République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université ABBAS LAGHROUR Khenchela

Faculté des Sciences et Technologies

Département mathématique et Informatique

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

De master informatique

Option

Sécurité et Technologie Web

Thème

Méthode hybride pour la prédiction des pannes dans le Cloud computing

Encadré par :

- **Dr.** HIOUAL Ouidad

Réalisé par :

- BOUZIANE Ines
- GOUDJIL Rima

Année Universitaire 2019/2020

Remerciements

En premier lieu nous voulons remercier le bon DIEU qui nous a donnés la volonté, la patience, le courage et la force pour avoir pu achever ce travail.

Nous tenons à remercier, bien sûr, en priorité, notre encadreur, ***Dr Ouided Hioual.*** *Pour nous avoir donner l'opportunité de réaliser ce sujet sous sa direction, la confiance et l'aide qu'elle nous a accordé ainsi que sa disponibilité et ses conseils précieux et son temps consacré tout au long du travail*

Nous voulons remercier également tous les membres du jury qui Nous fait

l'honneur d'apprécier et d'évoluer.

Nous désirons remercier nos chers parents qui nous ont soutenus et

encouragé

durant toute notre vie et pendant notre cursus d'étude.

Enfin nous adressons nos meilleurs et chaleureux remerciements à tous nos professeurs et personnes qui nous avons aidée de près ou de loin dans la réalisation de ce travail.

Dédicaces

♥ *Je dédie ce mémoire à ma chère famille :*

À mes très chers parents et grands parents qui m'ont donné un magnifique modèle de labeur et de persévérance ♥

♥ *L'espoir de ma vie et la lumière de minuits: ma mère HIND* ♥

♥ *Ma source de Vie d'amour d'affection de joie de bonheur d'espoir de motivation de courage et de générosité B.Imed* ♥

♥ *À mon cher frère Amine et ma chère copine et sœur Souha Chorfi* ♥

♥ *A mes chères tantes Nadia Soumia Khalida Nadjeh Razika Sana* ♥

♥ *À mon cher oncle Adel*♥

♥ *À la personne qui m'a soutenu pendant tout le parcours de mes études Faiza* ♥

♥ *A la personne que j'aime beaucoup Salah Menani* ♥

♥ *A toute ma famille grand et petit* ♥

♥ ***Bouziane Ines*** ♥

Dédicaces

♥Je tiens à dédier ce modeste travail :

♥Avant tout à ma très chère mère, ma fierté et mon bonheur ;♥

♥A mon très cher père, l'homme qui ma donné le désir d'apprendre et le savoir-vivre ;♥

♥A mes très chers sœurs ; Linda, Massouda, Widad, Fadila, Aya, Ibtehal;♥

♥A toute ma famille surtout ma grande mère, mes cousines et cousins ;♥

♥A tous mes chères amies ; ♥

♥A tous mes collègues qui m'ont aidé à accomplir ce travail ;♥

♥A toutes mes collègues de mon groupe STW, et mes professeurs.♥

♥Goudjil rima ♥

Résumé

Le Cloud Computing fait référence à l'utilisation des capacités de calcul des ordinateurs distants, où l'utilisateur dispose d'une puissance informatique considérable sans avoir à posséder des unités puissantes. La probabilité de voir une faute intervenir durant l'exécution devient forte lorsque le nombre de nœuds augmente, puisqu'il est impossible d'empêcher totalement les fautes, une solution consiste à mettre en place des mécanismes de tolérance aux fautes. La prédiction des fautes est devenue une tâche majeure pour les ingénieurs informatiques et les développeurs de logiciels, car l'apparition de faute augmente le coût de l'utilisation des ressources. Dans ce travail nous avons proposé une méthode hybride de prédiction des fautes dans le Cloud computing basé sur les systèmes multi-agents et la technique de réplication, dans le but est de détecter les fautes et minimiser le temps perdu, cette méthode permet de garantir la continuité des services du Cloud Computing d'une façon transparent et efficace en présence des fautes.

Mots clés: prédiction des fautes; Cloud computing; système multi-agents ; Datacenter; réplication.

Abstract

Cloud Computing refers to the use of the computing capabilities of remote (distant) computers, where the user has considerable computing power without having to own powerful units. The probability of a failure occurring during execution becomes high as the number of nodes increases; since it is impossible to completely prevent failures; one solution is to implement fault tolerance mechanisms. Predicting failures has become a major task for IT engineers and software developers, as the appearance of failure increases the cost of resources usage. In this work we proposed a hybrid method to predict failures on Cloud Computing based on the multi-agent system and a replication technique, in the purpose of detecting failures and minimizing the wasted time, this method ensures the continuity of cloud computing services in a transparent and efficient way in the presence such of failures.

Keywords: failures Prediction (breakdowns); Cloud computing; Multi-agent system; Datacenter; Replication.

Table de Matières

CHAPITRE 03 : LA TOLERANCE AUX FAUTES DANS LE CLOUD

CHAPITRE 04 : DESCRIPTION ET CONCEPTION DE LA METHODE PROPOSEE

CHAPITRE 05 : IMPLEMENTATION

Liste des figures

Chapitre I : Les Systèmes multi-agents

Chapitre II : Cloud Computing

Chapitre III : La prédiction des fautes dans le cloud

Chapitre IV : Description et conception de la méthode proposée

Chapitre V : Implémentation

Introduction Générale

1. Contexte du travail

Avec le développement de l'Internet, l'informatique s'est basée essentiellement sur les communications entre serveurs, postes utilisateurs, réseaux et data centers. Au début des années 2000, les deux tendances à savoir la mise à disposition d'applications et la virtualisation de l'infrastructure ont vu le jour. La convergence de ces deux tendances a donné naissance à un concept fédérateur qu'est le Cloud Computing (informatique en nuage).

Le Cloud Computing peut être défini comme un modèle informatique distribué et spécialisé, configurable de manière dynamique, mutualisée, évolutive (Puissance de calcul élevée, espace de stockage important, etc.) et fourni à la demande via des réseaux de communication. Ce nouveau paradigme offre plusieurs avantages comme le déploiement rapide, le paiement à l'usage, la réduction de coûts, l'évolutivité facile, la délivrance plus rapide de service, l'accès au réseau omniprésent, etc. En raison de ces diverses caractéristiques, le Cloud Computing devient une solution intéressante pour les entreprises et les chercheurs.

L'approche du Cloud computing s'appuie principalement sur le concept de virtualisation. Ce concept est un ensemble de techniques permettant de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, isolés les uns des autres. Un cloud est constitué d'un ensemble de machines virtuelles qui utilisent la même infrastructure physique.

Les techniques de virtualisation permettent notamment la migration de machines virtuelles d'un nœud physique à un autre. En fait, les objectifs recherchés à travers la migration de machines virtuelles sont multiples : la répartition de charge, l'accès aux ressources, la tolérance aux fautes ou encore pour accomplir certaines activités de maintenance.

Les systèmes multi agents représentent également un paradigme de calcul et de gestion distribué sous forme d'un ensemble d'agents qui interagissent dans un environnement pour résoudre un problème commun en utilisant les ressources et les connaissances de chaque agent.

Malgré le fait que le cloud computing et les systèmes multi agents soient deux modèles de calcul distribués, il existe plusieurs différences mais aussi une grande synergie entre les deux modèles qui peut être bénéfique dans les deux sens. D'une part, les efforts de recherche menés dans le domaine du cloud tendent vers des solutions pratiques pour affranchir les contraintes de l'outil informatique traditionnel à savoir les problèmes de l'espace de stockage, la

portabilité des données, l'élasticité et l'agilité. Le but donc est d'avoir une haute disponibilité du service à des coûts réduits. D'autre part, l'utilisation des systèmes multi agents dans la recherche se distingue généralement dans trois types d'utilisations : la simulation de phénomènes complexes, la résolution de problèmes, et la conception de programmes tout en mettant l'accent sur l'aspect d'intelligence individuelle et collective. Ceci dit, dans les deux sens les deux paradigmes restent complémentaires. Le cloud offre une infrastructure puissante et distribuée nécessaire pour le développement et l'exécution d'applications agents. Tandis que le cloud peut bénéficier de l'intégration des caractéristiques des systèmes multi agents principalement leur intelligence pour surmonter le problème de sécurité et confidentialité des données.

2. Problématique et objectifs

À l'instar des autres services informatiques, les services de Cloud ne sont pas à L'abri d'une défaillance. En plus dans des tels systèmes le taux de fautes croit avec la taille du système lui-même. Dans un tel contexte, la sûreté de fonctionnement des applications est un élément de première importance, c'est pourquoi un mécanisme de tolérance aux fautes devient nécessaire pour en assurer certains aspects.

Un système tolérant aux fautes peut être capable de tolérer un ou plusieurs types de fautes, y compris les défauts matériels transitoires, intermittents ou permanents, les erreurs de logiciel et de conception, les erreurs de l'opérateur ou les perturbations induites ou les dommages physiques.

