République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option Réseaux et Systèmes Distribués(RSD)

Thème

Développement d'un agent BDI pour la prise de décision d'ordonnancement dans un environnement Cloud Computing

Réalisé par :

- HARROU Abdelaziz
- GUERMOUDI Abdelkader

Présenté le 23 Juin 2016 devant la commission d'examination composée de MM.

- LEHSAINI Mohamed *(Président)*
- BENMAMMAR Badr *(Encadrant)*
- BOUAFIA Zouhir *(Co- Encadrant)*
- BENZIANE Yaghmoracen *(Examinateur)*
- BENAMAR Abdelkrim *(Examinateur)*

Année Universitaire : 2015-2016

Remerciements

En premier lieu, nous remercions الله عز وجل *de nous avoir donné la force et la patience nécessaire pour achever ce mémoire.*

Nous tenons à remercier notre Encadrant Monsieur BENMAMMAR Badr, enseignant à l'université de Tlemcen pour son aide à réaliser notre mémoire ainsi que pour sa disponibilité et son soutien.

Nous remercions également Monsieur BOUAFA Zoheir enseignant à l'université de AÏN TEMOUCHENT, d'avoir co-encadré notre travail, de son suivi et ses conseils.

Nous tenons également à adresser nos remerciements aux membres du jury Messieurs LEHSAINI Mohamed, BENAMAR Abdelkrim et BENZIANE Yaghmoracen 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 systèmes distribués 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

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 et mon instruction.

À mon épouse pour son soutien inconditionnel tout au long de mes études.

À ma fille Fatima Zohra.

À mes sœurs et à mes frères pour leur appui et leur soutien multiforme.

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

A mon binôme pour le frère agréable qu'il restera pour moi.

A tous mes amis.

Dédicaces

Je dédie ce modeste travail:

A mes très chers parents qui n'ont jamais cessé de m'encourager pour entreprendre mes études et atteindre mes objectifs.

A ma très chère femme et mon adorable fille pour leur soutien, leur patience et leurs encouragements

A Mes très chers frères et sœur.

A tous les membres de ma famille, tantes, oncles, cousins et cousines.

A ma belle famille

A mon binôme qui m'a été un collègue très intime, et mes compatriotes de la promotion « Master 2016 » et à toute sa famille

A mes très chers Amis

A tous mes camarades de promotion

A tous ceux qui m'ont aidé et encouragé pour l'élaboration de ce mémoire.

Table de matière

Introduction générale

Le cloud computing présente une technologie prometteuse qui facilite l'exécution des applications scientifiques et commerciales. Il fournit des services flexibles et évolutifs, à la demande des utilisateurs, via un modèle de paiement à l'usage. Généralement, il peut fournir trois types de services: SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service), et quatre modèles de déploiement : cloud public, cloud privé, cloud hybride et cloud communautaire.

Dans le même temps, les systèmes multi-agents (SMA) représentent un nouveau concept dans les applications distribuées. Les SMA sont basés sur de multiples agents interagissent entre eux pour résoudre des problèmes en utilisant une approche décentralisée où plusieurs agents contribuent à la solution en coopérant l'un avec l'autre.

Dans les SMA, une nouvelle approche est apparue, s'agit elle l'approche BDI, qui décrit l'état interne des agents, dit *agents BDI*, en termes d'attitudes mentales, et aussi de définir une architecture de contrôle grâce à laquelle l'agent peut sélectionner son plan. Le mécanisme de raisonnement d'un agent BDI, déclenché par des événements perçus, se base sur les attitudes mentales de celui-ci qui sont représentées par les Croyances (Beliefs), les Désirs (Desires) et les Intentions (Intentions).

La phase de prise de décision dans l'ordonnancement des tâches nécessite d'effectuer plusieurs choix. Il faut préciser quels nœuds sont responsables de la prise de décision. Dans le cas d'une approche multi-agents, il faut spécifier quels nœuds prennent les décisions, afin d'éviter les actions contradictoires. Cela peut nécessiter de mettre en place un mécanisme de collaboration et de synchronisation. L'architecture BDI est celle qui est préférée dans la communauté de recherche agents intelligents et systèmes multi-agents. La prise de décision conçus selon une architecture BDI (Belief-Desire-Intention) permet de raisonner, de collaborer et de prendre des décisions d'adaptation dirigées par des buts en tenant compte de l'état de l'environnement qui change avec le temps. L'objectif de cette technique vise à optimiser plusieurs métriques. Notamment, l'équilibrage de charge, le coût de communication, le délai d'exécution le temps d'attente et de réponse.

Pour cela, nous avons entrepris notre étude selon les trois chapitres suivants :

Chapitre I : nous étudierons quelques notions sur le cloud computing (architecture, principe, caractéristiques, ……).

Chapitre II : Ce chapitre est consacré à la présentation de quelques notions des systèmes multi-agents, et nous détaillons un peu de plus l'approche des agents BDI.

Chapitre III : Dans ce chapitre, nous avons décrit notre contribution à résoudre le problème de l'ordonnancement dans le cloud computing.

CHAPITRE I

CLOUD COMPUTING

« L’informatique dans le nuage »

I.1 Introduction

Indéniablement, la technologie de l'internet se développe d'une manière exponentielle depuis sa création. Actuellement, une nouvelle "tendance" a fait son apparition dans le monde de l'IT[1], il s'agit du Cloud Computing, il s'appuie sur le WEB 2.0[2], offre des opportunités aux sociétés de réduire les coûts d'exploitation des logiciels par leurs utilisations directement en ligne.

Dans ce chapitre nous allons présenter les notions fondamentales du Cloud Computing, ses enjeux, ses évolutions et son utilité ainsi que la technologie qui la constitue et les différents acteurs du secteur.

I.2 Définition

Le cloud computing, ou l'informatique en nuage, est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet. Le cloud computing se caractérise par sa grande souplesse.

Selon la définition du NIST, le cloud computing est l'accès via un réseau de télécommunications, à la demande et en libre-service, à des ressources informatiques partagées configurables. Il s'agit donc d'une délocalisation de l'infrastructure informatique.

Une autre définition du NIST : ***« L'informatique dans le nuage est une nouvelle façon de délivrer les ressources informatiques, et non une nouvelle technologie »***

Le Cloud est donc une nouvelle façon de concevoir l'informatique, du réseau aux applications, entraînant une nouvelle forme de consommation. Le Cloud permet aux organisations d'activer et stopper les ressources suivant l'évolution des besoins, mettre dynamiquement à jour des éléments d'infrastructure et applications, sans devoir s'inquiéter de la création de nouvelles infrastructures pour chaque nouvelle demande.

[1] Technologies de l'information et de la communication
[2] Web participatif, social et intelligence collective. Concept proposé par Tim O'Reilly en 2005.

I.3 Historique

C'est la cinquième génération de l'informatique après les MainFrames, les PCs, les Clients/Serveurs et le Web [1].

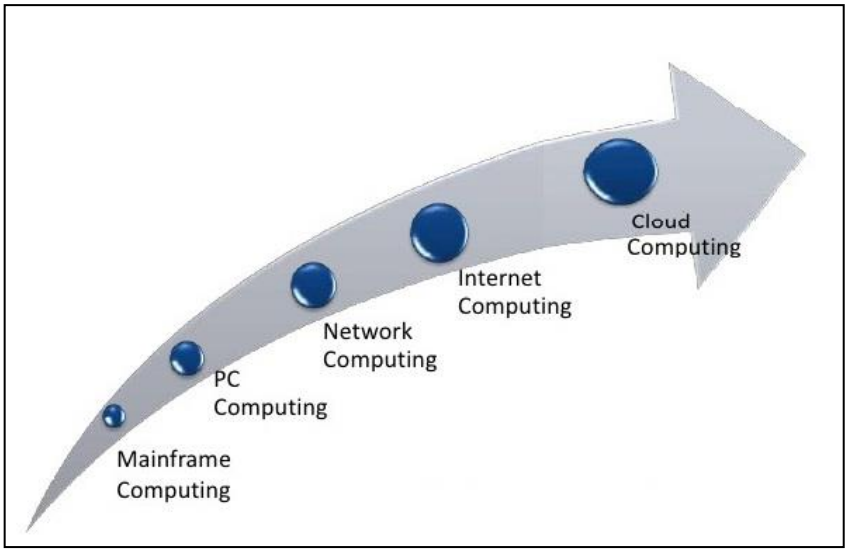

Figure I.1 : Evolution de l'informatique

L'origine du concept du Cloud provient du fait que les informaticiens symbolisent Internet sous la forme d'un nuage dans leurs schémas.

Figure I.2 : Schéma symbolique d'internet

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

Pour revenir au mot « cloud computing », les origines sont plutôt obscures. Au départ, le terme de « nuage » est présenté comme une métaphore pour représenter internet. Ce terme a été utilisé dès 1994. Cependant l'idée même du cloud computing remonte à bien plus tôt. A l'époque, de larges unités centrales ont été utilisées dans les entreprises et dans les universités. Celles-ci représentaient un coût élevé et

l'investissement dégagé pouvait être rentabilisé en autorisant le partage de l'accès à d'autres utilisateurs. Plus connu sous le nom de « time-sharing » ou temps partagé, reconnu comme l'avènement du cloud computing. Etant donné l'évolution rapide de l'internet et la simplicité à se procurer du matériel informatique, le cloud computing s'est de plus en plus développé.

I.4 Principe du Cloud computing

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

I.5 Caractéristiques du Cloud computing

I.5.1 Service à la demande :

La capacité de stockage et la puissance de calcul sont adaptées automatiquement au besoin d'un consommateur. Ce qui contraste avec la technique classique des hébergeurs où le consommateur doit faire une demande écrite à son fournisseur en vue d'obtenir une augmentation de la capacité - demande dont la prise en compte nécessite évidemment un certain temps. En cloud computing la demande est automatique et la réponse est immédiate.

I.5.2 Accès via des réseaux à large échelle :

Les services de *cloud computing* sont mis à disposition sur l'Internet, et utilisent des techniques standardisées qui permettent de s'en servir aussi bien avec un ordinateur qu'un téléphone ou une tablette.

I.5.3 Mutualisation :

Elle permet de combiner des ressources hétérogènes (matériel, logiciel, trafic réseau) en vue de servir plusieurs consommateurs à qui les ressources sont automatiquement attribuées. La mutualisation améliore l'évolutivité et l'élasticité et permet d'adapter automatiquement les ressources aux variations de la demande.

I.5.4 Elasticité rapide :

Les ressources peuvent être allouées et libérées de manière élastique selon les besoins.

I.5.5 Paiement à l'usage :

La quantité de service consommée dans le cloud est mesurée, à des fins de contrôle, d'adaptation des moyens techniques et de facturation [2].

I.6 Modèles de déploiement

I.6.1 Cloud public :

Les utilisateurs ont accès à des services cloud via l'Internet public sans savoir précisément où sont hébergées leurs données ni où sont exécutés leurs traitements. Les ressources informatiques et bases de données de l'utilisateur peuvent être hébergées dans n'importe quel Datacenter du prestataire et peuvent passer d'un Datacenter à l'autre afin d'optimiser les capacités du prestataire.

I.6.2 Cloud privé :

Ensemble des ressources sont exclusivement mises à disposition d'un seul client. Le cloud privé peut être géré par l'entreprise utilisatrice elle-même ou par un prestataire externe

I.6.3 Cloud hybride :

Ils associent à la fois des infrastructures et des Cloud privés et publics. Une partie des données ou des infrastructures est gérée en interne par l'entreprise, dans ses locaux ou chez un prestataire et communique avec des ressources Cloud.

I.6.4 Cloud communautaires :

Terme désignant un cloud ou « nuage » qui permet à plusieurs entreprises ou organisations de partager des ressources en mode cloud, ces ressources étant exclusivement dédiées à ces organisations (avec des dispositifs d'allocations des ressources ou de répartition de charge entre elles)[2].

I.7 Modèles de service

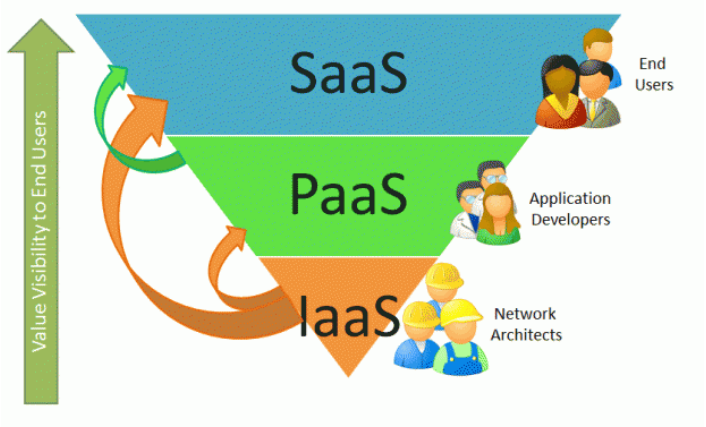

Figure I.3 : Les couches du cloud computing [22]

I.7.1 SaaS (Software as a Service):

Il s'agit de la mise à disposition d'applications sous la forme de service (CRM[3], outils collaboratifs, messagerie, ERP[4], …). Ce concept consiste à proposer un abonnement à un logiciel plutôt que l'achat d'une licence. Plus d'installation, plus de mise à jour à gérer (elles sont continuées chez le fournisseur), plus de migration de données etc. Paiement à l'usage.

I.7.2 PaaS (Platform as a Service) :

Il s'agit des plateformes du nuage, regroupant principalement les serveurs mutualisés et leurs systèmes d'exploitation. En plus de pouvoir délivrer des logiciels en mode SaaS, le PaaS dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires.

[3] La gestion de la relation client (GRC), ou gestion des relations avec les clients, en anglais *customer relation ship management* (CRM), est l'ensemble des outils et techniques destinés à capter, traiter, analyser les informations relatives aux clients et aux prospects, dans le but de les fidéliser en leur offrant le meilleur service

[4] Enterprise resource planning, signifiant littéralement en anglais, « planification des ressources de l'entreprise », et traduit en français par « progiciel de gestion intégré » (PGI).

I.7.3 IaaS (Infrastructure as a Service) :

Il s'agit de la mise à disposition, à la demande, de ressources d'infrastructures (serveurs, moyens déstockage, réseau …) dont la plus grande partie est localisée à distance dans des Datacenter. L'IaaS permet l'accès aux serveurs et à leurs configurations pour les administrateurs de l'entreprise. Le client a la possibilité de louer des clusters, de la mémoire ou du stockage de données. Le coût est directement lié au taux d'occupation. Une analogie peut être faîte avec le mode d'utilisation des industries des commodités (électricité, eau, gaz) ou des Télécommunications [3].

IaaS	PaaS	SaaS
• Amazon – offres EC2 et AWS • Microsoft – offre Azur	• Microsoft – offre Azur • Google – offre Google App Engine	• Google – offre Google Apps (messagerie et bureautique) • Sales Force – CRM (Customer Relationship Management) • Microsoft – offre Office 365 (outils collaboratifs)

Table I.1 : Les grands acteurs mondiaux de Cloud public

I.8 Eléments constitutifs du Cloud Computing

I.8.1 La virtualisation :

La virtualisation est un mécanisme informatique qui consiste à faire fonctionner plusieurs systèmes, serveurs ou applications, sur un même serveur physique. La virtualisation est un composant technique clé dans le Cloud Computing.

La virtualisation repose sur le mécanisme suivant :

- Un système d'exploitation principal (appelé « système hôte ») est installé sur un serveur physique unique. Ce système sert d'accueil à d'autres systèmes d'exploitation.

- Un logiciel de virtualisation est installé sur le système d'exploitation principal. Il permet la création d'environnements clos et indépendants sur lesquels seront installés d'autres systèmes d'exploitation (« systèmes invités »). Ces environnements sont des « machines virtuelles ».

- Un système invité est installé dans une machine virtuelle qui fonctionne indépendamment des autres systèmes invités dans d'autres machines virtuelles. Chaque machine virtuelle dispose d'un accès aux ressources du serveur physique (mémoire, espace disque…).

I.8.2 Datacenter :

Un centre de traitement de données (data centre en anglais) est un site physique sur lequel se trouvent regroupés des équipements constituants du système d'information de l'entreprise (mainframes, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne et/ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée [4].

I.9 Points complémentaires

I.9.1 Modèles « as a service » :

Dans le monde des offres de services des fournisseurs/hébergeurs du microcosme informatique, tout devient « as a service ». Le terme générique est ITaaS (Information Technology as a Service, technologies de l'information « as a Service »). Le as a service sous-entend la caractéristique d'élasticité du Cloud et donc la flexibilité d'adaptation permettant d'emboîter le pas de l'innovation et de la création de valeur [5].

Voici une liste non exhaustive de services/applications as a service :

- Stockage ou Storage as a Service.
- Sauvegarde ou Backup as a Service.
- Bureau ou Desktop as a Service.
- Communication as a Service.

I.9.2 Multitenants :

La notion de multitenants consiste à héberger des services/applications de plusieurs clients sur la même infrastructure de Cloud[5].

I.9.3 Interconnexion :

Un service/application du Cloud ne peut être vu uniquement comme un système en vase clos. Il dépend d'un écosystème.

Il lui faut des API pour les programmeurs afin de faciliter les échanges avec d'autres services/applications et afin de gérer le service à distance. Il lui faut, dans le contexte du Cloud public en particulier, prendre en compte les API du Web 2.0 qui peuvent être utilisées sous forme d'applications composites dénommées également mash-up. Une application composite est une application qui combine du contenu ou un service provenant de plusieurs applications plus ou moins hétérogènes [5].

