approche basée agent mobile pour le cloud computing. Mémoire de master, Université de Batna 2.

[Elzeki et al., 2012] Elzeki, O. M., Reshad, M. Z., and Abu Elsoud, M. (2012). Improved max-min algorithm in cloud computing. *International Journal of Computer Applications*, 50(12).

[Ferber, 1997] Ferber, J. (1997). *Les systèmes multi-agents : vers une intelligence collective*. InterEditions.

[Figer, 2012] Figer (2012). Cloud computing - informatique en nuage. *Gestion de contenus numériques*. Année d'édition de la version électronique 2012.

[GmbH, 2009] GmbH, T.-S. I. (2009). *Livre blanc : Le « Cloud Computing »*. T-Systems International GmbH, Hahnstraße 43d, 60528 Frankfurt am Main, Allemagne.

[Hamdani et al., 2019] Hamdani, N., Kerroum, S., and Ouallouche, N. (2019). Etude et comparaison des failles de sécurité d'OpenStack et OpenNebula. Mémoire de master, Université Mouloud Mammeri de Tizi-Ouzou.

[Hennion et al., 2012] Hennion, R., Tournier, H., and Bourgeois, E. (2012). *Cloud computing : décider, concevoir, piloter, améliorer*. Editions Eyrolles.

[Kamilia and Chahinas, 2015] Kamilia, K. and Chahinas, L. (2015). Conception et réalisation d'une application mobile sous «Android» pour la réservation de vols. Mémoire de master, Université Mouloud Mammeri.

[Khalil et al., 2017] Khalil, K. M., Abdel-Aziz, M., Nazmy, T. T., et al. (2017). Cloud simulators–an evaluation study. *International Journal Information Models and Analyses*, 6(1).

[Khelifi and Bellil, 2014] Khelifi, M. and Bellil, A. (2014). Implémentation d'une solution sécurisée dans une architecture d'un Cloud Computing. Mémoire de master, Université Mouloud Mammeri.

[Maaref, 2012] Maaref, S. (2012). Cloud en afrique : Situation et perspectives. Rapport de l'étude.

[Marlene and Ernest, 2017] Marlene, M. N. and Ernest, R. Z. (2017). *La migration en temps reel des machines virtuelles dans le cloud computing*. Thèse de doctorat, Université Mouloud Mammeri.

[Marzougui, 2014] Marzougui, B. (2014). *Contribution à la modélisation et à la vérification des systèmes multi-agents*. Thèse de doctorat, CNAM, Paris.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Special Publication 800-145, National Institute of Standards and Technology (NIST).

[Meroufel, 2016] Meroufel, B. (2016). *Coopération dynamique des agents pour la tolérance aux pannes dans les systèmes à larges échelles*. Thèse de doctorat, Université Oran 1 Ahmed Ben Bella.

[Mohialdeen, 2013] Mohialdeen, I. A. (2013). Comparative study of scheduling algorithms in cloud computing environment. *Journal of Computer Science*, 9(2) :252–263.

[Nithya and Jayapratha, 2014] Nithya, G. and Jayapratha, G. (2014). A multi-agent brokering approach and jumper firefly algorithm for job scheduling in cloud computing. In *2014 International Conference on Intelligent Computing Applications*, pages 52–58. IEEE.

[Patel and Patel, 2013] Patel, R. and Patel, S. (2013). Survey on resource allocation strategies in cloud computing. *International Journal of Engineering Research & Technology (IJERT)*, 2(2) :1–5.

[Priyadarsini and Arockiam, 2014] Priyadarsini, R. J. and Arockiam, L. (2014). Performance evaluation of min-min and max-min algorithms for job scheduling in federated cloud. *International Journal of Computer Applications*, 99(18) :47–54.

[Rahmani and Otmani, 2021] Rahmani, A. and Otmani, S. (2021). Allocation multiobjectif des workflows aux ressources d'un environnement Cloud Computing. Mémoire de master, Université d'Ain Témouchent Belhadj Bouchaïb.

[Singh and Chana, 2016] Singh, S. and Chana, I. (2016). A survey on resource scheduling in cloud computing : Issues and challenges. *Journal of Grid Computing*, 14 :217–264.

[Thomas, 2005] Thomas, V. (2005). *Proposition d'un formalisme pour la construction automatique d'interactions dans les systèmes multi-agents réactifs*. Thèse de doctorat, Université Henri Poincaré-Nancy I.

[Touaf, 2005] Touaf, S. (2005). *Diagnostic logique des systèmes complexes et dynamiques dans un contexte multi-agent*. Thèse de doctorat, Université Joseph-Fourier-Grenoble I.

[Voorsluys et al., 2011] Voorsluys, W., Broberg, J., and Buyya, R. (2011). Introduction to cloud computing. *Cloud computing : Principles and paradigms*, pages 1–41.

[Weiss, 1999] Weiss, G. (1999). *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT press.

[Wooldridge, 1999] Wooldridge, M. (1999). Intelligent agents. In Weiss, G., editor, *Multiagent systems : A modern approach to distributed artificial intelligence*, pages 27–73. MIT Press, Cambridge, MA.

[Yang et al., 2021] Yang, Y., Ren, F., and Zhang, M. (2021). An agent-based adaptive mechanism for efficient job scheduling in open and large-scale environments. *Journal of Systems Science and Systems Engineering*, 30(4) :400–416.

[Yang et al., 2024] Yang, Y., Ren, F., and Zhang, M. (2024). A bdi agent-based task scheduling framework for cloud computing. *arXiv preprint arXiv :2401.02223*.

[Yeboah et al., 2015] Yeboah, T., Odabi, I., and Hiran, K. K. (2015). An integration of round robin with shortest job first algorithm for cloud computing environment. In *International Conference On Management, Communication and Technology*, volume 3, pages 1–5.

[Zeng et al., 2015] Zeng, L., Veeravalli, B., and Li, X. (2015). SABA : A security-aware and budget-aware workflow scheduling strategy in clouds. *Journal of Parallel and Distributed Computing*, 75 :141–151.

[Zhu et al., 2015] Zhu, X., Chen, C., Yang, L. T., and Xiang, Y. (2015). Angel : Agent-based scheduling for real-time tasks in virtualized clouds. *IEEE Transactions on Computers*, 64(12) :3389–3403.