La tolérance aux fautes dans le Cloud computing est un gros problème qui a été très largement étudié depuis une trentaine d'années. La littérature propose aujourd'hui un grand nombre de protocoles de tolérance aux fautes qui sont par : réplication, recouvrement arrière, que l'on peut diviser en deux catégories : les protocoles par points de reprise et les protocoles par journalisation. Chacune de ces catégories présente des propriétés différentes en termes de performance, et n'est parfois pas applicable selon le système ou selon l'application considérée.

Il ya aussi plusieurs approches ont été proposées pour résoudre le problème de tolérance aux fautes dans le Cloud computing, y compris l'approche système multi agents.

Pour résoudre le problème de prédiction des fautes au niveau de Cloud computing, nous proposons une méthode hybride en utilisant la technique de réplication basée sur le paradigme de l'agent, qui s'est révélé efficace pour les applications distribuées.

3. Organisation de la thèse

Cette thèse comporte cinq chapitres. L'introduction générale présente le contexte et la problématique de la recherche.

- Dans le premier chapitre, nous allons tenter de cerner le domaine des systèmes multi-agents. Nous l'abordons d'abord sous l'aspect « agent » en présentant les caractéristiques relatives à cette unité de base du système multi-agents. Puis, nous passons à la dimension collective du système.
- Le deuxième chapitre présente un état de l'art sur le Cloud Computing. On donne une introduction sur le Cloud Computing, y compris la définition, ses caractéristiques, les types des services Cloud, les modèles de déploiement ainsi que la sécurité, les avantages et les inconvénients du cloud et nous terminerons par une conclusion.
- Le troisième chapitre décrit les différents concepts et les grandes lignes liés à la tolérance aux fautes ainsi que les diverses techniques de gestion des fautes utilisées.
- Le quatrième décrit en détail la méthode proposée et la conception.
- Le dernier chapitre décrit l'implémentation.

Finalement, la conclusion générale fait une synthèse de notre contribution globale et met en évidence les perspectives de cette recherche.

Chapitre I :

Les Systèmes multi-agents

1. Introduction

La notion d'agent et de système multi-agents (SMA) est relativement récente en informatique, mais elle tend à prendre de plus en plus d'importance. Ce domaine est né au début des années 80, de l'idée de distribuer les connaissances et le contrôle dans les systèmes d'Intelligence Artificielle. Il offre aujourd'hui une alternative intéressante pour la conception, la mise en œuvre ou la simulation et la compréhension de systèmes coopératifs, distribués et ouverts [Ferber, 2006], [Sycara ,1998], [Nguyen et al, 2007].

Les systèmes multi-agents (SMA) sont des systèmes qui se basent sur le partage des connaissances et des compétences sur plusieurs agents les quels doivent s'organiser et coordonner leurs actions afin de réaliser l'objectif global du système. Ils sont fondés sur des modèles d'interaction complexes qui se traduisent par des stratégies de résolution comme la coopération, la coordination, la négociation. Ces systèmes sont de plus en plus utilisés dans la résolution de problèmes complexes. Leur principe de modularité permet de diminuer la complexité de conception du système global et d'améliorer sa maintenance.

Dans ce chapitre, nous allons tenter de cerner le domaine des systèmes multi-agents. Nous l'abordons d'abord sous l'aspect « agent » en présentant les caractéristiques relatives à cette unité de base du système multi-agents. Puis, nous passons à la dimension collective du système.

2. Entité isolée : L'Agent

2.1 Définition

Il existe plusieurs définitions d'un agent, Il est donc nécessaire, pour avoir une bonne vision de ce concept, de présenter plusieurs d'entre elles.

Selon Ferber [Ferber, 1995], il le définit comme étant « une entité physique ou virtuelle évoluant dans un environnement dont il n'a qu'une représentation partielle et sur lequel il peut agir. Un agent est capable de communiquer avec d'autres agents et est doté d'un comportement autonome. Un agent possède généralement des accointances qui sont l'ensemble des agents avec lesquels ils communiquent ou bien interagit ».

Selon Yves Demazeau [Demazeau, 1996], « un agent est une entité réelle ou virtuelle dont le comportement est autonome, évoluant dans un environnement qu'il est capable de percevoir et sur lequel il est capable d'agir et d'interagir avec les autres agents ».

Selon Russel et al [Russell et al., 2006], : « Un agent est tout ce qui peut être compris comme percevant son environnement à travers des senseurs et comme agissant sur cet environnement par l'intermédiaire d'effecteurs ». (Voir figure 1.1).

On remarque que ces définitions abordent une notion essentielle : l'autonomie. En effet, ce concept est au centre de la problématique des agents. L'autonomie est la faculté d'avoir ou non le contrôle de son comportement sans l'intervention d'autres agents ou d'êtres humains.

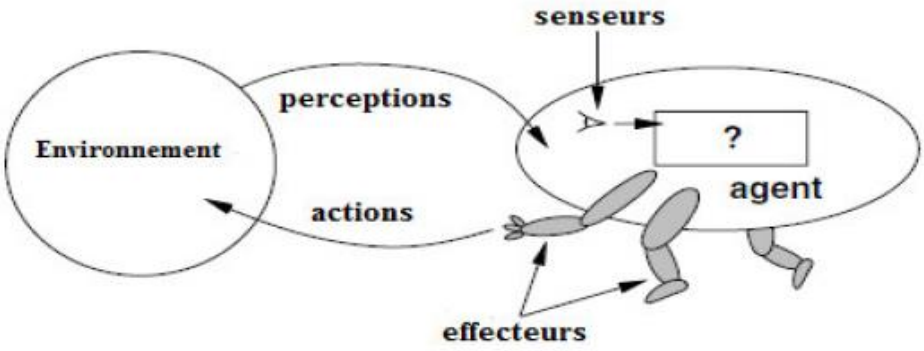

Figure 1.1. Agent /Environnement [Russell et al. 2006]

2.2 Propriétés d'agent

Il existe dans la littérature plusieurs propriétés d'agents peuvent être identifiées :

- **Autonomie :** les agents contrôlent leurs actions et leurs états internes. Le système dans son ensemble est capable de réagir sans l'intervention d'un humain ou d'un autre agent.
- **Réactivité :** ils perçoivent leur environnement et réagissent aux changements qui s'y produisent dans le temps requis.
- **Initiative :** le comportement des agents est déterminé par les buts qu'ils poursuivent et par conséquent ils peuvent produire des actions qui ne sont pas seulement des réponses à leur environnement.
- **Habilité sociale :** pour satisfaire ses buts un agent peut demander l'aide d'autres agents avec lesquels il partage la réalisation de tâches.
- **Raisonnement :** un agent peut décider quel but poursuivre ou à quel événement réagir, comment agir pour accomplir un but, ou suspendre ou abandonner un but pour se dédier à un autre.
- **Apprentissage :** l'agent peut s'accommoder progressivement à des changements dans des environnements dynamiques grâce à des techniques d'apprentissage.

- **Mobilité** : dans des applications déterminées il peut être intéressant de permettre aux agents de migrer d'un nœud à un autre dans un réseau tout en préservant leur état lors de sauts entre nœuds [J. I. et al, 2002][N. R. Jennings,2000][o. Khayati,2005].

2.3 Architectures d'agent

Un agent évolue dans un environnement, il doit être en mesure de recevoir des informations de cet environnement par des récepteurs, et d'agir sur ce même environnement par des effecteurs, suivant un comportement décidé selon le raisonnement de l'agent. Dans cette section en va présenter les différents types d'agents :

2.3.1 Les agents réactifs

Les agents réactifs souvent qualifiés de peu intelligents, les agents réactifs possèdent une représentation très simplifiée de leur environnement [Ferber ,1995]. Leurs capacités consistent à réagir uniquement en mode stimulus/ action vu comme une forme de communication (ou perception) (voir figure 1.2).

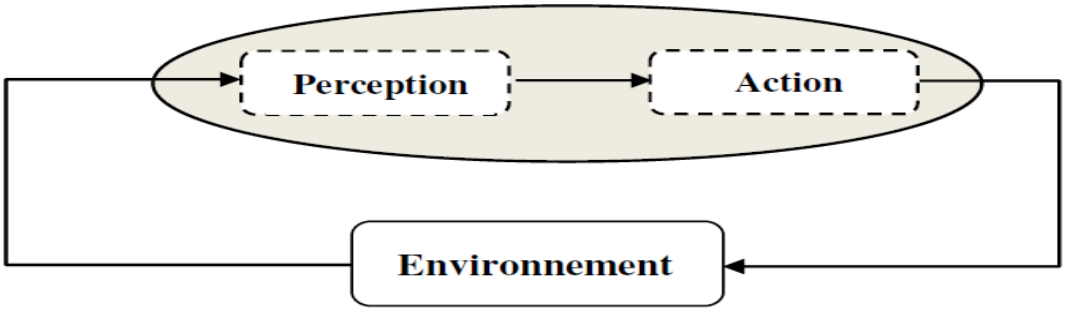

Figure 1.2. Agent réactif

2.3.2 Les agents cognitifs

Les agents cognitifs possèdent une représentation (partielle mais sophistiquée) de leur environnement, ont des buts explicites et sont capables de planifier leur comportement, de mémoriser leurs actions passées, de communiquer par envoi de messages ou via des langages d'interaction élaborés, de négocier, etc. (Voir figure 1.3)

L'une des architectures cognitives les plus connues est sans doute l'architecture BDI :

Belief (Croyance), Desire (Désir), Intention (Intention) qui comme son nom l'indique, est basée sur les notions d'attitudes mentales que sont la croyance, le désir et l'intention :

- Les croyances : ce que l'agent connaît de son environnement.
- Les désirs : correspondent aux états possibles de l'environnement que l'agent souhaiterait voir réalisés.