Une application composite permet par exemple à une société de vente et de location immobilières d'intégrer ses opportunités sur des cartes provenant, elles, d'un service fourni par un prestataire externe.

Il lui faudra éventuellement s'appuyer sur des systèmes d'échanges de données basés sur la messagerie (service SMTP) ou le transfert de fichiers (services FTP ou, de préférence, SFTP).

Il sera nécessaire de prendre en compte dès le départ de la mise en place du service/application l'impact de :

- L'**authentification** permettant l'accès au service/application.
- La **fédération d'identités** pour faciliter aux utilisateurs d'une entreprise la gestion des mots de passe des services/applications auxquels ils accèdent.
- Le **bus d'intégration applicatif** pour gérer et cadencer les échanges avec d'autres services/applications.
- Le **centre de services** (point unique d'entrée pour les appels utilisateurs).

Ces points sont particulièrement structurants et doivent être abordés avant la contractualisation et/ou la conception d'un service/application du Cloud.

I.10 Sécurité dans le Cloud Computing

La sécurité et la conformité émergent systématiquement comme les principales préoccupations des responsables informatiques lorsqu'il est question de Cloud Computing, des préoccupations encore plus accentuées lorsqu'il s'agit de Cloud public. La sécurité permet de garantir la confidentialité, l'intégrité, l'authenticité et la disponibilité des informations.

Certaines questions légitimes reviennent sans cesse :

- Mes données sont-elles sûres dans le Cloud ?
- Où sont stockées mes données ?
- Qui va avoir accès à mes données ?
- Aurais-je accès à mes données à n'importe quel moment ?
- Que deviendrons mes données s'il y a interruption du service ?

La mise sur pied d'une solution de Cloud Computing comporte des problèmes de sécurité inhérents à la solution elle-même. Le fait de centraliser toutes les informations sur un site pose un grand nombre de problèmes. On peut citer comme problème potentiel :

- Une possible interruption massive du service.
- Une cible de choix pour les hackers.
- Interface et API non sécurisé.

Ce point de vulnérabilité du Cloud Computing fait depuis quelques années l'objet de recherches avancées. Il a été créé un organisme chargé de mettre sur pied des normes en matière de sécurité dans le Cloud Computing. Cet organisme s'appelle CSA[5]. Du travail de cet organisme, il en est ressorti certaines techniques utilisées de nos jours pour améliorer la sécurité du Cloud Computing. Parmi ces techniques on peut citer :

- **La multi-location** : cette technique permet de créer des instances d'une même donnée sur plusieurs sites différents. Elle permet une récupération facile en cas de désastre.

[5] Cloud Security Alliance : est un organisme dont la mission est de promouvoir l'utilisation des meilleures pratiques pour fournir l'assurance de la sécurité dans le Cloud Computing.

- **Le chiffrement** : le chiffrement de l'accès à l'interface de contrôle, le chiffrement des données dans le Cloud.

La sécurité absolue n'existe pas, donc le problème de sécurité reste le plus souvent un problème de confiance entre le fournisseur et le consommateur de service. Cette confiance se traduit par la signature d'un contrat nommé SLA (Service Level Agreement). Ce contrat précise les taux de disponibilité du service. En règle générale, et pour la plupart des fournisseurs, ce taux est supérieur à 99 % [6].

I.11 Avantages et inconvénients du Cloud Computing

Avantages	Inconvénients
– Disponibilité et extensibilité – Dynamicité – Tolérance aux pannes – Mutulaisation des ressources	– Hétérogénéité – Absence de localité – Portage des applications – Sécurité

Table I.2 : Avantages et inconvénients du Cloud Computing

I.12 Conclusion

Nous avons abordé dans ce chapitre le paradigme Cloud computing qui permet aux entreprises de disposer d'infrastructures et de logiciels directement en ligne sur Internet. On a vu les trois services du Cloud l'IaaS, PaaS et le SaaS. Ces trois services peuvent se déployer sous quatre formes de topologies différentes : le Cloud public, le Cloud privé, le Cloud hybride et enfin le cloud communautaire.

Le Cloud computing véhicule un concept de simplicité mais implique dans la réalité une vraie démarche de gestion de la complexité. Gouvernance, standardisation, processus, outils, technologies, tous les acteurs impactés sont entraînés dans l'inexorable spirale de l'évolution vers le Cloud, qu'il soit public, privé, hybride [5].

CHAPITRE II

Système Multi Agent

II.1 Introduction

L'explosion de la technologie Internet et des réseaux a contribué à bouleverser un bon nombre d'habitudes établies depuis plusieurs décennies. Les documents papier échangés de mains en mains laissent progressivement place aux documents électroniques transmis automatiquement par des machines. Les domaines d'application des systèmes informatiques deviennent dès lors de plus en plus vastes et complexes.

Il leur est demandé de gérer de plus en plus d'interactivité, de réactivité, de mobilité dans leurs utilisations les plus répandues. Ainsi, l'apparition des systèmes multi-agents (SMA) apporte une nouvelle dimension au concept de modélisation pour représenter une application du monde réel avec un degré approprié de complexité et de dynamique. Les SMA, inspirés des systèmes naturels complexes, sont composés d'agents logiciels qui interagissent pour accomplir un but commun. Les relations entre les agents sont dynamiques et totalement autonomes, i.e. aucun contrôle central ne supervise l'exécution globale du système. En outre, idéalement, les développeurs doivent passer moins de temps à se préoccuper des subtilités d'une technologie en particulier, et plus de temps à la mise en œuvre de solutions à leurs problèmes. Ainsi, pour aider les utilisateurs de SMA, des bibliothèques ont tendance à être mises au point. Puis sont apparus les plates-formes qui sont des environnements de développement, ainsi que les méthodes de conception qui sont un processus de développement allant de la définition des besoins jusqu'à l'implantation voire au déploiement du système.

II.2 Historique

Les systèmes multi-agent sont apparus au carrefour des recherches sur l'intelligence artificielle distribuée et sur la vie artificielle. Ces systèmes sont développés à partir de schémas de raisonnement ou d'organisations empruntés aux domaines de la vie et de la société [7].

II.2 .1 L'intelligence artificielle :

L'objectif de l'IA est d'étudier les modes de raisonnement à partir de systèmes virtuels. Ces systèmes sont capables de résoudre un problème en utilisant des symboles, c'est à dire un langage simplifié.

Dans un premier temps, l'idée était de distribuer l'intelligence ou la connaissance en utilisant une assemblée de spécialistes virtuels. Les spécialistes se concertaient par le biais d'un espace commun de mémoire (tableau noir). Chaque spécialiste pouvait y déposer, modifier et effacer des données, au bout d'un moment sur le tableau, la solution devait émerger des actions des différents spécialistes. Une autre méthode simulée était l'élaboration par l'assemblée des spécialistes, le spécialiste possédant toutes les connaissances pour résoudre, à lui seul, le problème posé. Hewitt [8] en 1991 remarqua alors l'importance du contrôle des interventions des spécialistes. Il s'orienta vers un contrôle distribué et non plus celui d'un choix séquentiel. Il développa ce contrôle distribué par l'envoi de messages entre les différents acteurs. Ce fut la base du langage d'acteur.

II.2 .2 Premiers systèmes multi-agents :

D'après Ferber [7], deux systèmes ont marqué le développement des systèmes multi-agent :

- Le modèle DVMT (Distributed Vehicule Monitoring Test) permettait d'obtenir une image du trafic routier. Il traitait les informations transmises par plusieurs capteurs. Celles-ci pouvaient être contradictoires, redondantes ou bruitées. Ce modèle a permis d'étudier notamment les protocoles de coopération, de négociation et de planification dans les systèmes multi-agent.
- Le système MACE, a démontré que la communication n'est pas suffisante si elle n'est pas alliée à une représentation de son environnement par l'agent [9].

II.3 <u>Définition d'un agent.</u>

Un agent est définit comme étant un Système Informatique, **situé** dans un environnement, et qui agit d'une façon **autonome** et **flexible** pour atteindre certains objectifs pour lesquels il a été conçu [10].

- Situé : un agent est dit situé, s'il est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement,
- Autonome : un agent est dit autonome s'il est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.

- Flexible : un agent est dit flexible s'il est capable d'agir de manière réactive, proactive et sociale:
  - ✓ Réactif : être capable de percevoir son environnement et de réagir dans le temps.
  - ✓ Proactif : prendre l'initiative et être opportuniste au bon moment.
  - ✓ Social : être capable d'interagir avec les autres agents quand la situation l'exige (pour compléter ses tâches ou coopérer avec eux).

Un agent est donc une entité physique ou virtuelle en situation dans un environnement avec lequel il interagit de façon autonome et flexible. Cette notion de situation ou d'agent situé implique que les agents autonomes doivent agir dans un monde réel, en perpétuel changement, partiellement observable et intrinsèquement imprévisible. De même ils doivent réagir en temps réel parce que l'environnement change constamment, et parce qu'ils doivent éventuellement tenir compte des actions d'autres agents [11].

II.4 Différentes catégories et modèles d'agents

Après avoir défini les agents et les systèmes multi-agents, nous présentons dans cette partie les différents modèles d'agents, afin de comprendre leurs caractéristiques et leurs modes de fonctionnement. Nous distinguons deux grandes familles d'agents les agents réactifs et les agents cognitifs.

II.4 .1 Agents cognitifs :

C'est le premier modèle d'agents qui a été proposé. Il est nommé aussi agent délibératif. Il est basé sur l'IA symbolique, et il permet de planifier les actions d'un agent au sein de son environnement [12]. En effet, les agents cognitifs sont capables à eux seuls de réaliser des opérations relativement complexes.

Généralement, ils coopèrent les uns avec les autres pour atteindre un but commun (résolution d'un problème, une tâche complexe, etc). Ils possèdent un ensemble de représentations explicites (sur l'environnement, sur les autres agents et sur eux-mêmes) décrits dans une base de connaissances sur laquelle ils peuvent raisonner. Ils réagissent en fonction de leurs connaissances, leurs buts, de leurs échanges d'informations avec les autres agents et de la perception de l'environnement (voir

figure II.1). Ils sont dotés de moyens et mécanismes de communication pour gérer les interactions avec d'autres agents (coopération, coordination et négociation).

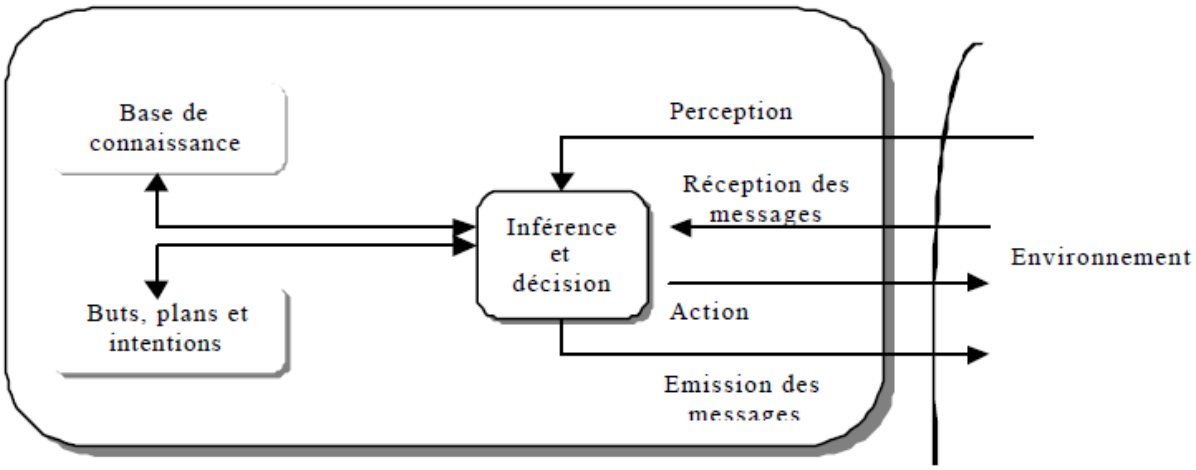

Figure II.1 : Modèle d'un agent cognitif

Ce modèle d'agent est une métaphore du modèle humain et s'appuie sur la sociologie des organisations [13]. Nous pouvons aussi trouver son origine dans la volonté de faire coopérer des systèmes experts classiques dans le domaine d'IA.

L'agent cognitif traite généralement des informations qualitatives tout en utilisant un raisonnement qualitatif ou symbolique

II.4 .2 Agents réactifs

Parmi les critiques du raisonnement symbolique figure Brooks, qui, par le biais de plusieurs papiers [14], manifesta son opposition au modèle symbolique et proposa une approche alternative appelée aujourd'hui IA active. Selon lui, le comportement intelligent devrait émerger de l'interaction entre divers comportements plus simples.

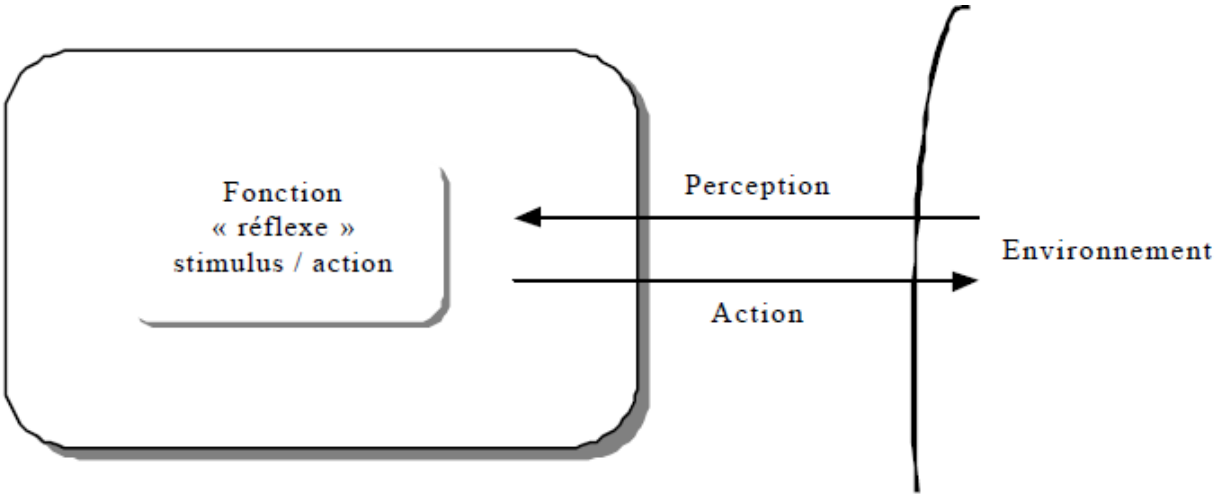

Figure II.2 : Modèle d'un agent réactif

Dans ce même contexte, nous considérons que les agents réactifs n'ont de connaissance explicite, ni de l'environnement, ni des autres agents, ni de leur passé, pas plus que de leurs buts (pas de planification de leurs actions). Ce sont des agents qui réagissent uniquement à leur perception de l'environnement et qui agissent en fonction de cette perception (voir figure II.2).

Ce modèle d'agent est une métaphore du modèle «fourmi», il s'appuie sur les sciences de la vie et l'intelligence collective.

Dans la même catégorie que les agents réactifs, mais avec davantage de rationalité, se trouvent les agents hédoniques. Ces agents apprennent, par auto renforcement, à modifier leur comportement afin d'augmenter leur "plaisir ou satisfaction ". Ils sont capables d'anticipations "hédoniques" et d'adaptation lente à partir de leur expérience historique, ce qui suppose un niveau de rationalité plus élevé que l'agent purement réactif.

Les agents réactifs, traitent généralement des informations quantitatives tout en utilisant des calculs élémentaires ou d'optimisation. Ils peuvent être construits par des réseaux connexionnistes comme les Réseaux de Neurones (RN) ou en utilisant des simples algorithmes de calcul comme les Algorithmes Génétiques (AG).

Contrairement aux agents réactifs, les agents cognitifs sont beaucoup plus complexes et plus difficiles à mettre en œuvre. Chaque agent se fonde sur ses

propres compétences de façon isolée pour résoudre un problème, ce sont les principales limites de cette architecture.

Pour faire face aux inconvénients de ces deux modèles, les chercheurs ont combiné ces deux facettes, opposées mais complémentaires, de la conception des agents. Cette combinaison fait apparaître les agents hybrides.

II.4 .3 Agents hybrides

Les agents hybrides sont conçus pour combiner des capacités réactives à des capacités cognitives, ce qui leur permet d'adapter leur comportement en temps réel à l'évolution de l'environnement. Dans le modèle hybride, un agent est composé de plusieurs couches, rangées selon une hiérarchie. La plupart des architectures considèrent que trois couches suffisent amplement. Ainsi, au plus bas niveau de l'architecture, on retrouve habituellement une couche purement réactive, qui prend ses décisions en se basant sur des données brutes en provenance des senseurs. La couche intermédiaire fait abstraction des données brutes et travaille plutôt avec une vision des connaissances de l'environnement. Finalement, la couche supérieure se charge des aspects sociaux de l'environnement (communication, coopération, négociation), c'est-à-dire du raisonnement tenant compte des autres agents.