- Les intentions correspondent aux projets de l'agent pour satisfaire ses désirs. [Ferber ,1995].

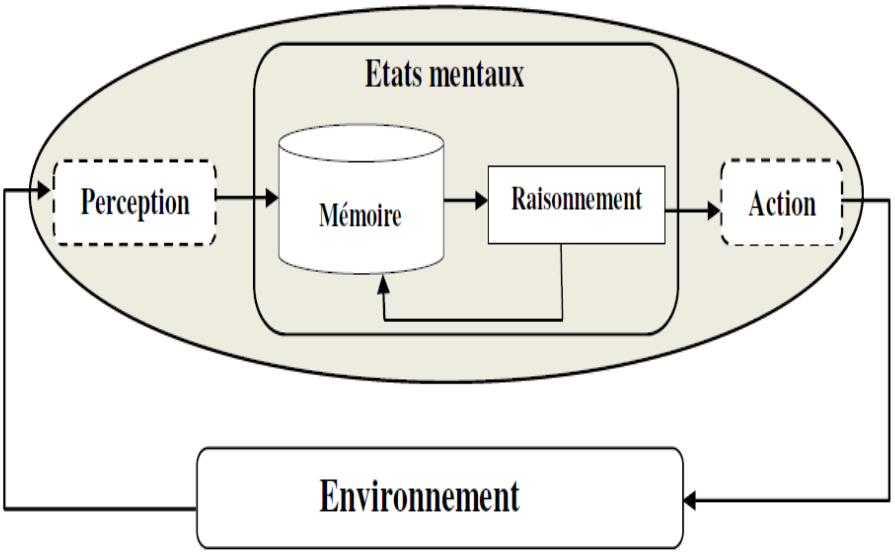

Figure 1.3. Agent cognitif [Ferber ,1995]

2.3.3 Agents hybrides (mixtes)

Les agents hybrides intègrent l'aspect cognitif et réactif. L'idée est de combiner les deux types d'approches qui peuvent être vues comme complémentaires. Dans une telle architecture un agent est composé de modules qui gèrent indépendamment la partie réactive et la partie cognitive. Le problème central reste de trouver le mécanisme idéal assurant cette combinaison.

2.4 Comportement d'agent

Un agent se caractérise essentiellement par la manière dont il est conçu et par ses actions. En d'autres termes par son architecture et son comportement [Ferber, 1995]. Ce dernier terme est fondamental dans la définition et la modélisation d'un agent.

Le comportement caractérise l'ensemble des propriétés que l'agent manifeste dans son environnement. On peut le comprendre par les réponses de l'agent aux sollicitations de son environnement ou en regardant sa manière d'évoluer. Il est analysable sans connaître les détails d'implémentation. Il s'agit d'un phénomène qui peut être appréhendé par un observateur extérieur qui au regard des actions qu'entreprend l'agent, peut induire ou spécifier ce qu'une architecture est censée produire.

3. Les systèmes multi-agents (SMA)

Un système multi-agents peut être considéré comme un ensemble d'agents partageant un environnement commun et capable d'interagir entre eux et sur l'environnement. Néanmoins, et pareil au concept « agent », il n'existe pas de définition acceptée en unanimité dans la littérature.

3.1 Définitions

Plusieurs définitions ont été proposées, nous allons présenter quelque définition :

Selon Anne Nicole [Anne Nicole, 99], « Un Système Multi-Agents (SMA) comporte plusieurs agents qui interagissent entre eux dans un environnement commun. Certains de ses agents peuvent être des personnes ou leurs représentants (avatars), ou même des machines mécaniques.

S'il y a moins de trois agents, on parle plutôt d'interaction homme/machine, ou machine /machine que de systèmes multi-agents ».

Selon Wooldridge et Jennings [Wooldridge et al., 2000], « Un SMA est un ensemble d'agents ayant des buts ou des tâches, et qui interagissent pour les accomplir ».

Selon Ferber [Ferber, 1995], un système multi-agents est un système composé des éléments suivants (voir figure 1.4) :

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

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

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

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

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

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

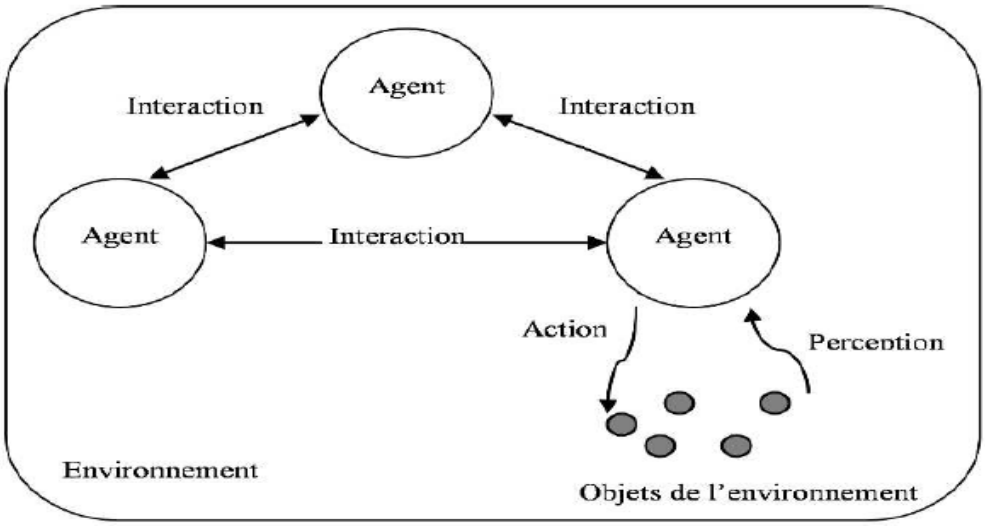

Figure.1.4. Représentation d'un système multi-agent selon Ferber [Ferber,95]

3.2 Les caractéristiques d'un système multi-agents

Un SMA est généralement caractérisé par :

1) Chaque agent possède des informations et/ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel ;

2) Il n'y a pas de contrôle global du système multi-agents. Mais, dans le cas du contrôle centralisé, il peut exister une entité qui a une vue globale de l'activité du système et qui a la charge de contrôler tout le système [N. R. Jennings et al, 1998][K. Sycara, 1998] ;

3) Les données et les connaissances sont décentralisées.

3.3 Typologie des systèmes multi-agents

3.3.1 Système multi-agents ouvert/fermé

Un système multi-agents ouvert [Vercouter.l, 2000] partage les caractéristiques des systèmes ouverts. Les entités élémentaires du système, i.e. les agents, n'ont pas la possibilité d'avoir une représentation complète de l'environnement, tandis que le système dans sa globalité doit être modulaire et extensible. La modularité concerne le fait que le système multi-agents est composé de plusieurs sous-systèmes mises en relation. Ces sous-systèmes ont chacun leur propre mode de fonctionnement. L'extensibilité se traduit par le fait que le système multi-agents supporte l'ajout et le retrait dynamique d'éléments. A l'inverse, le qualificatif fermé signifie que l'ensemble des agents qui compose le système reste le même. Certaines applications comme le e-commerce nécessitent des systèmes multi-agents ouverts où les agents y entrent et en sortent librement.

3.3.2 Système multi-agents homogène/hétérogène

Un système multi-agents homogène est composé d'agents homogènes. Deux agents ont cette particularité s'ils sont identiques du point de vue de leurs modèles et de leurs architectures. Le qualificatif hétérogène est utilisé pour préciser que le système multi-agents est composé d'agents différents du point de vue de leurs modèles et de leurs architectures.