II.5 Différence entre un Agents et un Objets

Un **objet** est défini par un ensemble de services offerts (ses méthodes) qu'il ne peut refuser d'exécuter si un autre objet le lui demande. Les **objets** exécutent des tâches; ils n'ont ni but, ni recherche de satisfaction. En revanche les **agents**, bien plus élaborés, disposent d'objectifs qui leur donnent une autonomie de décision vis à vis des messages qu'ils reçoivent.

Les **objets** utilisent un mécanisme d'envoi de message qui se résume à un simple appel de méthode. Pour les **agents**, les interactions sont plus complexes et font intervenir des communications de haut niveau, où l'important est que l'agent décide par lui-même comment interagir et réagir aux messages qu'il reçoit.

Un **agent** est une entité capable de recevoir et d'émettre des messages. Ce comportement est minimaliste, et correspond d'ailleurs tout à fait à la définition d'un objet. Néanmoins, un objet est contraint de répondre aux requêtes qui lui sont soumises, contrairement à l'agent qui dispose de son propre libre arbitre.

Concrètement, les objets font ce qu'on leur demande, tandis qu'avec les agents, il faut négocier. Donc, il est important que les agents contrôlent eux-mêmes leur comportement et les ressources qu'ils possèdent. C'est ce qui caractérise leur autonomie, et d'une certaine manière leur donne conscience de leurs possibilités.

Néanmoins, le lien existant entre la notion d'objet et celle d'agent reste fort. Il n'y a pas de frontière nette : un objet peut être considéré comme un agent dont le langage d'expression se résume à l'emploi de mots clés correspondant à ses méthodes comme le montre le tableau ci-dessous [18].

	POO	**POA**
Unité de base	objet	agent
Paramètre définissant l'état de l'unité de base	pas de contraintes	croyances, décisions, obligations, habiletés
Processus de calcul	envoi de messages et méthodes pour la réponse	envoi de messages et méthodes pour la réponse
Types de messages	pas de contraintes	informer, demander, offrir, promettre, accepter, rejeter, …
Contraintes sur les méthodes	pas de contraintes	consistance, vérité, …

Tableau II .1 Programmation Orientée Objets versus Programmation Orientée Agents

II.6 Définition d'un Système Multi-agent

Wooldridge et Jennings [10], présente un SMA comme étant un ensemble d'agents en interaction afin de réaliser leurs buts ou d'accomplir leurs tâches. Les interactions peuvent être directes par l'intermédiaire des communications, comme elles peuvent être indirectes via l'action et la perception de l'environnement. Les interactions peuvent être mises en œuvre dans un but de :

- Coopération entre les agents, lorsqu'ils ont des buts communs.

- Coordination, pour éviter les conflits et tirer le maximum de profit de leurs interactions afin de réaliser leurs buts.
- Compétition, lorsque les agents ont des buts antagonistes.

II.6 .1 Avantages

L'utilisation des SMA présente une série d'avantages:

✓ **Système dynamique :** les SMA héritent des bénéfices de l'IA au niveau du traitement symbolique (au niveau des connaissances). En revanche, contrairement aux approches traditionnelles de l'Intelligence Artificielle qui simulent, dans une certaine mesure, les capacités du comportement humain, les SMA permettent de modéliser un ensemble d'agents qui interagissent [20].

✓ **Nombre important d'agents :** un grand nombre d'agents est au cœur du problème dans ce type de modélisation contrairement à la théorie des jeux où rarement plus de trois acteurs sont représentés.

✓ **Souplesse de l'outil informatique :** qui permet de modifier le comportement des agents, ajouter ou supprimer des actions possibles, étendre les informations disponibles à l'ensemble des agents à la différence des modèles traditionnellement utilisés en science économique.

✓ **Une résolution distribuée de problèmes :** il est possible de décomposer un problème en sous-parties de résoudre chacune de façon indépendante pour aboutir à une solution stable [19].

II.6 .2 Domaines d'application des SMA

Les systèmes multi-agents sont à l'intersection de plusieurs domaines scientifiques tels que l'information répartie, le génie logiciel, l'intelligence collective, l'intelligence artificielle. Par conséquent, ils font appel à plusieurs autres disciplines, telles que la sociologie, la psychologie sociale, les sciences cognitives, la biologie, etc. Il en découle que les SMA sont appliqués dans divers domaines, comme l'industrie, le commerce, le divertissement et la médecine. Parmi les applications industrielles nous pouvons citer le pilotage des Système de production, la télécommunication comme le contrôle de réseaux, le contrôle de trafic aérien et la gestion du trafic et du transport. Parmi les applications commerciales à base d'agent, on trouve la gestion de l'information, et le

commerce électronique. Un grand domaine d'application des SMA est celui des divertissements, comme dans les jeux, le théâtre interactif et la réalité virtuelle[20].

II.7 Interactions entre agent

La notion d'interaction est au centre de la problématique des SMA. Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement des futurs agents.

Les situations sont diverses et variées: l'aide d'un robot par un autre, l'échange de données entre serveurs informatiques, l'utilisation d'une imprimante par deux programmes simultanément, la répartition de charge sur plusieurs processeurs, etc. les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisations sociales.

II.7 .1 Communication entre agents

La communication dans les systèmes multi-agents est à la base des interactions et de la création des organisations dont nous avons parlé précédemment. Sans la communication, un agent n'est qu'un individu isolé, sourd et muet qui ne fait qu'agir sur lui-même. La communication permet aux agents d'échanger des informations, des demandes de services, etc. Elle permet aussi à un agent d'agir sur un autre en lui fournissant des informations qui auront pour conséquence la remise en question de son comportement ou encore en lui demandant de modifier son comportement [21].

La communication entre agents peut revêtir diverses formes. Il existe deux principaux langages de communication KQML et ACL qui ont été abordés afin de normaliser la communication entre agents.

• KQML (Knowledge Query Manipulation Language) développe en 1993 par le consortium DARPA-KSE (Knowledge Sharing Effort).

• ACL (Agent Communication Language) proposé en 1997 par la FIPA (Foundation for Intelligent Physical Agents).

KQML et ACL se distinguent au niveau de la sémantique des actes du langage utilisé.

En effet, le langage ACL fortement inspiré des travaux de KQML, propose un langage auquel s'ajoute la définition de protocoles d'interactions.

Lorsque deux agents vont communiquer, leur objectif est d'échanger des informations mais avant tout de se comprendre. Donc, quel que soit le langage ou la forme de communication, l'importance réside dans la possibilité pour un agent de pouvoir comprendre les autres agents. Dans la majorité des cas, la communication se fait par envoi de messages et parfois par envoi de signaux ou stimuli dans l'environnement.

II.7.2 Similitudes et différences entre KQML et FIPA-ACL

KQML et FIPA-ACL sont presque identiques en ce qui concerne leurs concepts de base et les principes qu'ils observent. Les deux langages diffèrent principalement dans les détails de leurs cadres sémantiques. Dans un sens, cette différence est substantielle :

- il n'est pas possible de proposer une traduction systématique entre les performatives de KQML et celles complètement équivalentes de FIPA, ou vice-versa.
- les différences inéluctables pourraient avoir peu d'importance pour les programmeurs d'agents intelligents, si leurs agents ne sont pas de véritables agents BDI.

Les deux langages ont la même syntaxe. C'est-à-dire, un message de KQML et un message de FIPA-ACL sont syntaxiquement identique excepté, naturellement, dans leurs différences sur les noms des primitives de communication.

Sémantiquement, les deux langages diffèrent au niveau de ce qui constitue la description de la sémantique :

- pré-conditions, post-conditions, et conditions d'accomplissement pour KQML.
- pré-conditions de faisabilité et effets rationnels pour FIPA-ACL.

Ils diffèrent également au niveau du choix et des définitions des modalités qu'ils utilisent (le langage employé pour décrire les états des agents).

II.8 Agent BDI

L'idée phare de l'approche BDI est de décrire l'état interne des agents, dit *agents BDI*, en termes d'attitudes mentales, et aussi de définir une architecture de contrôle grâce à laquelle l'agent peut sélectionner son plan. Le mécanisme de raisonnement d'un agent BDI, déclenché par des événements perçus, se base sur les attitudes mentales de celui-ci qui sont représentées par les Croyances (Beliefs), les Désirs (Desires) et les Intentions (Intentions) :

• Les croyances expriment ce que l'agent croît sur l'état courant de son environnement.

• Les désirs (ou buts) sont une notion abstraite spécifiant les préférences de l'agent. Leur caractéristique principale est qu'un agent peut avoir des désirs inconsistants et qu'il n'a donc pas à croire que ses désirs sont réalisables.

• Les intentions représentent les désirs que l'agent s'engage à réaliser. Cependant, étant limité par ses ressources, l'agent peut ne pas poursuivre tous ses désirs même si ces derniers sont consistants. Il est nécessaire alors qu'il choisisse un certain nombre de désirs pour lesquels il s'engage à réaliser.

L'agent a une représentation explicite de ses croyances, désirs et intentions. On dénote par ***B*** l'ensemble des croyances de l'agent, par ***D*** l'ensemble de ses désirs, et par ***I*** l'ensemble de ses intentions. Les ensembles *B, D* et *I* peuvent être représentés au moyen de divers modèles de représentation de connaissances, par exemple en utilisant la logique des prédicats du premier ordre, la logique d'ordre supérieur, le modèle des règles de production, ou bien comme de simples structures de données [22].

Afin de bien comprendre la théorie du paradigme BDI proposée pour la première fois par Bratman[6], voici un scénario réaliste qui l'explique :

"L'agent Kamel a la croyance que, si quelqu'un passe son temps à étudier, cette personne peut faire une thèse de doctorat. En plus, Kamel a le désir de voyager beaucoup, de faire une thèse de doctorat et d'obtenir un poste d'assistant à l'université.

[6] **Michael E. Bratman**(né le 25 Juillet, 1945) est un philosophe américain Professeur à l'École des sciences humaines et professeur de philosophie à l'Université de Stanford

Le désir de voyager beaucoup n'est pas consistant avec les deux autres, et Kamel, après réflexion, décide de choisir, parmi ces désirs inconsistants, les deux derniers. Comme il se rend compte qu'il ne peut pas réaliser ses deux désirs à la fois, il décide de faire d'abord une thèse de doctorat.

En ce moment, Kamel a l'intention de faire une thèse et, normalement, il va utiliser tous ses moyens pour y parvenir. Il serait irrationnel de la part de Kamel, une fois sa décision prise, d'utiliser son temps et son énergie, notamment ses moyens, pour voyager autour du monde. En fixant ses intentions, Kamel a moins de choix à considérer car il a renoncé à faire le tour des agences de voyage pour trouver l'offre de voyage qui le satisferait au mieux."

II.9 Caractérisation des agents BDI

De nombreuses architectures multi-agents ont été proposées. Certaines d'entre elles, comme les machines à états finis ou les architectures motivationnelles, peuvent être très utiles pour la définition d'agents simples. Cependant, elles ne sont pas adaptées à la définition d'agents cognitifs complexes, connus pour leur pro-activité et leur anticipation, car leur pouvoir de représentation est très limité.

Le paradigme Belief-Desire-Intention (BDI) est l'une des approches majeures dédiées à la conception et à la construction des agents rationnels ainsi que des systèmes multi-agents.

Parmi les applications qui exploitent ces concepts, citons le système du contrôle du trafic aérien OASIS à l'aéroport de Sydney en Australie et le système du management de business SPOC [23].

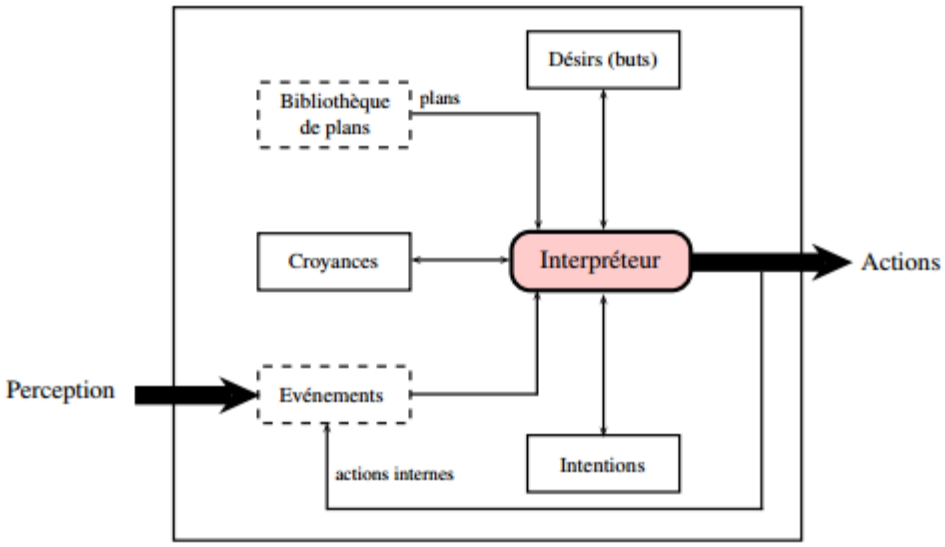

Figure II.3: L'architecture d'un agent BDI

II.10 Architecture des agents BDI

Le mécanisme de raisonnement de l'agent, basé sur ses Croyances (B), ses Désirs(D) et ses Intentions (I), est déclenché par les événements (*Evt*) perçus par celui-ci lui permettant de produire des plans d'actions consistants. Ces plans sont extraits à partir d'une *bibliothèque de plans partiels* (*Library of plans*), appelée *LibP*.

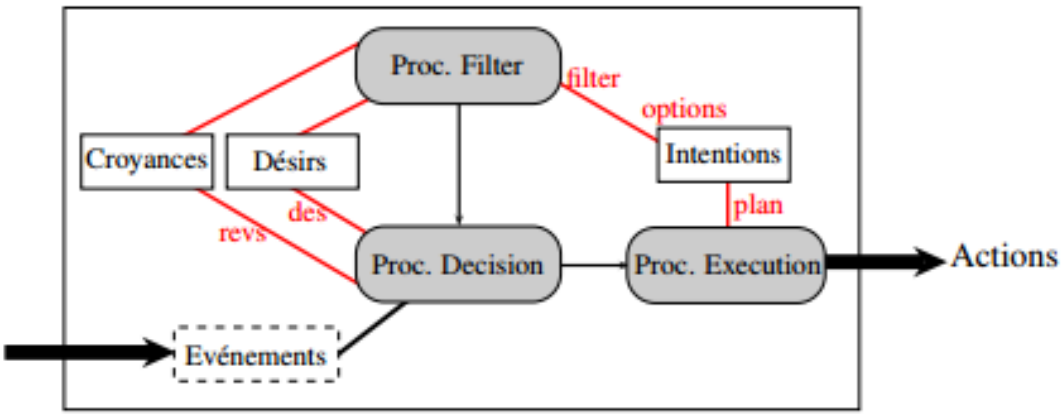

Figure II.4: Une vue fonctionnelle de l'interpréteur BDI

La figure II.4 illustre le schéma de fonctionnement de l'interpréteur associé à l'agent BDI, les fonctions suivantes décrivent son mécanisme de raisonnement :

• ***revs***: est la fonction de révision des croyances de l'agent déclenchée à partir des nouvelles perceptions.

• ***des*** : est la fonction responsable de la mise à jour des désirs de l'agent si ses croyances ou ses intentions changent afin de maintenir la consistance des désirs sélectionnés.

•***filter***: est la fonction qui décide des intentions à poursuivre parmi les options possibles, prenant en compte les nouvelles opportunités.

• ***options***: est la fonction qui associe à chaque intention de l'agent, des plans partiels dédiés en utilisant la bibliothèque de plans *LibP*.

• ***plan***: est la fonction qui permet de construire un plan exécutable à partir de plans partiels en utilisant la fonction *options*.

L'algorithme suivant conçu par Michael Wooldrige[7], décrit le mécanisme de raisonnement de l'agent BDI.

Soient $\mathbf{B_0}$, $\mathbf{D_0}$ et $\mathbf{I_0}$ les croyances, désirs et intentions initiales de l'agent.

II.11 Algorithme de contrôle d'agent BDI

1 **B** =$\mathbf{B_0}$

2 **D** =$\mathbf{D_0}$

3 **I** = $\mathbf{I_0}$

4 **répéter**

- 4.1 obtenir nouvelles perceptions **p**
- 4.2 **B** = ***revc***(**B**, **p**)
- 4.3 **I** = ***options***(**D**, **I**)
- 4.4 **D** = ***des***(**B**, **D** ,**I**)
- 4.5 **I** = ***filtre***(**B**, **D**, **I**)
- 4.6 **PE** = ***plan***(**B**, **I**)
- 4.7 exécuter(**PE)**

jusqu'à ce que l'agent soit arrêté

fin

[7] Chef de département et professeur d'informatique au département des sciences informatiques de l'Université d'Oxford, et Senior Research Fellow à Hertford College Université

II.12 Conclusion

L'avènement de l'Intelligence Artificielle et la conception des SMA ont beaucoup impactés le développement de systèmes complexes. Les SMA proposent, en effet, des architectures évoluées qui permettent de pallier les insuffisances de ces systèmes d'une façon générale, Ils sont des thèmes de recherche en cours d'exploration. Ils font intervenir plusieurs domaines de recherche tels que les systèmes répartis, la biologie, l'IA, la psychologie cognitive et la sociologie.

L'amélioration se traduit principalement en termes de coût et de simplicité de conception. Ils ont l'avantage de supporter des modèles d'interaction comme : la coopération, la coordination et la négociation.

CHAPITRE III

Implémentation de l’application

III.1 Introduction

Les systèmes à grand échelle peuvent être classifiés sous formes de plusieurs paradigmes informatiques telles que : Clusters, grille de calcul, P2P, et plus récemment Cloud computing. Ce dernier terme désigne une infrastructure constituée de milliers de ressources de calcul et de stockage qui changent avec le temps.

Le problème d'ordonnancement des tâches étant un problème relativement ancien. Il existe de nombreux algorithmes d'ordonnancement, allant de schémas statiques (ou encore "hors-ligne", lorsque le système dispose de toutes les tâches à allouer dès le début) très simples aux algorithmes dynamiques (ou encore "en-ligne", lorsque l'ordonnancement suit le changement du système ainsi que l'arrivée des tâches) les plus complexes.

La majorité des algorithmes d'ordonnancement des tâches dans les Clouds sont de nature centralisée, qui malgré leurs efficacités ils ne sont pas aptes à réagir rapidement sur les plates-formes à grande échelle lorsque certains nœuds se trouvent surutilisés, ce qui risque de conduire à une violation des niveaux de qualité de service. Une façon de traiter ce problème consiste à décentraliser l'ordonnancement.

Plusieurs approches ont été proposées pour décentraliser l'ordonnancement, parmi elles l'approche Multi-agents, car elles recourent à l'installation des agents logiciel sur un ensemble des nœuds de l'infrastructure à gérer, le travail d'ordonnancement étant réparti entre les agents.

Nous présentons dans ce travail, une nouvelle approche distribuée d'ordonnancement basée sur l'architecture multi-agents. Nous nous somme focalisés sur la phase de prise de décision dans l'ordonnancement des tâches qui nécessite d'effectuer plusieurs choix. La prise de décision est conçue selon une architecture BDI (Belief-Desire-Intention) qui permet de raisonner, de collaborer et de prendre des décisions d'adaptation dirigées par des buts en tenant compte de l'état de l'environnement qui change avec le temps.

Notre travail vise essentiellement à concevoir un algorithme d'ordonnancement de tâches dans le Cloud computing. Cet algorithme vise à optimiser (selon la stratégie de l'utilisateur) plusieurs métriques notamment l'équilibre de charge, le coût de

communication, le délai d'exécution, le temps d'attente et de réponse, la qualité de service (QoS), l'économie d'énergie et réduire le coût d'exécution des tâches.

III.2 Environnement de développement

Dans la réalisation de ce projet, nous avons employé le langage JAVA sous l'IDE Netbeans, ainsi que la plate-forme SMA JADE.

III.2.1 Java

Le langage Java est aujourd'hui largement connu et plébiscité par la communauté informatique.

Le but ici n'est donc pas de revenir sur ce qu'est Java en faisant une présentation générale, mais plutôt de voir en quoi il est particulièrement adapté aux développements de systèmes multi-agents. En premier lieu, Java est un langage objet. Les agents pouvant être considérés comme des objets avancés, ce type de langage est le mieux adapté pour les développer. De plus, la modularité apportée par les langages objet devient rapidement indispensable dés lorsqu'on modélise des agents dont l'architecture est un peu complexe. Par ailleurs, un des principes fondamentaux des systèmes multi-agents est l'hétérogénéité des architectures et des technologies pouvant être employées pour le développement et l'exécution des agents. Java est un langage semi-interprété qui est exécuté par une machine virtuelle (la JVM) déclinée sur de nombreuses plates-formes (Linux, AIX, Mac OS, Solaris, Windows, . . .). La portabilité du code que cela apporte permet ainsi de déployer un système multi-agents à travers un réseau hétérogène de machines.

III.2.2 NetBeans IDE 8.1

Pour notre choix de l'environnement de développement Java, nous avons opté sur l'utilisation de NetBeans, il est un environnement de développement intégré (IDE) open source. Il est développé par Sun et se trouve sous licence CDDL (Common Development and Distribution License).

En plus de Java, il propose tous les outils nécessaires à la création d'applications professionnelles pour les particuliers, les entreprises. NetBeans IDE est facile à installer et à utiliser.

Il comprend toutes les caractéristiques d'un IDE moderne (coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web, etc...).

III.2.3 JADE

La plate-forme SMA JADE est un outil développé par CSELT[8] en Java fournissant des bibliothèques de classes pour le développement d'agents et permettant d'animer ces agents au sein d'une plate-forme SMA. Cet outil est utilisé par une grande partie de la communauté multi-agents et prend en charge certaines caractéristiques essentielles aux multi-agents tels que la communication et la concurrence sans pour autant contraindre exagérément la structure de l'agent. Cet outil qui a plus l'avantage d'être gratuit, très utilisé, aussi abondamment documenté et de nombreux exemples sont disponibles. JADE supporte une distribution sur plusieurs machines ce qui peut être intéressant pour répartir les coûts de calcul parfois lourd. Il dispose d'une interface permettant le contrôle et la supervision des agents. Enfin, et c'est ce qui nous intéresse ici, il est compatible FIPA-ACL.

III.3 Architecture globale de notre application

Notre contribution dans le cadre de ce PFE est la réalisation d'une application d'ordonnancement basée sur les systèmes multi agents et de la simulation de l'exécution des tâches dans le cloud computing.

Notre application est constituée de deux types d'agents (agents **BDI** et agents **Hosts**) qui coopèrent entre eux pour satisfaire les demandes de services des clients, et un seul agent Générateur de tâche.

III.3.1 Agent BDI :

C'est l'Agent le plus important de notre application, il représente sa partie intelligente. Comme son nom l'indique, ce module réagi en fonction de l'état présent, il doit prendre des décisions en temps réel pour la gestion de ses

[8]Pour *Centro Studi E Laboratori Telecomunicazioni*, laboratoire de Telecom Italia (Tilab).

ressources, pallier aux dysfonctionnements, et répondre aux demandes des autres agents.

La figure suivante représente l'architecture de l'agent BDI.

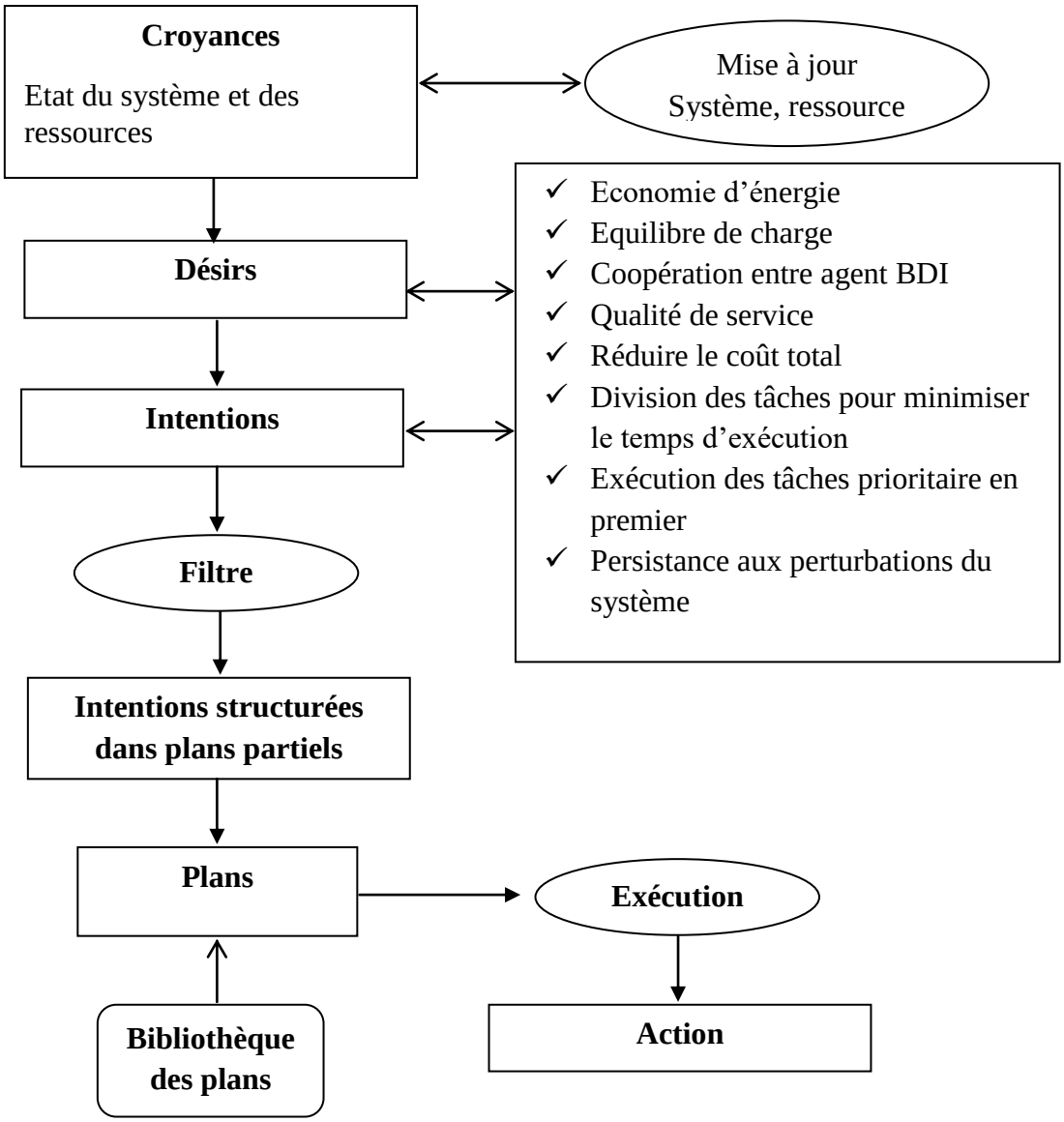

Figure III.1 Architecture de l'agent BDI

III.3.2 Agent Host :

Il est responsable de la simulation de l'exécution des tâches et renvoie les résultats à l'Agent BDI.

III.3.3 Agent Générateur de tâches :

C'est un Agent de type GUI (avec interface graphique), qui vise à :

- Générer des tâches et les affecter aux agents BDI.

- Afficher les résultats de la simulation.

La figure suivante montre l'architecture de notre application.

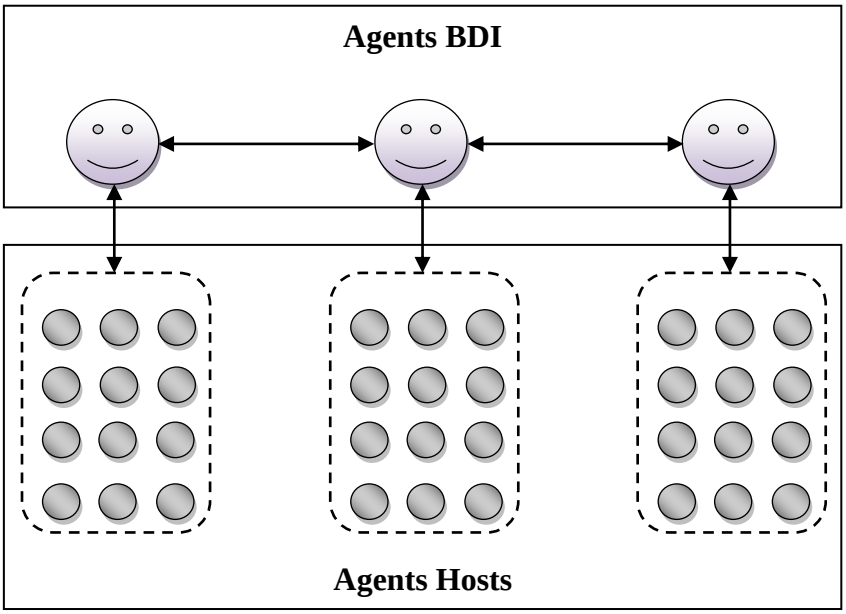

Figure III.2 Architecture globale de notre application

III.4 Interface graphique de configuration du système

La figure suivante montre l'interface graphique de configuration qui se compose de cinq onglets (onglet Agent BDI, onglet Hosts de calcul, onglet Hosts de stockage, onglet Tâche de calcul et onglet Tâche de stockage) permettant à l'utilisateur de configurer l'environnement de la simulation.

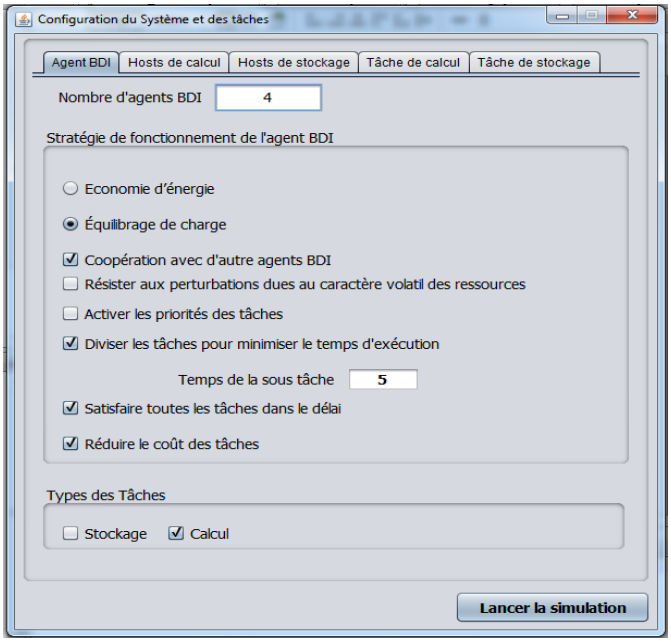

Figure III.3 : Interface graphique de configuration de l'agent BDI

III.4.1 Onglet Agent BDI :

Cet onglet permet à l'utilisateur de choisir la stratégie de fonctionnement de l'agent BDI et les types des tâches à générer.

Stratégie de fonctionnement de l'Agent BDI : L'agent BDI fonctionne selon deux plans principaux :

- **Economie d'énergie :** L'algorithme d'ordonnancement mis en œuvre par cette stratégie utilise au maximum les ressources d'un nombre restreint de nœuds. La consolidation permet d'utiliser moins de nœuds, et donc de recourir à une climatisation moins puissante. Ceci entraîne une consommation électrique plus faible (vu qu'il y a moins de nœuds à alimenter et à refroidir), d'où un coût financier moindre. En somme, cela permet de réaliser des économies d'échelle.

L'algorithme suivant explique le fonctionnement de ce plan.

```
// F : file d'attente des tâches
// T : liste des PC
// Simulation : teste la faisabilité de la tâche par le PC
// Transfert : transfert la tâche au PC
// Minimum(T) : retourne le pc ayant le plus de charge de travail (RAM, CPU, DISQUE, File d'attente) dans la liste T.
Début
    Tanque(F  != vide)faire
        T ←liste_PC
        Tâche ←F[1]
        Trouve ← faux
        Tanque(trouve = faux&& T != vide)faire
            PC← Minimum(T)
            resultat←Simulation(PC, Tâche)
            si(resultat = vrai )alors
                Transfert(Tâche, PC)
                Trouve ← vrai
            Sinon
                T ←T – {PC}
                si (T = vide) alors
                    Rejeter(Tâche)
                finsi
            finsi
        fintanque
        F ←F – {Tâche}
    fintanque
fin
```

- **Equilibre de charge :** Cette stratégie vise à garantir la pleine utilisation des ressources disponibles et de s'assurer que les charges sont distribuées de façon

équilibrée, c.-à-d. que les tâches ne surchargent pas une ressource tandis que d'autres ressources restées sous chargées. Dans ce cas, les nœuds les moins chargés sont utilisés en priorité pour exécuter les tâches.

L'algorithme suivant explique le fonctionnement du plan équilibre de charge.

```
//Maximiser(T) : choisir le pc ayant le moins de charge de travail (RAM, CPU, DISQUE, File d'attente) dans la liste T
début
    Tanque(F != vide)faire
        T ←liste_PC
        Tâche ←F[1]
        PC ←Maximiser (T)
        résultat ←Simulation(Tâche, PC)
        si(résultat = vrai)alors
            Transfert(Tâche, PC)
        sinon
            Rejeter(Tâche)
        finsi
        F ←F – {Tâche}
    fintanque
fin
```

Les deux plans principaux peuvent être s'accompagner avec d'autre plans secondaires.

✓ **Coopération avec d'autres agents BDI :** Cette option permet à l'agent BDI de faire une négociation avec les autres agents BDI pour choisir la meilleure décision globale.

L'algorithme suivant explique le fonctionnement du plan coopération.

```
// Négociation : négocier avec les autres agents BDI et renvoyer le meilleur PC en fonction de la stratégie pour exécuter la tâche, ou NULL s'il n'y a pas

début
    Tanque(F != vide)faire
        Tâche ←F[1]
        PC ←Négociation(les agents BDI, Tâche)
        si (PC != NULL)alors
            Transfert(Tâche, PC)
        sinon
            Rejeter(Tâche)
        finsi
        F ←F-{Tâche}
    fintanque
fin
```

✓ **Résister aux perturbations dues au caractère volatil des ressources :** Cette option permet à l'agent BDI de déléguer aux autres machines, les tâches qui ont été déjà affectées à une machine qui vient de tomber en panne.