4. Domaines d'application des SMA

Les applications des systèmes multi-agents couvrent de nombreux domaines. Citons les systèmes d'information coopératifs, la simulation sociologique, les outils documentaires adaptés au Web, les robots autonomes coopératifs, jeux vidéo (multi joueurs), résolution distribuée de problèmes, etc. Néanmoins les systèmes multi-agents développés actuellement peuvent être classés en trois catégories [Gleizes et al, 2008], [Ferber, 2005] :

- Les simulations dont l'objectif est la modélisation de phénomènes du monde réel, afin d'observer, de comprendre et d'expliquer leur comportement et leur évolution. Les systèmes multi-agents ont trouvé rapidement un champ extrêmement propice à leur développement dans le domaine de la modélisation de systèmes complexes ne trouvant pas de formalisation mathématique adaptée. Dans le domaine des sciences du vivant d'abord, ensuite dans celui des sciences humaines et sociales, les SMA ont montré qu'il était possible de modéliser au niveau micro les comportements d'entités élémentaires et d'étudier au niveau macro le résultat global de l'interaction de ces entités. En effet, la simulation multi-agents permet de tester rapidement le changement de certaines hypothèses ; elle permet aussi d'intégrer de nouveaux agents et d'éditer, sur un plan pratique, les résultats pour comparer les expérimentations les unes aux autres ; de plus elle permet de préserver l'hétérogénéité du système à simuler.
- Les applications dans lesquelles les agents jouent le rôle d'êtres humains. La notion d'agent simplifie la conception de ces systèmes et amène de nouvelles problématiques centrées utilisateur telles que la communication, la sécurité… Les systèmes de ventes aux enchères dans laquelle les agents jouent les rôles de commissaire priseur et d'acheteurs, représentent une classe d'applications de cette catégorie.
- La résolution de problèmes : telle qu'elle avait été définie en Intelligence Artificielle, étendue à un contexte distribué. Dans ce cadre, l'objectif est de mettre en œuvre un ensemble de techniques pour que des agents, pertinents pour la résolution d'une partie

ou l'ensemble du problème, participent de manière efficace et cohérente à la résolution du problème global.

5. Conclusion

L'avènement de l'Intelligence Artificielle et la conception des SMA ont beaucoup impacté 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. L'amélioration se traduit principalement en matière 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.

Le cloud computing et les systèmes multi-agents représentent une nouvelle approche prometteuse pour l'exploration de données distribuées, et il existe quatre avantages de l'utilisation des agents pour l'exploration de données distribuées. A savoir, garder l'autonomie des sources de données, avoir une grande sociabilité vers des données distribuées massives, la stimulation de l'exploration de données distribuées multi-stratégie est l'activation de l'exploration de données collaboratives. Dans le chapitre suivant, en va présenter le cloud computing.

Chapitre II :
Cloud Computing

1. Introduction

De nos jours, le domaine des technologies de l'information et de la communication est devenu l'un des piliers de la société moderne. Le Cloud Computing est devenu un nouveau paradigme sur utilisé pour l'hébergement et la fourniture des ressources via Internet. Il représente la cinquième génération de l'informatique après les mainframes, les ordinateurs personnels, le paradigme client/serveur et le web (World Wide Web).

Dans ce chapitre, on donne une introduction sur le Cloud Computing, y compris la définition, ses caractéristiques, les types des services Cloud, les modèles de déploiement ainsi que la sécurité, les avantages et les inconvénients du cloud et nous terminerons par une conclusion.

2. Le Cloud Computing (l'informatique en nuage)

2.1 Définition

Dans la littérature, il existe plusieurs définitions du cloud computing :Selon le "National Institute of Standards and Technology (NIST) ", [P. Mell et T. Grance,2018], "Le Cloud Computing est un modèle qui permet d'accéder rapidement à un pool de ressources informatiques mutualisées, à la demande (serveurs, stockage, applications, bande passante, etc.), sans forte interaction avec le fournisseur de service " (voir figure 2.1).

Selon Gartner [Gartner IT glossary,2013], "le Cloud Computing est un style d'informatique dans lequel des moyens informatiques disponibles, évolutives et élastiques sont fournies sous forme de service aux utilisateurs externes utilisant les technologies Internet. L'opération de fourniture est basée sur cinq attributs : la multi-location (ressources partagées), une extensibilité massive, l'élasticité, le paiement à l'utilisation et l'auto-provisionnement des ressources "

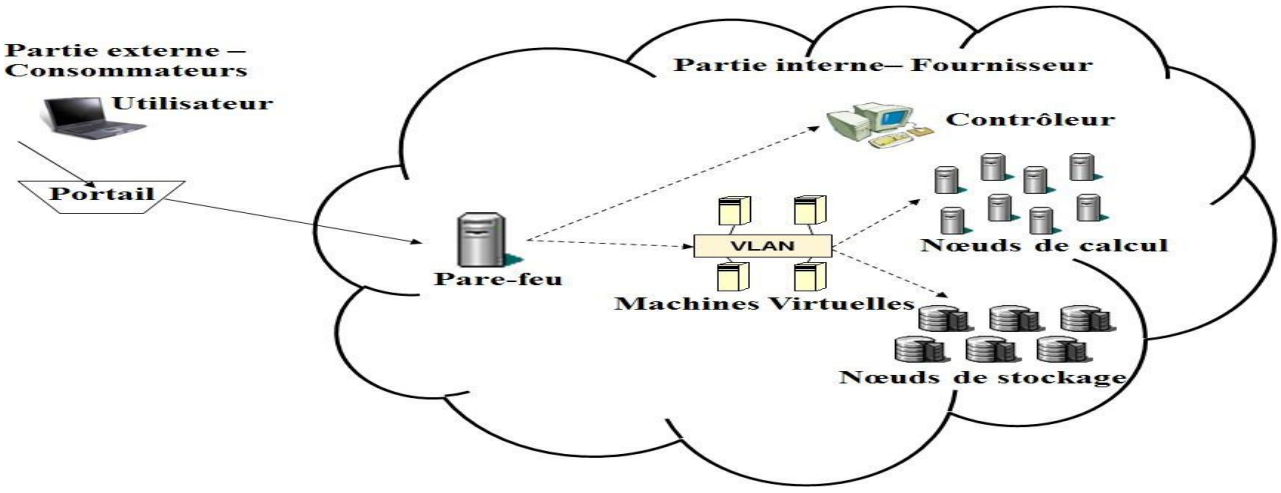

Figure.2.1. Le Cloud computing(Informatique en nuage)[Peter.M and Timothy.G].

2.2 Les principales caractéristiques des Clouds

Le modèle Cloud computing se différencie par les cinq caractéristiques essentielles suivantes [P. Mell et al, 2018][G. IT,2013].

1. **Accès réseau universel** : Un environnement de type Cloud Computing est accessible via le réseau, quel que soit le périphérique.
2. **Mise en commun (Pooling) de ressources** : Dans un environnement de type Cloud Computing, on ne pense pas en nombre de serveurs, taille de disques, nombre de processeurs..., mais en puissance de calcul, capacité totale de stockage, bande passante disponible.
3. **Elasticité** : Grace au Cloud, il est possible de disposer de plus de ressources très rapidement pour soutenir une forte demande. Inversement, au-delà de la provision de ressources, il est possible avec le Cloud de diminuer les ressources utilisées si celles-ci sont supérieures à ce qui est nécessaire.
4. **Libre-service (Self-Service)** : Dans un environnement de type Cloud Computing, il est possible à un utilisateur de consommer les services ou les ressources sans pour autant nécessiter une demande d'interventions auprès du fournisseur : équipe IT ou fournisseur externe (par exemple, un développeur qui souhaite tester son application sur une machine virtuelle représentative d'un poste standardisé de son entreprise peut, au travers d'un portail web, provisionner ou utiliser une machine).
5. **Service mesurable ou facturable** : Dans un environnement de type Cloud Computing, le fournisseur de la solution est capable de mesurer de façon précise la consommation des différentes ressources (CPU, stockage, ...) ; cette mesure lui permet de facturer à l'usage le client [Sylvain Caicoya et al, 2011].
6. **Basé Virtualisation :** Un système Cloud Computing est un système complètement virtuel. La virtualisation désigne l'abstraction des détails du matériel physique et fournit des ressources virtuelles pour les applications de haut niveau. Elle permet la mise en commun des ressources dans des centres de données.
7. **Sécurité efficace :** dans tous les systèmes, la sécurité joue un rôle très important, surtout dans le cas où les données sensibles résident dans le Cloud. La perte ou l'accès illégal sur les données peuvent donner des effets mal entendus spécialement pour les données. Pour cela, les chercheurs et les fournisseurs de Cloud prennent en considération ce point par l'introduction des architectures de sécurité, politiques de chiffrement et d'authentification très efficaces [C. N. Höfer et al,2011].