L'algorithme suivant explique le fonctionnement de ce plan**.**

```
// T : liste des PC
// Persiste(PC) : renvoie vrai si le PC est en marche, faux si le PC est en panne
// Récupérer(PC) : récupère la liste des tâches (non achevées) déjà affectées au PC (l'agent
BDI sauvegarde toutes les tâches acceptées et les PC qu'ils les traitent).
// Retransférer (liste) : retransfert les tâche de liste aux PC
// Dormir(p) : faire une pause d'une durée p
début
    Tanque(vrai)faire
        Pour(n allant de 1 à Taille(T)) faire
            PC ← T[n]
            si(Persiste(PC) = faux)alors
                Liste ← Récupérer tâches non achevées (PC)
                Retransférer tâches (liste)
                T ←T – {PC}
            finsi
        Fin pour
        Dormir (p)
    fintanque
fin
```

✓ **Activer les priorités des tâches :** Cette option permet de favoriser les tâches ayant la priorité la plus élevée.

L'algorithme suivant explique l'insertion de la tâche selon sa priorité.

```
// Liste : la file d'attente
// Temps(tâche) : retourne le temps d'exécution de la tâche
// Taille(file d'attente) : retourne la taille de la liste file d'attente
// Priorité(tâche) : retourne la priorité de la tâche
// Liste[i] : représente la ième tâche dans la file d'attente
// Insérer(tâche, i) : insère la tâche à la position i dans la file d'attente
// Cet algorithme s'exécute au niveau de l'agent BDI et aussi au niveau de l'agent Host
  Début
    Pour n allant de 1 à Taille(liste)
        si (Priorite(tâche) >Priorité(liste[n])) alors
            Insérer(tâche, n)
            Retour
        finSi
    Fin pour
    Insérer(tâche)
    Retour
  Fin
```

✓ **Diviser les tâches pour minimiser le temps d'exécution :** Cette option permet de diviser une tâche en plusieurs sous tâches de même taille et les exécuter en parallèle.

L'algorithme suivant explique la division d'une tâche en plusieurs sous tâches.

```
// Temps(tâche) : retourne le temps d'exécution de la tâche
// fraction : temps de la sous tâche
début
    temps ← Temps(Tâche)
    Tanque(temps > 0)faire
        si(temps > fraction) alors
            Temps(sous tâche) ←  fraction
            temps =temps – fraction
        Sinon
            Temps(sous tache)  = temps
            Temps = 0
        finsi
    fintanque
fin
```

Le fonctionnement de l'agent BDI est découpé en trois phases : phase de réception de tâches, phase de la prise de décision et phase de la simulation de l'exécution de tâches.

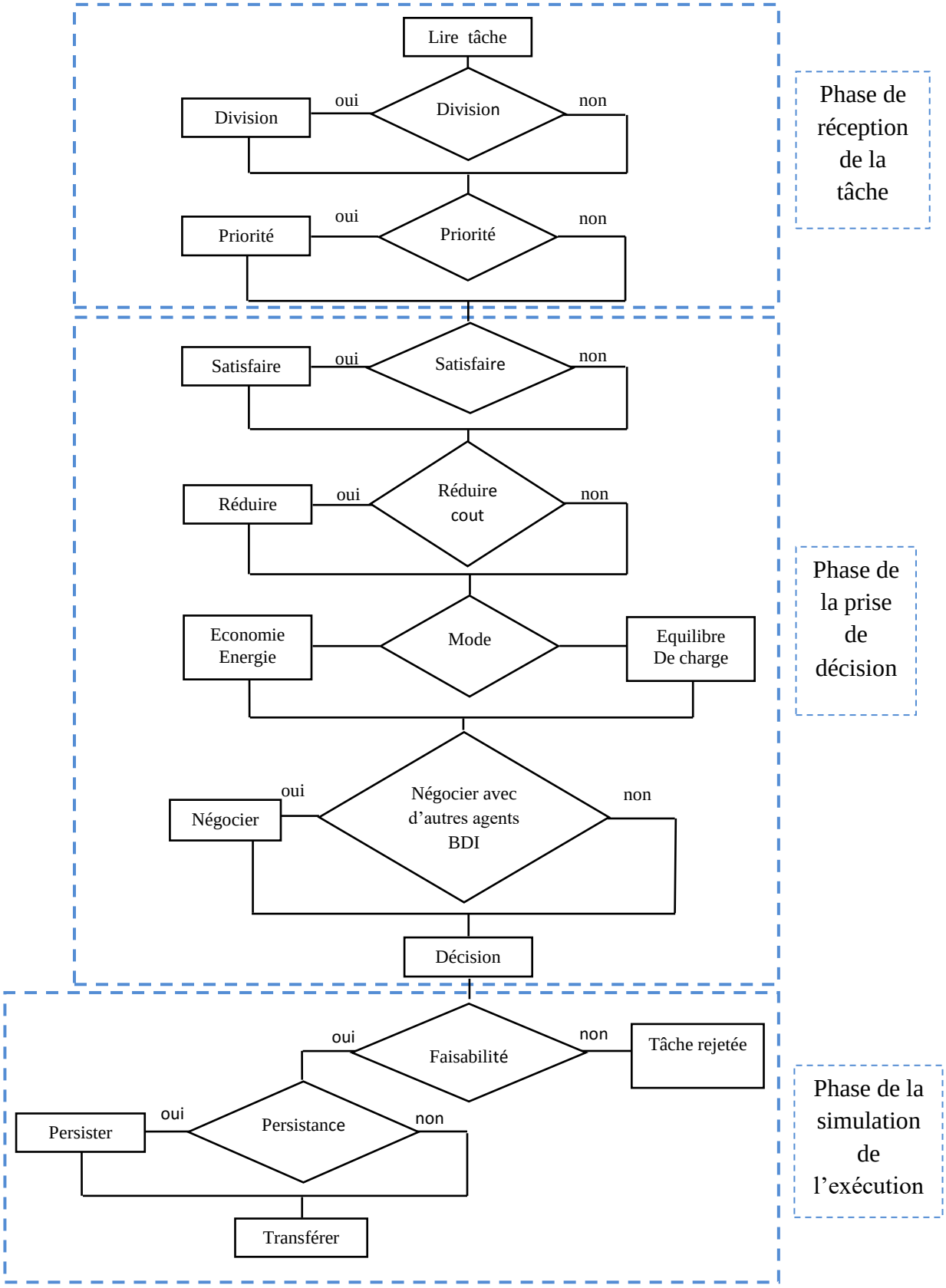

Figure III.4 : Organigramme de l'ordre d'exécution du plan

III.4.2 Onglet Agent Host de calcul et Host de stockage :

Chaque agent BDI est responsable d'un certain nombre de hosts de calcul et de hosts de stockage.

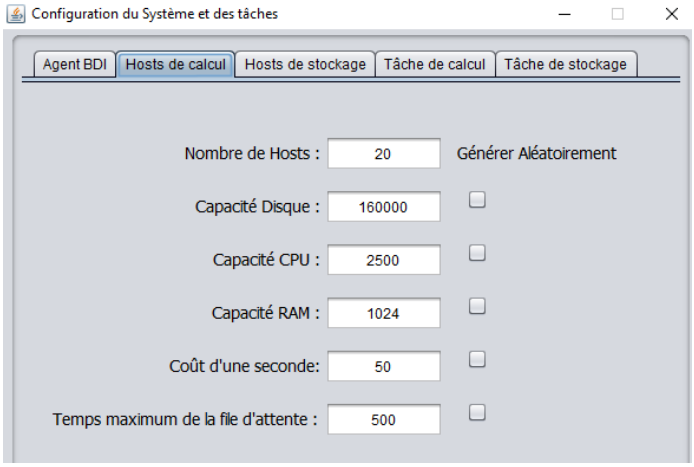

Figure III.5 : Interface du host de calcul et de host de stockage

III.4.3 Onglet Tâche de calcul :

La tâche de calcul est un ensemble d'instruction qui va être exécutées dans un host de calcul.

La figure suivante montre l'interface de la tâche de calcul.

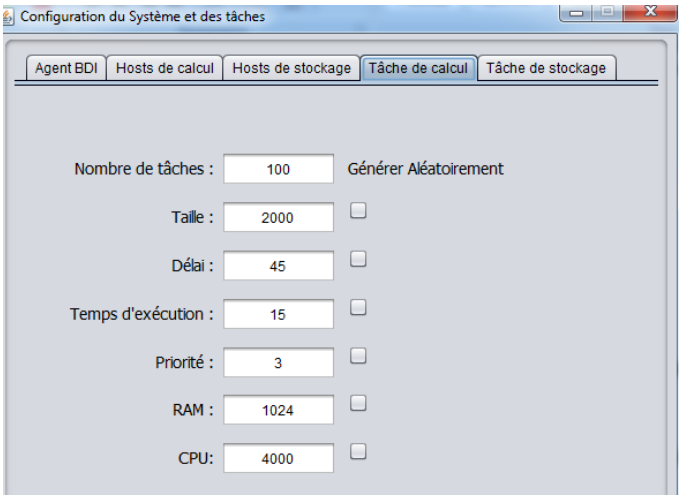

Figure III.6 : Interface de la tâche de calcul

Nombre de tâches : le nombre de tâche à générer.

Taille : la taille de la tâche.

Délai : temps de l'exécution de la tâche depuis sa création.

Temps d'exécution : le temps nécessaire pour exécuter la tâche.

Priorité : la priorité de la tâche.

RAM : l'espace ram de la ressource exigée pour satisfaire la tâche.

CPU : La vitesse minimale du CPU pour satisfaire la tâche.

L'utilisateur a le choix de fixer la valeur pour chaque propriété ou de cocher le champ Générer aléatoirement qui permet de générer des valeurs multiples et aléatoires.

III.4.4Onglet Tâche de stockage :

La tâche de stockage représente généralement des données qui vont être stockées sur un host de stockage.

La figure suivante montre l'interface de la tâche de stockage.

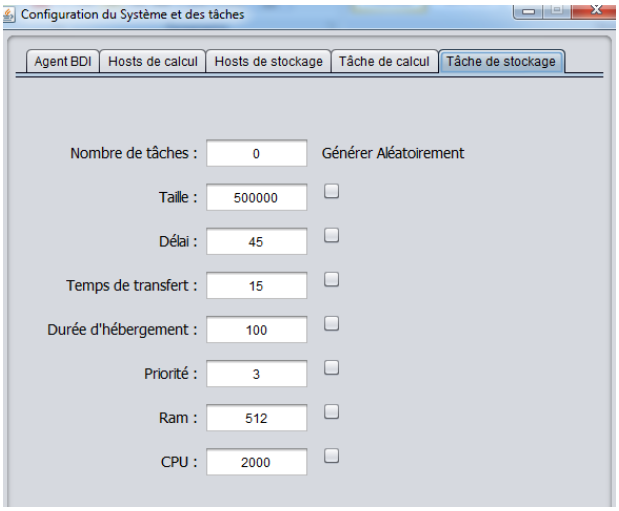

Figure III.7 : Interface de la tâche de stockage

Dans les hosts de stockage nous avons en plus :

Temps de transfert : c'est le temps nécessaire pour transférer la tâche vers le host.

Durée d'hébergement : représente la durée que doit occuper la tâche chez le host.

III.5 Simulation

III.5.1 Configuration du système :

Pour passer à l'étape de la simulation nous devons configurer notre système. L'étape de la configuration consiste à configurer l'infrastructure (les hosts de calcul et de stockage), ensuite configurer les tâches et finalement configurer l'agent BDI.

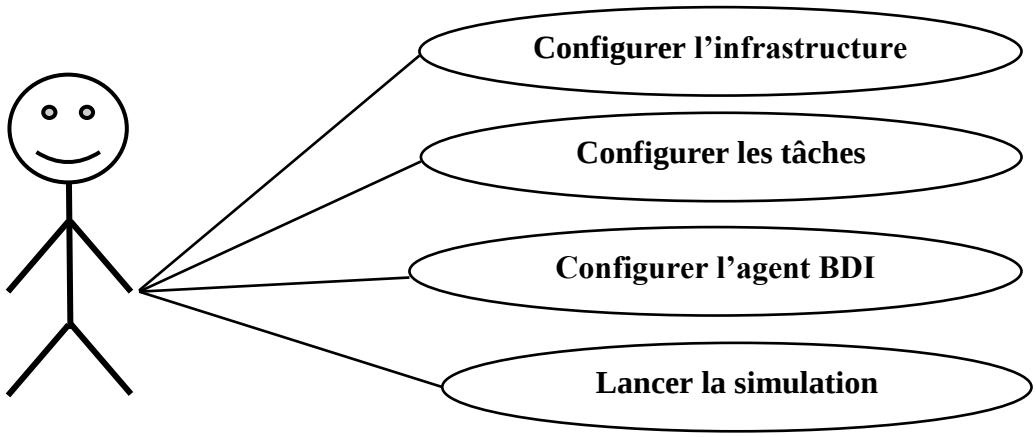

Figure III.8 : Configuration du système

III.5.2 Communication entre agents :

La figure suivante montre les messages échangés entre les agents.

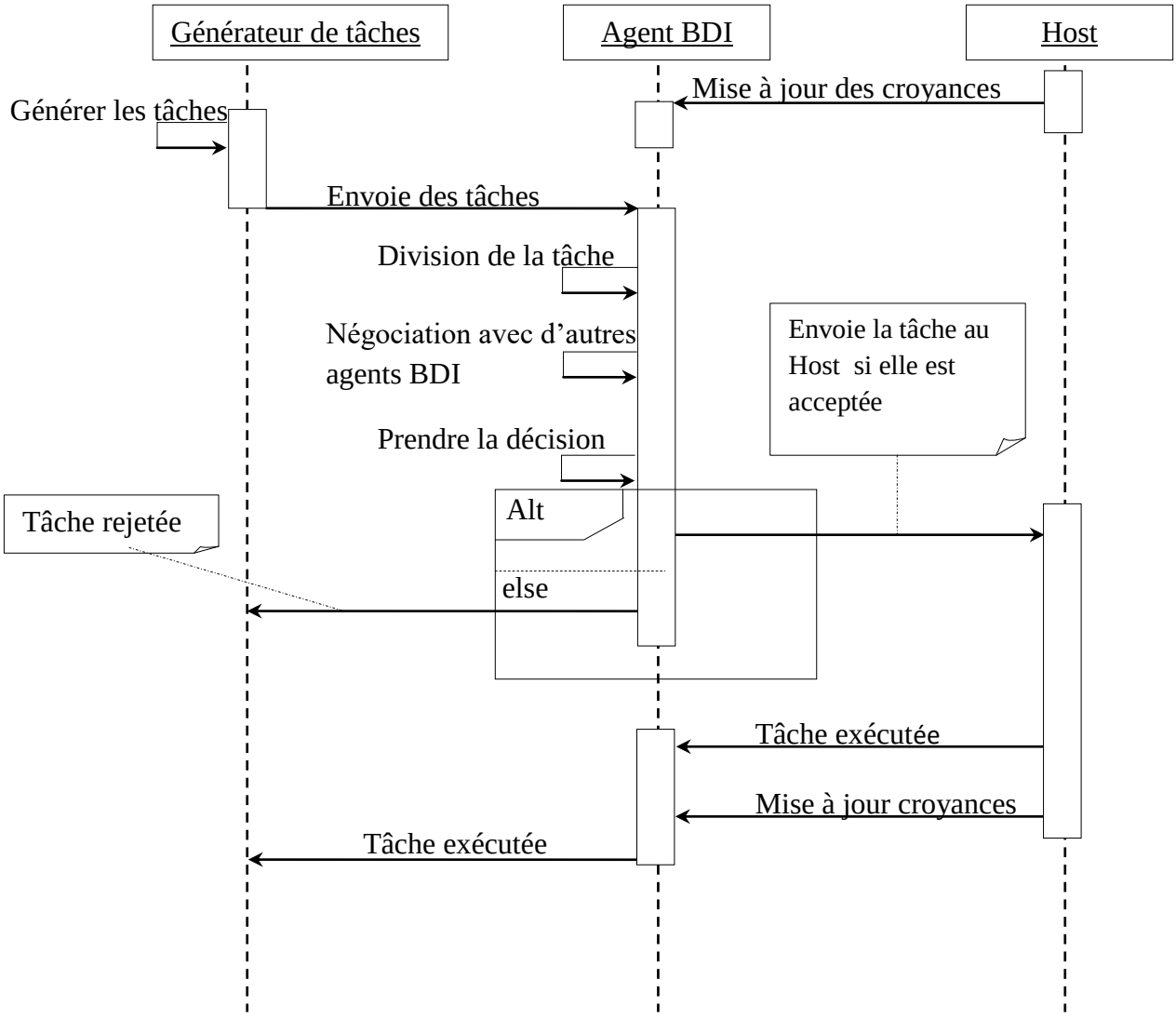

Figure III.9 : Diagramme de communication entre agents

III.6 Résultats et analyses :

Les différentes simulations et résultats obtenues ont été réalisés sur un PC Dual Core CPU 3 GHz, Ram 4 GB, Système d'exploitation Windows 7 32 bit.
Dans ce qui suit, nous allons montrer les résultats obtenus de notre application dans le cas d'un seul agent BDI.

Pour simplifier la présentation nous supposons que les hosts sont homogènes.

La configuration suivante est utilisée.

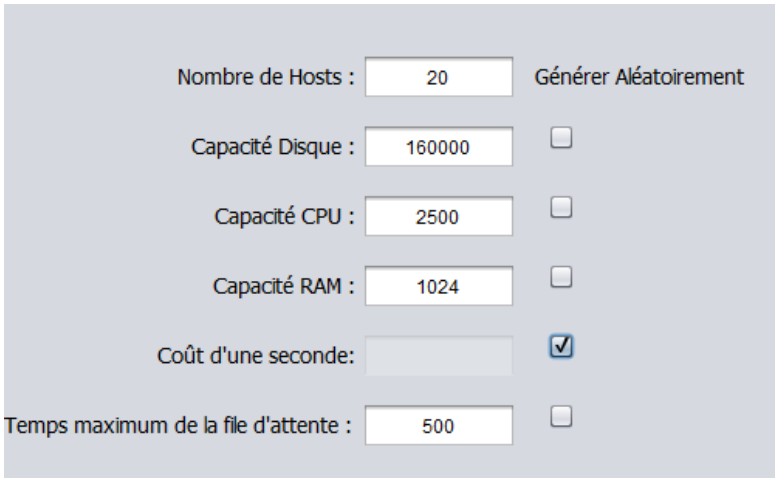

Figure III.10 : Configuration de hosts de calcul.

Par la suite nous avons créé 100, 200, 300, 400 et 500 tâches respectivement avec dans tous les cas un temps d'exécution égal à 15 secondes et un délai égal à 45 secondes.

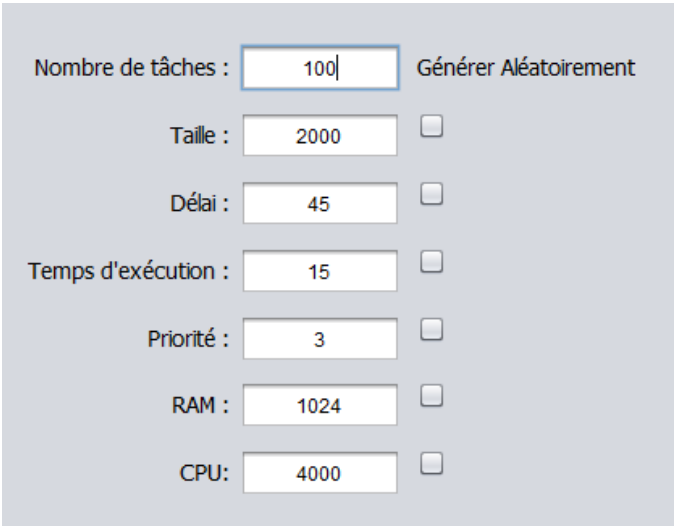

Figure III.11 : Configuration de tâches de calcul.

Nous avons appliqué les deux plans principaux (économie d'énergie et équilibre de charge) sur la configuration montrée ci-dessus et nous avons obtenu les résultats suivants :

Les deux figures suivantes montrent l'interface graphique de la simulation de l'exécution (pour un exemple de 100 tâches), elle est composée de deux onglets permettant à l'utilisateur de visionner la simulation.

L'onglet ressource montre l'état des hosts au moment de l'exécution.

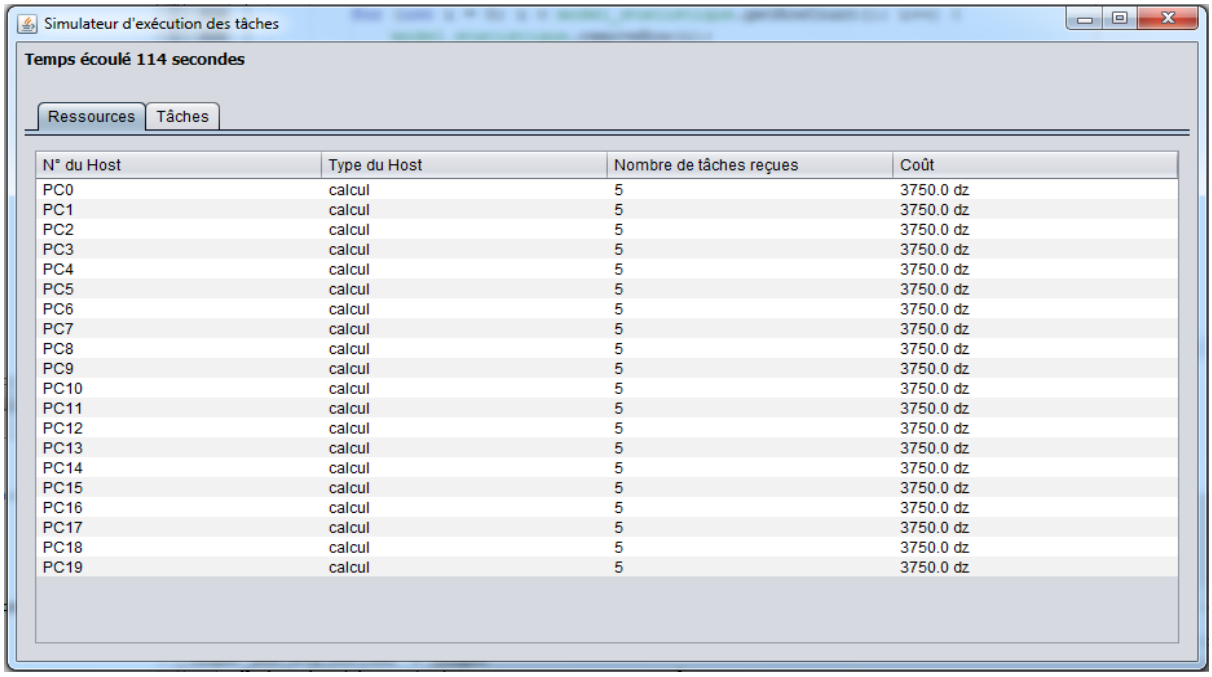

Figure III.12 : Etat des ressources.

L'onglet Tâches montre les informations relatives aux tâches.

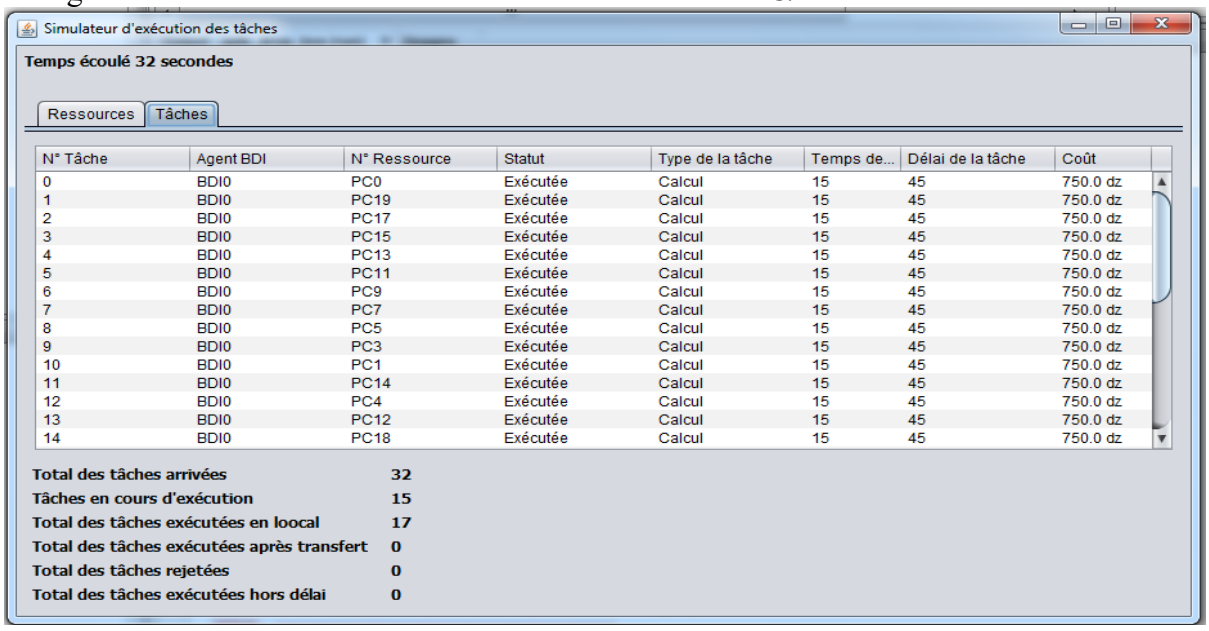

Figure III.13 : Etat des tâches.

La figure suivante montre le résultat récapitulatif de la simulation.

```
INFOS: ----------------------------------------
Agent container Container-3@192.168.1.10 is ready.
------------------------------------------------
mai 21, 2016 7:34:52 PM jade.core.PlatformManagerImpl$1 nodeAdded
INFOS: --- Node <Container-3> ALIVE ---
******** Plan : Eqilibrage de charge
Simulation de 100 tâches
Statégie employée Equilibre de charge/local
Temps de simulation 114 s
================================================================================
Nombre de tâches envoyées                       100
--------------------------------------------------------------------------------
Nombre de tâches exécutées                      100
--------------------------------------------------------------------------------
Nombre de tâches exécutée après transfert       0
--------------------------------------------------------------------------------
Nombre de tâches rejetées                       0
--------------------------------------------------------------------------------
Nombre de tâches dépassées le délai             0
================================================================================
Coût total de l'exécution 75000.0
Nombre des ressources inactives : 0
Nombre de tâches minimum pour une ressource : 5  Tâches
Nombre de tâches maximum pour une ressource : 5  Tâches
```

Figure III.14 : Résultat final de la simulation.

Le fichier log est un fichier qui représente tout l'historique du system comme le cheminement des tâches, les perturbations du système, le plan choisie …etc. Il est présenté comme suite :

Fichier Edition Format Affichage ?

```
N° 1 : 1464646756916 : [Tache créer le : 30/05/16 - 11:19:16][BDI1 : 30/05/16 - 11:19:16][PC26 : 30/05/16 - 11:19:16][début traitement à : 30/05/16 -
N° 2 : 1464646757932 : [Tache créer le : 30/05/16 - 11:19:17][BDI2 : 30/05/16 - 11:19:17][PC43 : 30/05/16 - 11:19:17][début traitement à : 30/05/16 -
N° 3 : 1464646758933 : [Tache créer le : 30/05/16 - 11:19:18][BDI1 : 30/05/16 - 11:19:18][PC24 : 30/05/16 - 11:19:18][début traitement à : 30/05/16 -
N° 4 : 1464646759934 : [Tache créer le : 30/05/16 - 11:19:19][BDI3 : 30/05/16 - 11:19:19][PC60 : 30/05/16 - 11:19:19][début traitement à : 30/05/16 -
N° 5 : 1464646760935 : [Tache créer le : 30/05/16 - 11:19:20][BDI2 : 30/05/16 - 11:19:20][PC47 : 30/05/16 - 11:19:20][début traitement à : 30/05/16 -
N° 6 : 1464646761989 : [Tache créer le : 30/05/16 - 11:19:21][BDI0 : 30/05/16 - 11:19:22][PC19 : 30/05/16 - 11:19:22][début traitement à : 30/05/16 -
N° 7 : 1464646762990 : [Tache créer le : 30/05/16 - 11:19:22][BDI1 : 30/05/16 - 11:19:22][PC37 : 30/05/16 - 11:19:22][début traitement à : 30/05/16 -
N° 8 : 1464646763992 : [Tache créer le : 30/05/16 - 11:19:23][BDI1 : 30/05/16 - 11:19:23][PC25 : 30/05/16 - 11:19:23][début traitement à : 30/05/16 -
N° 9 : 1464646764993 : [Tache créer le : 30/05/16 - 11:19:24][BDI2 : 30/05/16 - 11:19:24][PC46 : 30/05/16 - 11:19:24][début traitement à : 30/05/16 -
N° 10 : 1464646765995 : [Tache créer le : 30/05/16 - 11:19:25][BDI3 : 30/05/16 - 11:19:25][PC61 : 30/05/16 - 11:19:25][début traitement à : 30/05/16
N° 11 : 1464646767011 : [Tache créer le : 30/05/16 - 11:19:27][BDI0 : 30/05/16 - 11:19:27][PC4 : 30/05/16 - 11:19:27][début traitement à : 30/05/16 -
N° 12 : 1464646768012 : [Tache créer le : 30/05/16 - 11:19:28][BDI3 : 30/05/16 - 11:19:28][PC63 : 30/05/16 - 11:19:28][début traitement à : 30/05/16
N° 13 : 1464646769013 : [Tache créer le : 30/05/16 - 11:19:29][BDI1 : 30/05/16 - 11:19:29][PC21 : 30/05/16 - 11:19:29][début traitement à : 30/05/16
N° 14 : 1464646770014 : [Tache créer le : 30/05/16 - 11:19:30][BDI0 : 30/05/16 - 11:19:30][PC12 : 30/05/16 - 11:19:30][début traitement à : 30/05/16
```

Figure III.15 : Structure du fichier log.

Le tableau suivant récapitule les résultats relatifs au plan économie d'énergie et des plans secondaires.

Nombre de tâches	Critères	Economie d'énergie				
		Sans plus	**Plus Satisfaction**	**Plus réduire le coût**	**Plus Satisfaction et réduire le coût**	**Plus division tâches (5 s)**
100	Nombre de ressources inactives	17	6	17	6	17
	Nombre de tâches minimal pour une ressource	24	2	24	2	23
	Nombre de tâches maximal pour une ressource	39	9	39	9	39
	Nombre de tâches exécutées dans le délai	9	100	9	100	7
	Nombre de tâches exécutées hors délai	91	0	91	0	93
	Coût total	74250	70800	60000	70600	63450
	Temps de l'exécution	590	144	593	144	597
200	Nombre de ressources inactives	15	5	15	5	15
	Nombre de tâches minimal pour une ressource	30	6	30	7	30
	Nombre de tâches maximal pour une ressource	46	16	46	16	46
	Nombre de tâches exécutées dans le délai	15	200	15	200	11
	Nombre de tâches exécutées hors délai	185	0	185	0	189
	Coût total	153450	136050	120000	132450	132050
	Temps de l'exécution	699	244	697	243	698
300	Nombre de ressources inactives	13	5	13	5	13
	Nombre de tâches minimal pour une ressource	22	12	23	11	13
	Nombre de tâches maximal pour une ressource	53	23	53	23	32
	Nombre de tâches exécutées dans le délai	21	300	21	300	14
	Nombre de tâches exécutées hors délai	279	0	279	0	286
	Coût total	194850	218550	180000	200550	221199
	Temps de l'exécution	800	346	801	345	798

400	Nombre de ressources inactives	11	5	11	5	11
	Nombre de tâches minimal pour une ressource	1	18	3	17	1
	Nombre de tâches maximal pour une ressource	60	29	59	29	60
	Nombre de tâches exécutées dans le délai	25	400	27	400	13
	Nombre de tâches exécutées hors délai	375	0	363	0	387
	Coût total	293250	280650	240450	268350	289550
	Temps de l'exécution	901	447	899	448	903
500	Nombre de ressources inactives	11	4	10	5	10
	Nombre de tâches minimal pour une ressource	3	17	21	24	8
	Nombre de tâches maximal pour une ressource	52	36	58	36	66
	Nombre de tâches exécutées dans le délai	30	500	30	500	19
	Nombre de tâches exécutées hors délai	470	0	470	0	481
	Coût total	383400	381300	309450	336300	339800
	Temps de l'exécution	1003	547	1004	548	999

Tableau III.1 : Résultat de simulation du plan Economie d'énergie

Nous pouvons analyser ces résultats sous différents points de vue :

- Du point de vue Economie d'énergie : dans le plan**« Economie d'énergie »** et le plan **« Economie d'énergie plus réduire le coût »** le nombre de hosts inactifs est plus élevé.
- Du point de vue coût d'exécution : le plan « **Economie d'énergie plus réduire le coût »** offre le coût le plus bas.
- Du point de vue qualité de service : le plan « **Economie d'énergie avec satisfaction des tâches dans le délai »**et le plan« **Economie d'énergie avec satisfaction des tâches dans le délai plus réduire le coût »** garantissent une meilleure QoS.

Le tableau suivant récapitule les résultats relatifs au coût total de l'exécution.

Nombre de tâches	100	200	300	400	500
Economie d'énergie	74250	153450	194850	293250	383400
Economie d'énergie + Satisfaction des tâches dans le délai	70800	136050	218550	280650	381300
Economie d'énergie + Réduire le coût	60000	120000	180000	240450	309450
Economie d'énergie + Satisfaction des tâches dans le délai + Réduire le coût	70600	132450	200550	268350	336300
Economie d'énergie + Division des tâches	63450	132050	221199	289550	339800

Tableau III.2 : Résultat de simulation du plan Economie d'énergie (Coût total par nombre de tâches)

La figure suivante montre l'impact du nombre de tâches sur le coût total des tâches.