2.3 Modèles de déploiement

Les modèles de déploiement du Cloud Computing sont distingués en fonction de l'utilisation des ressources physiques par les intervenants. Les ressources peuvent être localisées chez l'utilisateur ou chez un fournisseur, peuvent être partagées ou non, et peuvent être pour des entreprises ou pour des autres types d'utilisateurs. Pour cette raison, le Cloud offre quatre modèles ou typologies de déploiement. (Voir Figure 2.2) :

2.3.1 Le Cloud privé

- Est un mode de consommation de l'informatique s'appuyant sur des ressources (serveur, stockage, réseau, licences logicielles…) mises à disposition exclusive d'une entreprise.
- Il peut se déployer sous deux formes distinctes :
  a) *Cloud privé interne* : hébergé par l'entreprise elle-même, parfois partagé ou mutualisé en mode privatif avec les filiales.
  b) *Cloud privé externe* : hébergé chez un tiers, il est entièrement dédié à l'entreprise et accessible via des réseaux sécurisés de type VPN (Réseau virtuel privé).

2.3.2 Le Cloud public

Le cloud public est accessible par Internet et géré par un prestataire externe. Il est ouvert au public ou à de grands groupes industriels. Cette infrastructure est possédée par une organisation qui vend des services Cloud[S. Srinivasan,2014].

2.3.3 Le Cloud hybride ou mixte

Le Cloud hybride associe l'utilisation, pour une même entreprise, d'un Cloud privé et d'un Cloud public. Ces infrastructures sont liées entre elles par la même technologie qui autorise la portabilité des applications et des données [A. Lee-Post et al,2014].

2.3.4 Le Cloud communautaire

Le Cloud communautaire est dédié à une communauté professionnelle spécifique incluant partenaires, sous-traitants, etc, pour travailler de manière collaborative sur un même projet ou Cloud gouvernemental dédié aux institutions étatiques.

On peut distinguer deux types de Cloud communauté. Le premier est le modèle fédéré où toute ressource inutile d'une organisation peut être utilisée par une autre organisation membre à la communauté. Le deuxième c'est le tiers de confiance, où un "broker" est le responsable de l'acquisition des différents services essentiels et les met à la disposition de tous les membres [M. Abdelhak,2018].

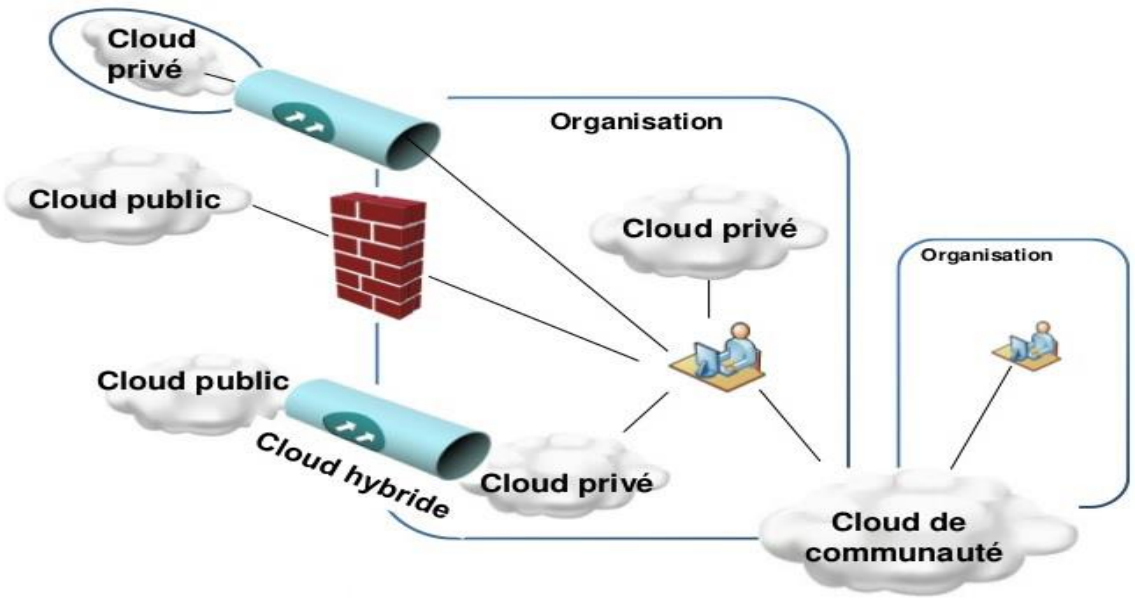

Figure.2.2. Les modèles de déploiement dans le Cloud computing

3. Modèles de service

Dans les Cloud computing, Il existe trois types de services : IaaS (Infrastructure as a Service), PaaS (Plateform as a Service) et SaaS (Software as a Service), comme il est montré dans la Figure 2.3.

3.1 IaaS (Infrastructure as a Service)

L'IaaS (ou l'infrastructure en tant que service) met à disposition des utilisateurs une infrastructure avec des ressources informatiques prêtes à l'emploi tel que des serveurs, des équipements réseau, de l'espace de stockage, etc. Les matériels et ressources informatiques des clients sont dématérialisés. L'IaaS est vu comme une abstraction d'un centre de données sur laquelle les consommateurs déposent leur environnement de production et leurs applications. Les clients disposent donc d'un environnement de production disposant de capacités matérielles qui peuvent s'adapter à tout moment de l'évolution des besoins. Du point de vue du client, ces ressources sont donc infinies, de nouvelles ressources pouvant être mises à leur disposition à la demande (et facturées à l'usage) en cas d'accroissement de la demande. Inversement, elles peuvent être retirées si elles ne sont plus nécessaires [H.D, 2009].

3.2 PaaS (Platform as a Service)

La PaaS (ou la plate-forme en tant que service) est un modèle de cloud computing où le fournisseur de services met à disposition de ses utilisateurs des plate-formes d'exécution, de déploiement et de développement d'applications qui sont prêtes à l'emploi et fonctionnelles.

PaaS fournit un niveau d'abstraction supplémentaire par rapport à IaaS : en plus de l'infrastructure matérielle, l'hébergement et le framework d'application sont dématérialisés. Les clients "poussent" leurs applications existant dans le Cloud, ou développent de nouvelles applications avec les outils proposés par les fournisseurs tels que *google App Engine* (plate-forme de conception et d'hébergement d'applications web basées sur les serveurs Google). Les utilisateurs ont à leur disposition des systèmes informatiques performants, configurés et maintenus par le fournisseur cloud pour y développer leurs propres applications. Les développeurs ne se préoccupent plus du déploiement, de la mise à jour et licence des logiciels ou de la nature des techniques pour exécuter leurs applications. La gestion des systèmes d'exploitation sous-jacents, des serveurs d'applications sou des serveurs web sont aussi à la charge du fournisseur de services. Ils se concentrent uniquement sur l'architecture et le codage de leurs applications, permettant ainsi d'améliorer nettement la productivité. Cette plate-forme favorise également la collaboration entre les salariés du client [P. Guillaume et al,2012].

3.3 SaaS (Software as a Service)

Les clients de ce modèle sont aussi bien des utilisateurs personnels que des entreprises. Ce modèle de service correspond à celui que nous rencontrons communément dans le Cloud public. Il dérive du monde des ASP (Application Service Provider) qui se sont développés initialement dans le monde du Web. Pour beaucoup de personnes et d'utilisateurs [Jean-Louis Caire et al,2014], le Cloud se résume uniquement à cet aspect ! Ce modèle représente l'accès à un service applicatif et à ses fonctionnalités associées. Tenons comme exemples : les réseaux sociaux, la messagerie personnelle, les applications bureautiques et l'impression photo. Pour un public de masse, le fournisseur propose des niveaux de services génériques peu ou pas personnalisables. Ceci lui permet de proposer des prix attractifs d'entrée de gamme. Une politique de prix d'entrée de gamme, des niveaux de services quelques fois floues où des clients en manque de maturité peuvent poser des soucis de contractualisation et d'engagement. Ce point est crucial pour les enjeux du Cloud [C. N. Höfer et al,2011].

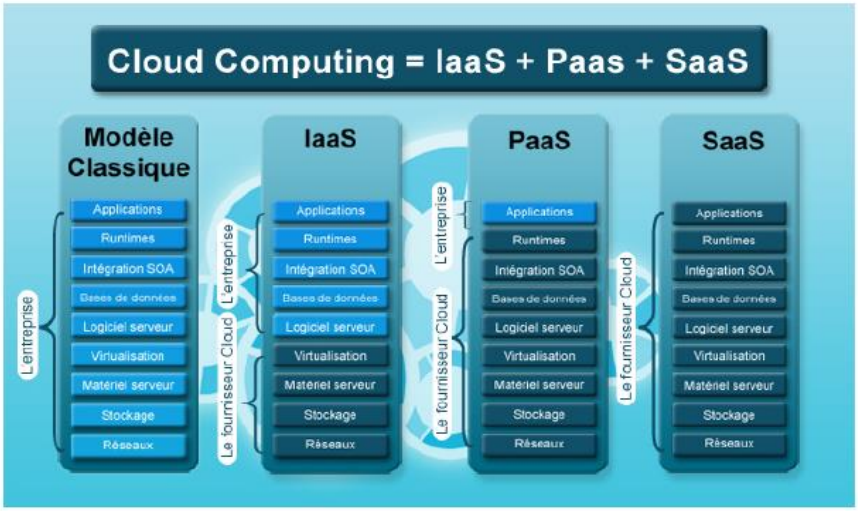

Figure.2.3. Les modèles de services dans le Cloud computing [cloud/cloud-kesako,2016]

3.4 Sécurité dans les Cloud computing

Les avantages du Cloud computing sont aujourd'hui une évidence. Les plus no- tables sont : la réduction des couts de maintenance de l'infrastructure informatique, la réduction de la consommation énergétique, la disposition rapide d'une plateforme prête à l'emploi pour le déploiement des applications, la disposition d'une solution de sauvegarde simple et accessible à tous, même aux non-informaticiens. Cependant, devant toutes les possibilités offertes par ce nouveau concept de l'informatique, il demeure des réticences dans son adoption. Ces réticences sont liées, pour la plupart, au facteur de sécurité, qui reste encore un véritable challenge [Guillaume Sigui,2014].