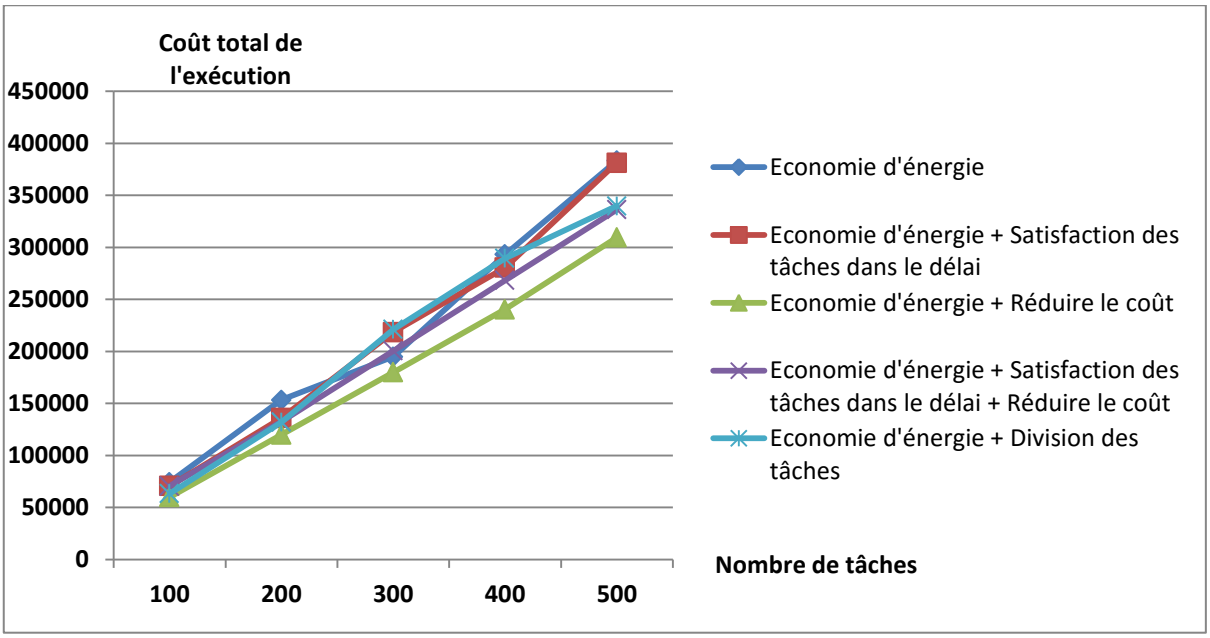

Figure III.16 : Courbe des coûts par nombre de tâches

Le plan **« Economie d'énergie plus réduire le coût »** offre un coût minimum, puisqu'il vise les ressources ayant le coût le plus bas.

Le tableau suivant récapitule les résultats relatifs au temps d'exécution.

Nombre de tâches	100	200	300	400	500
Economie d'énergie	590	699	800	901	1003
Economie d'énergie + Satisfaction des tâches dans le délai	144	244	346	447	547
Economie d'énergie + Réduire le coût	593	697	801	899	1004
Economie d'énergie + Satisfaction des tâches dans le délai + Réduire le coût	144	243	345	448	548
Economie d'énergie + Division des tâches	597	698	798	903	999

Tableau III.3 : Résultat de simulation du plan Economie d'énergie (Temps d'exécution par nombre de tâches)

La figure suivante montre l'impact du nombre de tâches sur le temps de l'exécution.

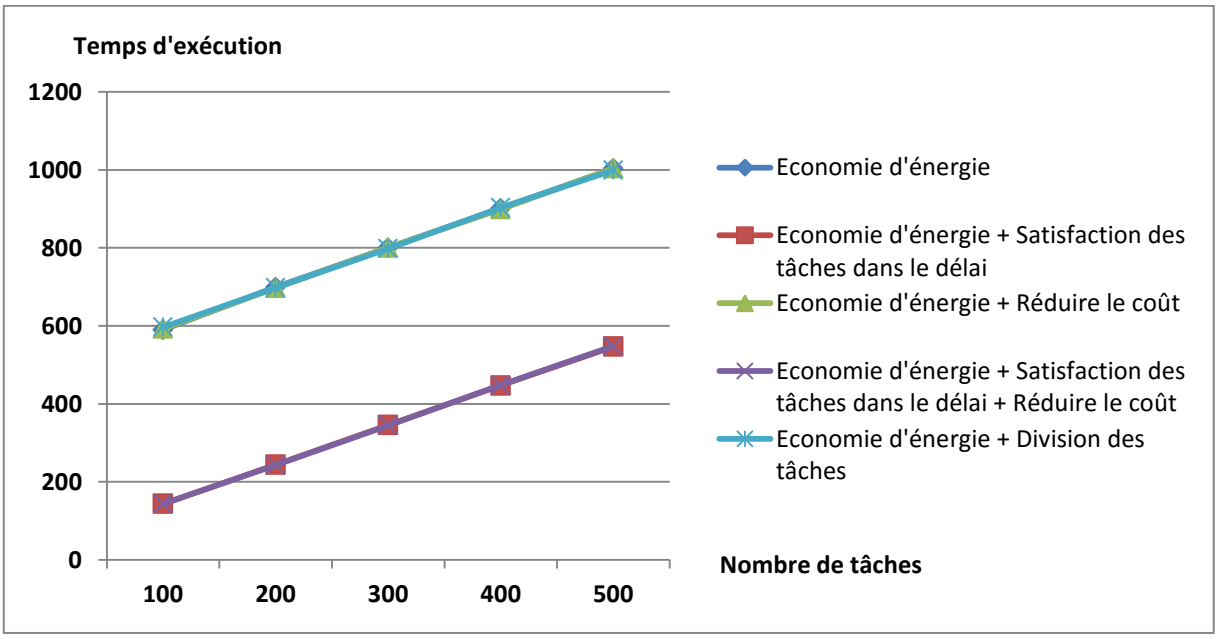

Figure III.17 : Courbe du temps d'exécution par nombre de tâches

Les plans avec « **satisfaction des tâches dans le délai** » terminent l'exécution de toutes les tâches en un temps réduit du fait qu'ils utilisent plus de ressources que les autres plans.

Le tableau suivant récapitule les résultats relatifs au plan équilibre de charge et des plans secondaires.

Nombre de tâches	**Critères**	**Equilibre de charge**				
		Sans plus	**Plus Satisfaction**	**Plus réduire le coût**	**Plus Satisfaction et réduire le coût**	**Plus division tâches (5 s)**
100	Nombre de ressources inactives	0	0	0	0	0
	Nombre de tâches minimal pour une ressource	5	5	5	5	5
	Nombre de tâches maximal pour une ressource	5	5	5	5	5
	Nombre de tâches exécutées dans le délai	100	100	40	100	100
	Nombre de tâches exécutées hors délai	0	0	60	0	0
	Coût total	73500	73500	60000	64200	73500
	Temps de l'exécution	115	114	198	142	104
200	Nombre de ressources inactives	0	0	0	0	0
	Nombre de tâches minimal pour une ressource	10	10	10	10	10
	Nombre de tâches maximal pour une ressource	10	10	10	10	10
	Nombre de tâches exécutées dans le délai	200	200	40	200	200
	Nombre de tâches exécutées hors délai	0	0	160	0	0
	Coût total	14700	14700	120000	131250	14700
	Temps de l'exécution	215	214	382	244	205
300	Nombre de ressources inactives	0	0	0	0	0
	Nombre de tâches minimal pour une ressource	15	15	15	15	15
	Nombre de tâches maximal pour une ressource	15	15	15	15	15

	Nombre de tâches exécutées dans le délai	300	300	40	300	300
	Nombre de tâches exécutées hors délai	0	0	260	0	0
	Coût total	220500	220500	180000	197850	220500
	Temps de l'exécution	317	314	573	344	304
400	Nombre de ressources inactives	0	0	0	0	0
	Nombre de tâches minimal pour une ressource	20	20	20	20	20
	Nombre de tâches maximal pour une ressource	20	20	20	20	20
	Nombre de tâches exécutées dans le délai	400	400	40	400	400
	Nombre de tâches exécutées hors délai	0	0	360	0	0
	Coût total	294000	294000	240000	266850	294000
	Temps de l'exécution	418	415	758	445	404
500	Nombre de ressources inactives	0	0	0	0	0
	Nombre de tâches minimal pour une ressource	25	25	25	25	25
	Nombre de tâches maximal pour une ressource	25	25	25	25	25
	Nombre de tâches exécutées dans le délai	0	0	12	0	0
	Nombre de tâches exécutées hors délai	0	0	460	0	0
	Coût total	367500	367500	300000	333600	367500
	Temps de l'exécution	518	515	954	545	505

Tableau III.4 : Résultat de simulation du plan Equilibre de charge

Les résultats montrent que le plan **« Equilibre de charge plus division »** garantit le temps d'exécution le plus court.

Du coté coût total d'exécution, le plan **« Equilibre de charge plus réduire le coût »** offre le coût le plus bas.

Le tableau suivant récapitule les résultats relatifs au coût total de l'exécution.

Nombre de tâches	100	200	300	400	500
Equilibre de charge	73500	147000	220500	294000	367500
Equilibre de charge+ satisfaction des tâches dans le délai	73500	147000	220500	294000	367500
Equilibre de charge + réduire le coût	60000	120000	180000	240000	300000
Equilibre de charge + satisfaction des tâches dans le délai + réduire le coût	64200	131250	197850	266850	333600
Equilibre de charge + division des tâches	73500	147000	220500	294000	367500

Tableau III.5 : Résultat de simulation du plan Equilibre de charge (Coût total par nombre de tâches)

La figure suivante montre l'impact du nombre de tâches sur le coût total des tâches.

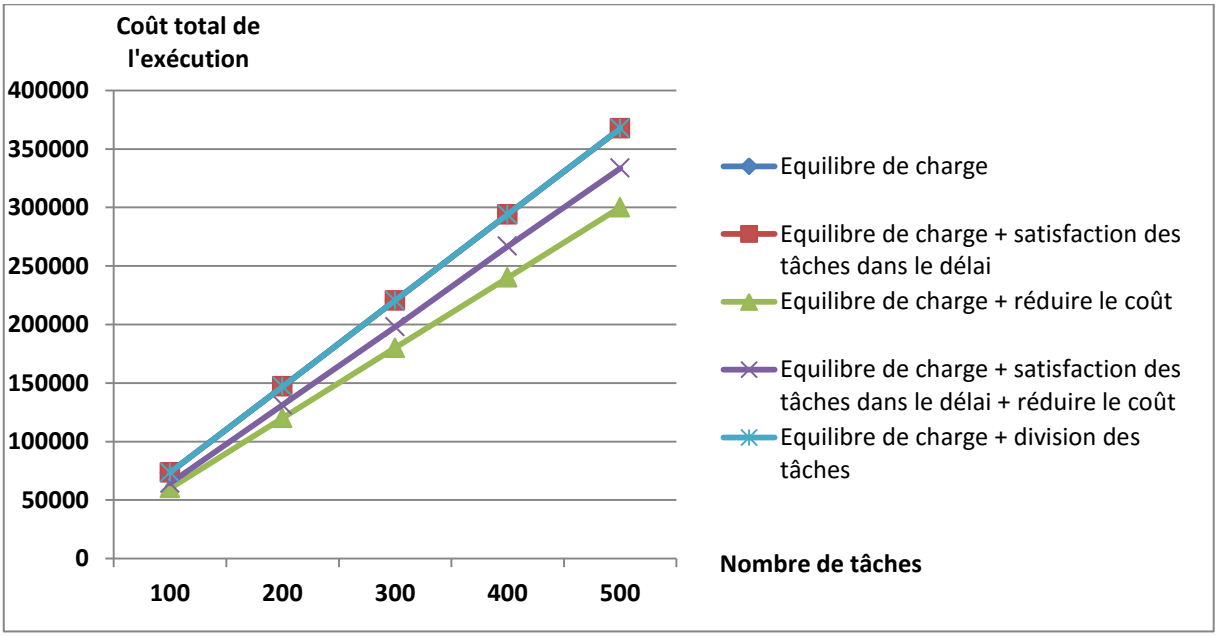

Figure III.18 : Courbe du coût total d'exécution par nombre de tâches

Le plan **« réduire le coût »** offre le meilleur coût, parce qu'il vise les ressources ayant le coût le plus bas.

Le tableau suivant récapitule les résultats relatifs au temps d'exécution.

Nombre de tâches	**100**	**200**	**300**	**400**	**500**
Equilibre de charge	115	215	317	418	518
Equilibre de charge+ Satisfaction des tâches dans le délai	114	214	314	415	515
Equilibre de charge +Réduire le coût	198	382	573	758	954
Equilibre de charge +Satisfaction des tâches dans le délai +Réduire le coût	142	244	344	445	545
Equilibre de charge +Division des tâches	104	205	304	404	505

Tableau III.6 : Résultat de simulation du plan Equilibre de charge (temps d'exécution par nombre de tâches)

La figure suivante montre l'impact du nombre de tâches sur le temps d'exécution.

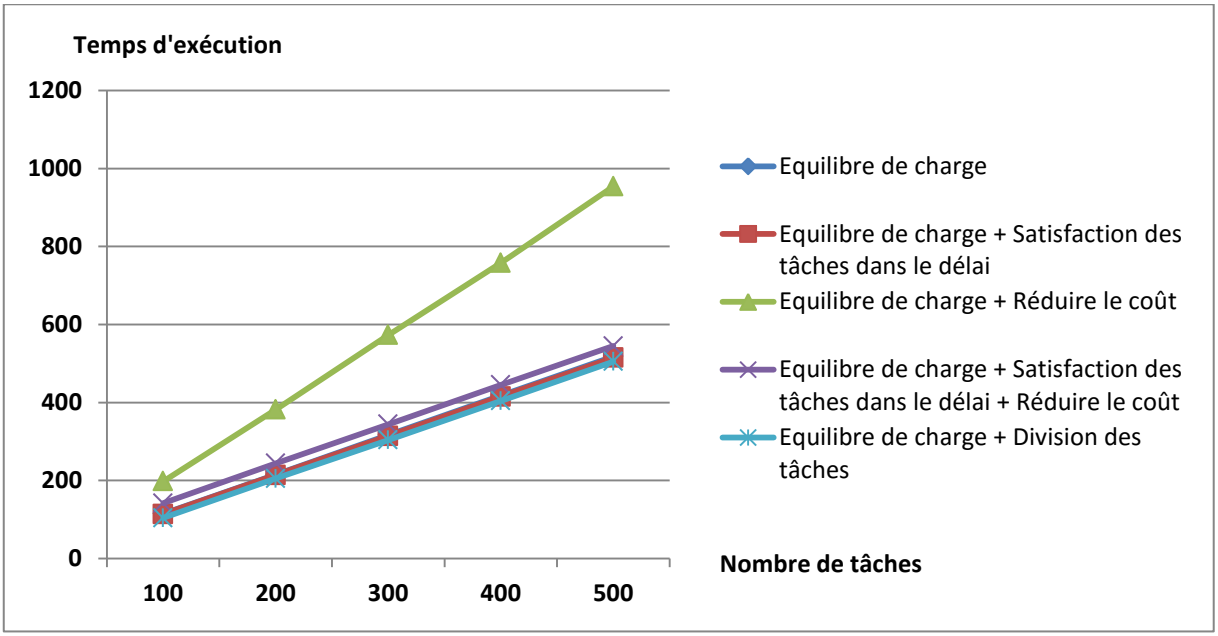

Figure III.19 : Courbe du temps d'exécution par nombre de tâches

Le plan de « **division des tâches** » offre le meilleur résultat puisqu'il divise les tâches en sous tâches et les exécute en parallèle.

La partie suivante représente les résultats obtenus de notre application en utilisant la coopération entre les agents BDI. Donc, nous avons créé quatre agents BDI avec cinq hosts de calcul pour chaque agent BDI.

Nous avons créé ainsi 100, 200, 300, 400 et 500 tâches respectivement avec dans tous les cas un temps d'exécution égal à 15 secondes et un délai égal à 25 secondes.

Nous avons simulé l'exécution de ses tâches avec et sans coopération entre les agents BDI.

Le tableau suivant récapitule les résultats du plan économie d'énergie avec et sans coopération.

Nombre de ressources actives	**100**	**200**	**300**	**400**	**500**
Economie d'énergie sans coopération	4	7	8	9	12
Economie d'énergie avec coopération	3	5	7	9	10

Tableau III.7 : Résultat de simulation du plan Economie d'énergie (temps d'exécution par nombre de tâches)

La figure suivante montre le nombre de ressources utilisées en fonction du nombre de tâches.

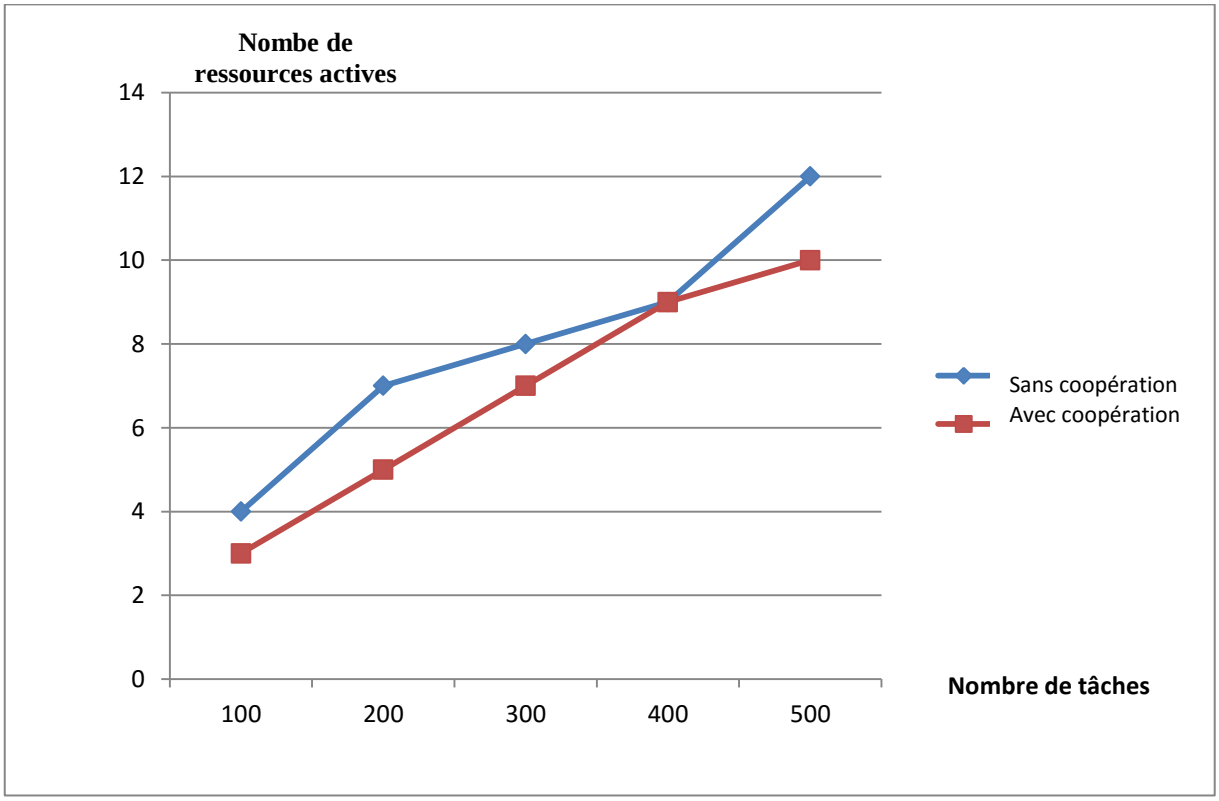

Figure III.20 : Courbe de ressources utilisées par nombre de tâches

Les résultats montrent que la coopération des agents BDI dans le plan économie d'énergie permet de réduire le nombre de ressources utilisées puisque la prise décision se base sur l'état de toutes les ressources.

Le tableau suivant récapitule les résultats du plan économie d'énergie (nombre de tâches exécutées dans le délai).

Nombre de tâches	100	200	300	400	500
Sans coopération	6	13	18	17	15
Avec Coopération	3	5	8	9	12

Tableau III.8 : Résultat de simulation du plan Economie d'énergie (des tâches exécutées dans le délai)

La figure suivante montre les tâches exécutées dans le délai en mode économie d'énergie avec et sans coopération.

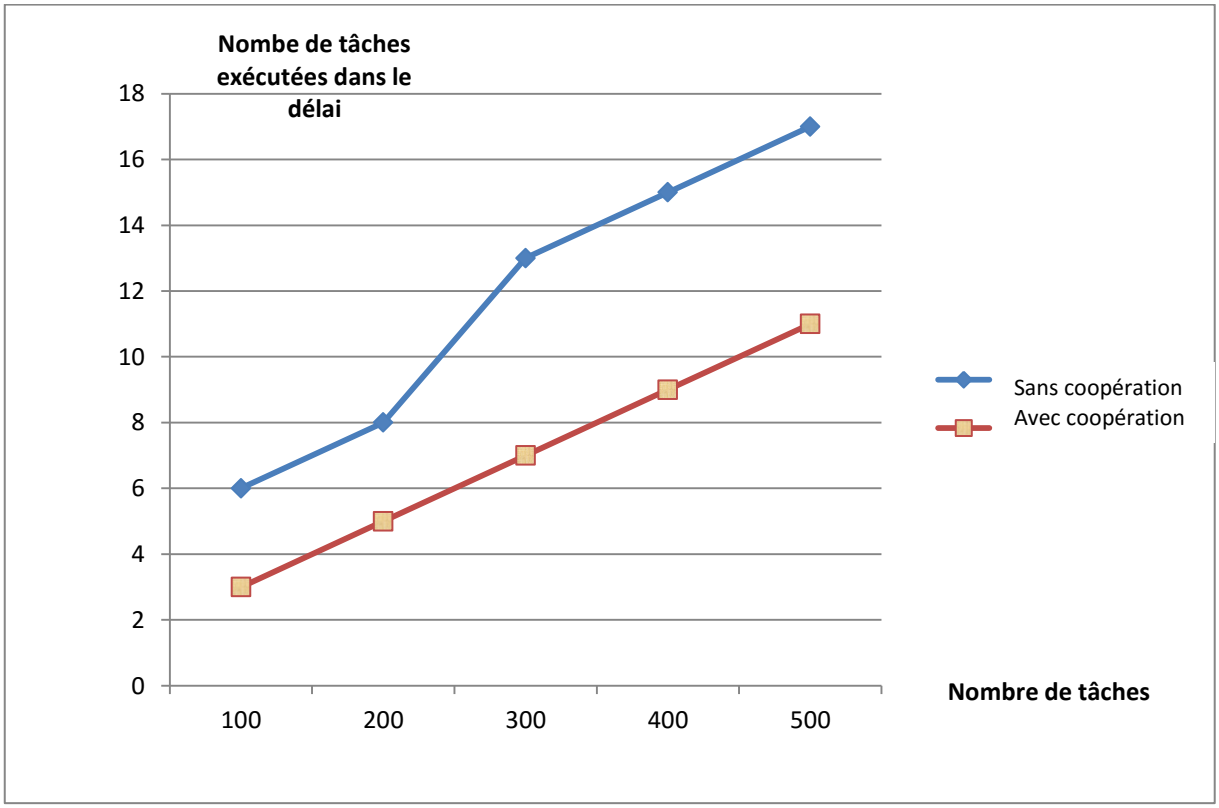

Figure III.21 : Courbe des tâches exécutées dans le délai, avec économie d'énergie

Les résultats montrent que la coopération des agents BDI réduit le temps d'exécution des tâches dans le délai en mode économie d'énergie.

Le tableau suivant récapitule les résultats du plan équilibre de charge avec et sans coopération.

Nombre de tâches	**100**	**200**	**300**	**400**	**500**
Sans coopération	100	179	273	382	473
Avec Coopération	100	200	300	400	500

Tableau III.10 : Résultat de simulation du plan Equilibre de charge (des tâches exécutées dans le délai)

La figure suivante montre l'impact du nombre de tâches sur celles exécutées dans le délai, en mode équilibre de charge avec et sans coopération.

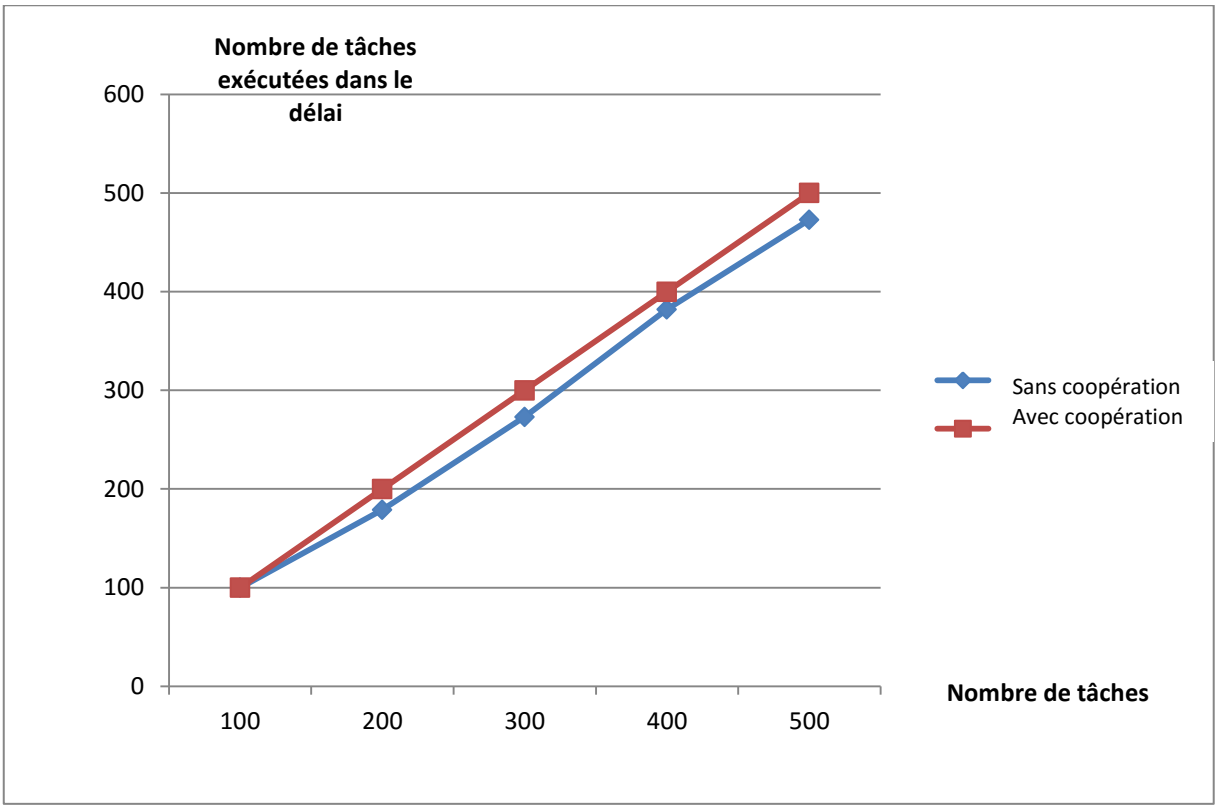

Figure III.22 : Courbe des tâches exécutées dans le délai, avec équilibre de charge.

Les résultats montrent que la coopération des agents BDI améliore la QoS du système parce que la coopération des agents BDI permet de mieux distribuer la charge sur la totalité des ressources.

III.7 Conclusion

Dans ce chapitre, nous avons présenté l'architecture globale de notre application où un ensemble d'agents coopèrent leurs efforts pour assurer une bonne prise de décision. Le fonctionnement de cette dernière se déroule en deux phases : « l'ordonnancement des tâches» et la « simulation de l'exécution des tâches ». En ce qui concerne l'ordonnancement, nous avons présenté les différents algorithmes de prise de décision. Pour valider les décisions prises dans la phase d'ordonnancement, nous avons analysé les résultats obtenus de nos simulations. Nous constatons que chaque algorithme fonctionne correctement pour satisfaire les besoins du plan choisit. Nous avons montré ainsi l'objectif de chaque plan et son influence sur la QoS, le coût total, le temps d'exécution ainsi que la satisfaction des tâches dans le délai. Nous avons réalisé également une comparaison entre les différentes stratégies afin de donner à l'utilisateur une idée sur la stratégie qui doit être utilisée dans certaines situations.

Conclusion générale

Tout au long de ce mémoire, nous avons présenté les différentes technologies nécessaires pour proposer une application basée sur les agents BDI dans le contexte du cloud computing.

En premier lieu, nous avons effectué une étude générale sur le cloud computing afin d'identifier ses différentes caractéristiques et ses besoins. Nous avons illustré l'intérêt d'utiliser les agents BDI pour des applications réelles afin de montrer l'efficacité de l'utilisation de cette technologie.

Ensuite, nous avons présenté notre application à base d'agent BDI en se focalisant sur les différents algorithmes proposés ainsi que leurs intérêts.

Finalement, et dans un souci de montrer l'efficacité de ces algorithmes, nous avons analysé les résultats de notre expérimentation.

Ce mémoire constitue une base de travail à partir de laquelle, de nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté. Les perspectives que nous proposons peuvent donc s'orienter vers la réalisation des points suivants :

- ✓ Hosts multitâches (exécuter plusieurs tâche simultanément).
- ✓ Hosts hybride (host gère le stockage et de calcul).
- ✓ La gestion des dépendances fonctionnelles entre les tâches.

Référence bibliographique:

[1] Rajan, Sameer, and Apurva Jairath. "Cloud computing: The fifth generation of computing."Communication Systems and Network Technologies (CSNT), 2011 International Conférence on. IEEE, 2011.

[2] Jean-Paul Figer, « L'informatique en nuage » [Cloud Computing], ARMOSC, 2012.

[3] hoyar, Rahul, and Nitin Chop de. "Cloud Computing: Service models, Types, Database and ssues.".2013.

[4] Marc Jacob, « DATACENTERS ... Une chance pour la France », Livre blanc, 2010.

[5] Willy MUNSCH and Jean-Louis CAIRE, « Objectif Cloud : une démarche pratique orientée services », Livre, ENI-Edition, 2014.

[6] MIMOUNE MOUSSA**, «** Etude sur la sécurité du Cloud Computing **»,** Rapport de Master en Technologie de l'Information et de la Communication (TIC), université de m'sila, 2015.

[7] Jacques, Ferber. "Les Systèmes Multi-agents, Vers une intelligence collective." *Inter Editions, Paris*. 1995.

[8] Hewitt, Carl, and Henry Baker Jr. "Actors and continuous functionals". No. MIT-LCS/TR-194.MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1977.

[9] Ahmed TLILI, "Etude et Réalisation d'une Plate Forme Multi-Agents", Thèse de Magistère en Génie Logiciel et Intelligence Artificielle, Université Hadj Lakhdar de Batna. 2012.

[10] Wooldridge, Michael, and Nicholas R. Jennings. "Intelligent agents: Theory and practice." Knowledge engineering review 10.2 (1995): 115-152.

[11] A. Drogoul, J. Meyer, "Intelligence artificielle située". Hermès Paris Science publications, 1999.

[12] Fikes, Richard E., and Nils J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." Artificial intelligence 2.3-4 (1971): 189-208.

[13] Liu, Yingjiu. "De la nécessité et de la façon de coopérer, de s'auto-organiser et de se reconfigurer dans des systèmes de production complexes: modélisation et gestion d'un système virtuel par une approche multi-agents". Diss. Grenoble, INPG, 2002.

[14] Brooks, Rodney A. "Intelligence without representation." Artificial intelligence 47.1 (1991): 139-159.

[15] Reaidy, Jihad. "Etude et mise en œuvre d'une architecture d'agents en réseau dans les systèmes dynamiques situés: pilotage des systèmes de production complexes". Diss. Chambéry, 2003.

[16] Daknou, Amani. "Architecture distribuée à base d'agents pour optimiser la prise en charge des patients dans les services d'urgence en milieu hospitalier". Diss. Ecole Centrale de Lille, 2011.

[17] Chaouche, Ahmed Chawki. "Une approche multi-agent pour la conception de systèmes d'intelligence ambiante: Un modèle formel intégrant planification et apprentissage". Diss. Université Pierre et Marie Curie; Université Constantine 2-Abdelhamid Mehri, 2015.

[18] Shoham, Yoav. "Agent-oriented programming."Artificial intelligence 60.1 (1993): 51-92.

[19] Le Bars, Marjorie. "Un Simulateur Multi-Agent pour l'Aide à la Décision d'un Collectif: Application à la Gestion d'une ressource Limitée Agro-environnementale". Diss. Université Paris Dauphine-Paris IX, 2003.

[20] Chaib-Draa, Brahim, Imed Jarras, and Bernard Moulin. "Systèmes multi-agents: principes généraux et applications." Edition Hermès (2001): 1030-1044.

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

[22] http://turing.cs.pub.ro/auf2/html/chapters/chapter2/chapter_2_2_2.html, Consulté (Mars 2016).

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviations	Signification
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SMA	Systèmes multi-agents
BDI	Beliefs, Desires ,Intentions
NIST	National Institute of Standards and Technology
TaaS	Information Technology as a Service
API	Application Programming Interface
FTP	File transfer protocol
SMTP	Simple mail transfer protocol
CSA	Cloud Security Alliance
IA	Intelligence artificielle
DVMT	Distributed Vehicule Monitoring Test
KQML	Knowledge Query Manipulation Language
ACL	Agent Communication Language
FIPA	Foundation for Intelligent Physical Agents
JADE	Java Agent development framework.
QoS	Quality of service
JVM	Java Virtual Machine.
CDDL	Common Development and Distribution License
CSELT	Centro Studi Elaboratori Telecomunicazion
GUI	Graphical user interface

Résumé :

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

Dans ce mémoire, nous avons développé des algorithmiques répondants au problème d'ordonnancement de tâches. Ils sont basés sur le paradigme d'agents BDI en mettant en œuvre un modèle de collaboration pour l'ordonnancement dynamique de tâches.

Mots clés : Ordonnancement des tâches, Système multi-agents, BDI, Cloud computing.

Abstract :

Task scheduling in cloud computing environment is subject to specific constraints. The distributed nature of cloud computing does not allow a centralized approach. The distributed approach is confronted to the limitations of the visibility of the overall state of the dynamic system.

In this report, we developed algorithms to solve the problem of task scheduling. The algorithms are based on BDI agent's paradigm by implementing a collaborative model for dynamic scheduling tasks.

Key words : Task scheduling, Multi agent System, BDI, Cloud computing.

ملخص :

جدولة المهام في بيئة الحوسبة السحابية تخضع لقيود خاصة. طبيعة توزيع الحوسبة السحابية لا تتوافق مع أي رؤية مركزية. ويواجه النهج توزيعها مع القيود المفروضة على تسليط الضوء على الحالة العامة للنظام ديناميكي.

في هذه المذكرة، وضعنا منهجية و لغوراتيمية لتلبية جدولة المهام. ويستند هذا النهج على نموذج وكلاء BDI (المعتقدات، الرغبات، النوايا)، و تنفيذ نموذج التعاون من أجل الجدولة الحيوية للمهام.

كلمات مفتاحية : جدولة المهام، نظام متعدد الوكلاء، وكلاء BDI، الحوسبة السحابية.