Le Cloud computing est une approche informatique qui consiste à exploiter via Internet (ou tout autre réseau WAN) des ressources système et applicatives (serveurs, stockage, outils de collaboration et d'administration, etc.). Ces ressources distantes sont dites en Cloud. Plusieurs études menées par des spécialistes tels que ISACA (Information Systems audit and Control Association) et CSA (Cloud Security Alliance) ont permises d'identifier douze points qui constituent les menaces majeures à la sécurité des données et à celles des applications en Cloud [Guillaume Sigui,2014].

Ce sont notamment :

1. L'existence de brèches de sécurité tant sur l'une des couches logiques du datacenter que celles issues d'erreurs humaines ;
2. La fragilité dans la gestion des accès et des identités, bien que certains fournisseurs renforcent les interfaces d'authentification avec d'autres moyens tels que les certificats, les smartcards, la technique OTP et bien d'autres ;

3. L'utilisation d'API non sécurisées pour l'intégration des applications avec les services Cloud ;
4. L'exploit de vulnérabilités des systèmes d'exploitation sur les serveurs du Cloud et même sur les applications hébergées ;
5. Le piratage de compte, qui est un vieux type d'attaque informatique, vient avec une forte recrudescence depuis l'avènement d'Internet et encore celui du Cloud computing ;
6. Une action malveillante initiée en interne dans les effectifs du fournisseur. Une personne malveillante dans l'équipe de gestion du Datacenter peut facilement nuire à la confidentialité et l'intégrité des environnements hébergés ;
7. Les menaces persistantes avancées (APT : Advanced Persistent Threats) qui consistent en une forme d'attaque ou le hacker réussit à installer d'une façon ou d'une autre un dispositif dans le réseau interne de l'organisation, à partir duquel il peut extirper des données importantes ou confidentielles. C'est une forme d'attaque difficile à détecter pour un fournisseur de services Cloud ;
8. La perte de données qui peut être causée par une attaque informatique (logique) du Datacenter, une attaque physique (incendie ou bombardement), une catastrophe naturelle, ou même simplement à un facteur humain chez le fournisseur de services, par exemple en cas de faillite de la société ;
9. Les insuffisances dans les stratégies internes d'adoption ou de passage au Cloud. Les entreprises ou les organisations ne prennent pas souvent en compte tous les facteurs de sécurité liés à leur fonctionnement avant de souscrire à un service Cloud. Certaines négligences, tant au niveau du développement d'application qu'au niveau de l'utilisation basique, leur sont parfois fatales ;
10. Utilisation frauduleuse des technologies Cloud en vue de cacher l'identité et de perpétrer des attaques à grande échelle. Généralement, il s'agit de comptes créés pendant les périodes d'évaluation (la plupart des fournisseurs d'accès à Internet (FAI) proposent 30 jours d'essai gratuits) ou des accès achetés frauduleusement ;
11. Le déni de service qui est une attaque qui consiste à rendre indisponible un service par une consommation abusive des ressources telles que les processeurs, la mémoire ou le réseau. L'idée, pour le pirate, c'est de réussir à surcharger les ressources du Datacenter en vue d'empêcher d'autres utilisateurs de profiter des services ;
12. Les failles liées à l'hétérogénéité des techniques imbriquées dans l'architecture internent du Cloud, et l'architecture externe d'interfaçage avec les utilisateurs.

3.5 Les avantages et les inconvénients du Cloud Computing

3.5.1 Les avantages

- **Un démarrage rapide :** le cloud computing permet de tester le business planrapidement, à coûts réduits et avec facilité.
- **L'agilité pour l'entreprise :** Résolution des problèmes de gestion informatique simplement sans avoir à vous engager à long terme.
- **Un développement plus rapide des produits :** réduisons le temps de recherche pour les développeurs sur le paramétrage de s'applications.
- **Mises à jour et évolutivité** : pas besoin de mettre à jour l'ensemble des postes pour ajouter de nouvelles fonctionnalités, il suffit de mettre à jour l'application réseau et tous les utilisateurs bénéficient des nouveautés et des corrections. Il en résulte une plus grande cohérence de la méthode de travail et des documents produits par l'ensemble des contributeurs de l'organisation.
- **Mise en commun des ressources** : chaque utilisateur peut contribuer à l'enrichissement des données et des expériences de l'ensemble si des outils collaboratifs sont mis en place. Cet avantage facilite la gestion et la transmission des connaissances dans les entreprises.
- **Sécurité** : si les documents ne sont plus présents en local (et que l'utilisateur ne sauvegarde pas ses identifiants de connexion sur son poste) on évite le problème de l'ordinateur perdu ou piraté et des documents confidentiels perdus dans la nature.
- **Puissance de calcul** : le système déporté sur un réseau de serveurs offre une bien meilleure efficacité de calcul qu'un poste seul. Cette fonctionnalité est développée dans les domaines de la compression ou de l'application d'effets vidéo, plus largement dans le partage de vidéos (Youtube …etc.) mais aussi dans le jeu en ligne (encore à l'état de test pour le grand public, le goulot d'étranglement étant la bande passante de l'utilisateur). Ce domaine est toutefois réellement intéressant dans le cadre d'application nécessitant une puissance de calcul important pour une utilisation mobile.
- **Mobilité** : l'utilisateur peut à tout moment et à partir de n'importe quel appareil se connecter à ses applications et son workflow. Il peut y accéder à partir de n'importe quel type d'appareils à condition que celui-ci soit doté d'un navigateur [Mathur, P,2010].

3.5.2 Inconvénients

- **La bande passante peut faire exploser votre budget :** la bande passante qui serait nécessaire pour mettre cela dans le Cloud est gigantesque, et les coûts seraient tellement importants qu'il est plus avantageux d'acheter le stockage nous-mêmes plutôt que de payer quelqu'un d'autre pour s'en charger.
- **Les performances des applications peuvent être amoindries :** un Cloud public n'améliorera définitivement pas les performances des applications.
- **La fiabilité du Cloud :** un grand risque lorsqu'on met une application qui donne des avantages compétitifs ou qui contient des informations clients dans le Cloud,
- **Taille de l'entreprise :** si votre entreprise est grande alors vos ressources sont grandes, ce qui inclut une grande consommation du cloud.
- **Sécurité** : la plateforme cloud, si elle est externe (non installée sur le réseau interne ou avec une ouverture extérieure) doit être suffisamment sécurisée pour éviter le risque d'intrusion, de vol des données par piratage. L'autre risque est qu'un utilisateur oublie de se déconnecter sur un appareil accessible par des éléments externes à l'organisation. Il faut dans ce cas prévoir une déconnexion automatique en cas de non-activité du compte et bien segmenter les droits utilisateurs afin que ces derniers ne puissent accéder qu'aux données des projets dans lesquels ils sont impliqués. Plus généralement, une clause de confidentialité et la confiance dans son personnel sont primordiales pour que les données ne fuitent pas de manière volontaire [Jian J.Z et al, 2011].

4. Conclusion

Dans ce chapitre, nous avons donné un aperçu détaillé sur l'approche du Cloud computing. Nous avons présenté des définitions de cette notion et ses caractéristiques essentielles, nous avons montré comment le Cloud offre un large choix de services informatiques à la demande et avec facturation à l'usage aux utilisateurs selon leurs besoins. Ces services se présentent sous la forme d'un logiciel, plates-forme ou infrastructure et qui sont déployés sous quatre modèles possibles qui sont : le Cloud privé, le Cloud public, le Cloud communautaire et le Cloud hybride. Ensuite, nous avons présenté ses modèles et nous avons discuté la sécurité.

Grâce aux avantages offerts par le Cloud, les utilisateurs finaux trouvent que cette technique est un bon choix pour l'utilisation des services. Malgré, toutes ces solutions produites par le Cloud, il existe toujours des limites. Nous avons présenté les principaux avantages et inconvénients de la technique Cloud computing.

Le majeur problème dans le Cloud computing est la tolérance aux fautes. La tolérance aux fautes est une préoccupation majeure pour garantir la disponibilité et la fiabilité des services critiques ainsi que l'exécution de l'application. Afin de minimiser l'impact de l'échec sur le système et l'exécution de l'application, dans le chapitre suivant en vas présenter les différentes techniques de tolérances aux fautes.

Chapitre III :

La tolérance aux fautes dans le cloud

1. Introduction

La capacité d'un système à fonctionner malgré une défaillance d'une de ces composantes est appelée prédiction des fautes (parfois nommée tolérance aux fautes).

Puisqu'il est impossible d'empêcher totalement les fautes, une solution consiste à mettre en place des mécanismes de redondance, en dupliquant les ressources critiques, ou des techniques de sauvegarde et de retour en arrière. Lorsqu'une des ressources tombe en panne, les autres ressources prennent le relais afin de laisser le temps aux administrateurs du système de remédier à l'avarie [Malej ,2009].

Le Cloud est exposé à de différents types de fautes. Ainsi, diverses techniques de tolérance aux fautes peuvent être utilisées. Puisqu' il est impossible d'empêcher totalement les fautes, une solution consiste à mettre en place des mécanismes de tolérance aux fautes [Malej,2009].

Dans ce chapitre, nous présentons les différents concepts et les grandes lignes liés à la tolérance aux fautes ainsi que les diverses techniques de gestion des fautes utilisées.

2. Notion de sureté de fonctionnement

La tolérance aux fautes s'inscrit dans le contexte plus large de la sureté de fonctionnement. La sureté de fonctionnement est définie comme la capacité de fournir un service dans lequel un utilisateur peut raisonnablement placer sa confiance **[**Crouzet et al,2006**]**.

Pour bien comprendre les tenants et les aboutissants de la sureté de fonctionnement, nous présentons dans ce qui suit ses attributs, ses entraves et ses moyens (voir la figure 3.1.).

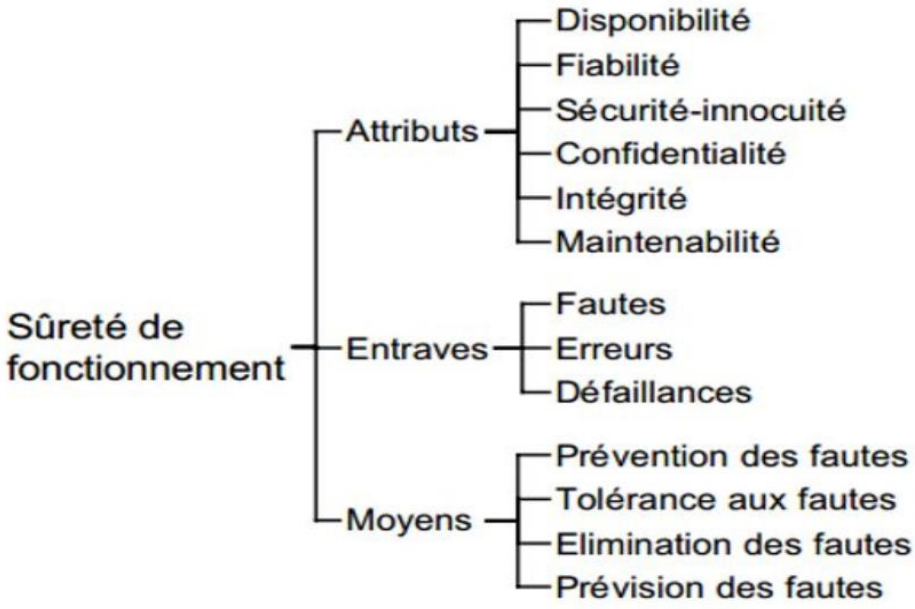

Figure.3.1: L'arbre de la sureté de fonctionnement[Crouzet et al,2006].

2.1 Attributs de la sureté de fonctionnement

Les attributs de la sureté permettent d'évaluer la qualité du service fourni par un système. Dans [17], six attributs de la sureté de fonctionnement sont définis :

- **Disponibilité :** c'est la propriété requise par la plupart des systèmes surs de fonctionnement.
- **Fiabilité :** cet attribut évalue la continuité du service,
- **Sécurité-innocuité :** Evalue la continuité du service avant l'arrivée d'une défaillance catastrophique.
- **Confidentialité :** cet attribut évalue la capacité du système à fonctionner en dépit de fautes intentionnelles et d'intrusions illégales ;
- **Intégrité :** l'intégrité d'un système définit son aptitude à assurer des altérations approuvées des données ;
- **Maintenabilité :** cette propriété d'écrit la souplesse du système vis-à-vis des modifications apportées en vue de sa maintenance.

2.2 Entraves

Il existe trois types d'entraves à la sureté de fonctionnement qui sont [Crouzet et al, 2006] : les fautes, les erreurs et les défaillances qui sont liées par des relations de causalité illustrées sur la Figure 3.2.

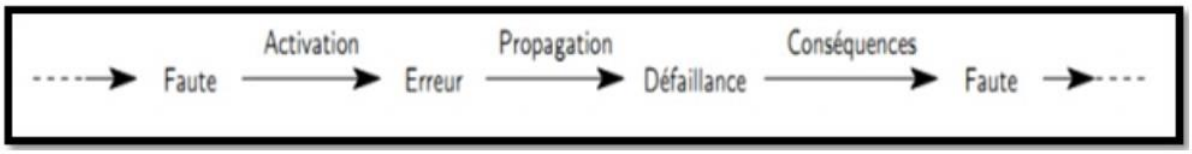

Figure.3.2: La chaine fondamentale des entraves a la sureté de fonctionnement **[***Laprie ,1994]*.

1. Fautes : Une faute est caractérisée par sa nature, son origine et son étendue temporelle [Laprie ,1994].
2. L'erreur : correspond à un état incorrect dans un système. Elle se produit âpresl'activation d'une faute ou à cause d'une autre erreur.
3. Défaillances **:** une défaillance dénote l'incapacité d'un élément du systèmeà assurer le service spécifié par l'utilisateur [Laprie ,1994].

2.3 Moyens d'assurer la sureté de fonctionnement

Il s'agit des méthodes et techniques permettant de fournir au système l'aptitude à délivrer un service conforme à l'accomplissement de sa fonction, et de donner confiance dans cette aptitude. **[**Ali,2005**]** Le développement d'un système sûrde fonctionnement passe par l'utilisation combinée de ces méthodes qui peuvent être classées en quatre catégories [haouati, 2010] qui sont **:** la prévention des fautes, L'élimination des fautes, la prévision des fautes, la tolérance aux fautes ou aux fautes.

3. La tolérance aux fautes

3.1 Définition

La tolérance aux fautes est une méthode de conception qui permet à un système de rester fonctionnelle, éventuellement de manière réduite au lieu de tomber complètement en faute et rester plus ou moins opérationnels lorsque l'un de ses composants ne fonctionne plus correctement avec une réduction du débit ou une augmentation du temps de réponse.

3.2 Classification des fautes

Les fautes peuvent être classées selon leur degré de gravité, leur degré de permanence et leur nature. Une première classification des fautes selon le degré de gravité des défaillances est proposée et permet de différencier :

- **Les fautes franches (crash faults) :** soit le système fonctionne normalement (les résultats sont corrects), soit-il ne fait rien.
- **Les fautes par omission :** le système perd des messages entrant ou sortant d'un processus.

  Les fautes de temporisation : les déviations par rapport aux spécifications concernent uniquement le temps.
- **Les fautes par valeurs :** les résultats produits par un composant défaillant sont incorrects.
- **Les fautes byzantines :** le système peut faire n'importe quoi, y compris avoir un comportement malveillant.

Une seconde classification des fautes selon leur degré de permanence est également possible. On distingue ainsi les fautes transitoires qui se produisent de manière isolée, les fautes intermittentes qui se déclenchent aléatoirement plusieurs fois et les fautes permanentes qui persistent dès qu'elles apparaissent jusqu'à réparation.

Il existe un troisième classement qui distingue la nature des fautes selon qu'elles soient accidentelles ou intentionnelles. Les fautes accidentelles qui se produise de manière accidentelle et les fautes intentionnelles qui sont créent avec une intension qui peut être malicieuse.

4. Méthodes de tolérance aux fautes

Parmi les méthodes qui permet le développement des systèmes surs de fonctionnement La tolérance aux fautes. Elle vise à éviter les défaillances et est mise en œuvre par la détection des erreurs et le rétablissement du système.

Le traitement de l'erreur peut se faire par trois techniques [LIMAM,2012]: la reprise, la poursuite et la compensation.

- **Reprise :** est la technique la plus couramment utilisée. L'état du système est sauvegardé régulièrement. Le système est ramené dans un état sauvegardé (point de reprise) survenu avant l'occurrence d'erreur.
- **Poursuite :** consiste à rechercher un nouvel état exempt d'erreur. Ceci peut par exemple être réalisé en associant un traitement exceptionnel lorsqu'uneerreur est détectée.
- **Compensation :** nécessite que l'état du système comporte suffisamment de redondance pour permettre sa transformation en un état exempt d'erreur. Elle est transparente vis-à-vis de l'application car elle ne nécessite pas de ré-exécuter une partie de l'application (reprise), ni d'exécuter uneprocédure dédiée (poursuite)[Vial,2009].

5. Techniques de tolérance aux fautes dans les systèmes répartir

La tolérance aux fautes dans les systèmes répartis et parallèles est le plus souvent réalisée par l'emploi d'un mécanisme de redondance.

5.1 Tolérance aux fautes par sauvegarde (Checkpointing)

Le processus d'enregistrement périodique des états du système pendant l'exécution sans défaillance, habituellement au serveur de stockage stable c'est le point de reprise. Le checkpointing peut fournir la tolérance aux fautes en cas de transit et les échecs permanents puisqu'il consiste à un retour en arrière, rechargement des derniers points de reprise cohérents, et reprendre l'exécution à partir de ces points. Le système doit stocker toutes les données nécessaires sur l'état actuel de l'application en cours d'exécution et son

environnement. Une bonne récupération ne peut être garantie que si le protocole check point enregistre un état global cohérent qui est un ensemble contenant un point de reprise par processus de l'application. Un état global forme une coupe de l'exécution, c'est-à-dire l'ensemble des évènements qui ont servi à la réduction depuis les différents états initiaux vers chacun des états représentés par les points de reprise. L'état cohérent doit satisfaire les deux propriétés suivantes : la cohérence et le recouvrement[Ling,2013].

5.2 Tolérance aux fautes par journalisation

Le principe de la tolérance aux fautes par journalisation est de sauvegarder l'histoire de l'application. Les protocoles par journalisation utilisent à la fois la sauvegarde locale de l'état des processus et la journalisation des évènements déterministes pour permettre la reprise de l'application. Il est alors possible de reprendre l'exécution des processus défaillants (et uniquement des processus défaillants) à partir de leur dernière sauvegarde en rejouant les évènements non déterministes sauvegardés. Chaque processus crée un journal des évènements non déterministes qu'il observe pendant l'exécution normale (sans faute). En cas de faute, le processus défaillant est capable de reconstruire son état d'avant la faute en partant de son état initial et en rejouant les évènements non déterministes inscrits dans son journal. Les protocoles de journalisation doivent assurer la condition de « non orphelina ». Il existe trois types de tolérances aux fautes par journalisation qui sont : journalisation pessimiste, journalisation optimiste et journalisation causale [Joseph et al ,2007],[Manetho,1993].

5.3 Migration

La migration de processus est le déplacement d'un processus en cours d'exécution d'une machine source vers une machine destination, ces deux machines étant reliées par un réseau de communication et n'utilisant pas de mémoire partagée.

La migration de processus d'une machine source vers une machine destination consiste à interrompre le processus qui s'exécute sur la machine source pour extraire son contexte d'exécution, à transférer ce contexte vers la machine destination, à créer, sur la machine destination, un nouveau processus auquel on affectera le contexte d'exécution transféré et à mettre à jour les liens de communication avec les autres processus. Une fois ceci fait, le processus sur la machine source doit être détruit tandis que le processus sur la machine destination est lancé et représente le processus déplacé. La Figure 3.3 illustre brièvement le mécanisme de migration de processus.

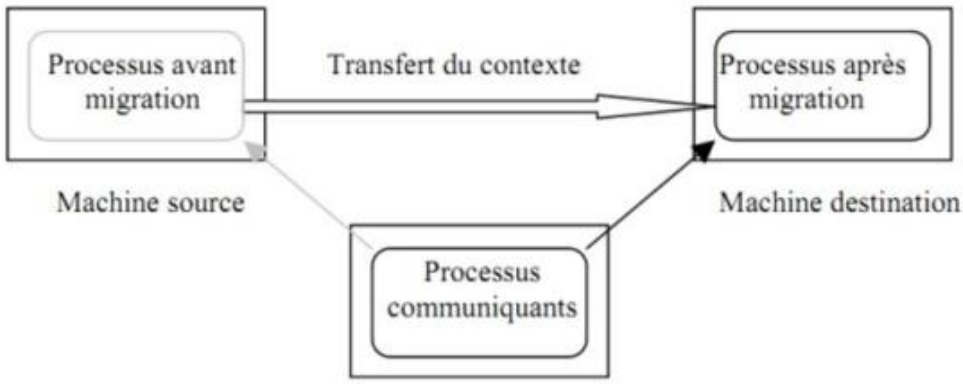

Figure 3.3 : Migration des processus

5.4 Réplication

L'idée d'utiliser la redondance dans les systèmes informatiques pour masquer les défaillances des composants a été introduite par Von Neumann en 1956 [Peres ,2008].

Avec plusieurs répliques, une entité répliquée continue à fournir un service a un client même si une ou plusieurs répliques sont défaillantes. La réplication met en œuvre un processus qui est chargé de la création, du placement et de la gestion de copies d'entités physiques et/ou logicielles. Les entités répliquées peuvent être des données, du code, des objets, des composants physiques ou une combinaison de tous ces éléments.

La création des copies ou répliques d'une entité consiste à reproduire la structure et l'état des entités répliquées. La copie d'un fichier est un autre fichier de même contenu. La copie d'un programme est un autre programme qui exécute le même code et dont l'état d'exécution est celui du programme initial [Siham,2011].

L'intérêt premier de cette réplication est que, si une donnée n'est plus disponible, le système peut continuer à assurer ses fonctionnalités en utilisant une donnée répliquée, ce qui permet d'augmenter la disponibilité des données et la tolérance aux fautes.

6. La tolérance aux fautes dans les Cloud

Les fautes apparaissant dans le Cloud doivent être détectés et prédits efficacement afin de lancer des outils pour tolérer les défaillances survenues. Les techniques de prédictions de fautes sont classées en deux grandes classe L'approche « Proactive » et l'approche « Réactive », ces deux techniques sont les plus utilisées pour prédire les fautes au niveau du Cloud.

- **L'approche Réactive**

Les techniques de tolérance aux fautes réactives sont utilisées pour réduire l'impact des défaillances sur un système lorsque les défaillances se sont réellement produites. Les techniques basées sur cette politique sont check point / Restart et retry et ainsi de suite.

- **L'approche proactive**

Cette approche de fautes permet de prévoir les fautes de manière proactive et de remplacer les composants suspects par d'autres composants de travail, évitant ainsi la récupération des fautes et des erreurs. La migration préemptive, le rajeunissement logiciel, etc. suivent cette politique.

7. Conclusion

Dans ce chapitre, nous avons dressé un état de l'art sur la sureté de fonctionnement, et nous avons donné un aperçu général sur les différentes notions de tolérance aux fautes dans les systèmes distribués et le Cloud Computing.

Notre but dans ce travail est de garantir la tolérance aux fautes dans le cloud Pour cela nous proposons dans le prochain chapitre une stratégie de tolérance aux fautes basée sur la réplication des données qui permet de garantir un bon fonctionnement du système en cas des fautes.

Chapitre IV :

Description et conception de la méthode proposée

1. Introduction

Le cloud computingest une technologie innovante dans le domaine de l'informatique distribuée qui fournit un service à la demande dans la technologie informatique. Cette technologie offre l'intégration de logiciels et de ressources qui présentent une évolutivité dynamique dans la nature. Ces systèmes sont bénéfiques, mais il y a aussi un inconvénient pour le même.

Le Cloud computing est confronté à de nombreux problèmes pour détecter et anticiper l'échec de ses services, car la tolérance aux fautes est le plus grand défi du Cloud computing. La tolérance aux fautes est un point important pour garantir la disponibilité et la fiabilité des services sérieux de l'application.

Nous proposons dans ce chapitre une méthode hybride basée sur les systèmes multi agent et la technique de réplication pour prédire les fautes dans le Cloud Computing.

2. Problématique

Le Cloud Computing est une plateforme, qui rend le traitement et le stockage disponible pour les utilisateurs finaux en tant que services. Le Cloud est un ensemble de ressources non structurées qui sont classés en trois domaines : (a) applications (ou logiciels), (b) plateforme, et (c) infrastructure. Récemment, le Cloud Computing est de plus en plus utilisé dans divers domaines de recherche, tels que l'agriculture, le commerce électronique, la santé, etc…

Le majeur problème dans le Cloud computing est la tolérance aux fautes. La tolérance aux fautes est une préoccupation majeure pour garantir la disponibilité et la fiabilité des services critiques ainsi que l'exécution de l'application. Afin de minimiser l'impact de l'échec sur le système et l'exécution de l'application.

Comme la tolérance aux fautes est très importante, nous proposons une méthode hybride de prédiction de fautes basée sur les systèmes multi agent et la technique de réplication

3. Travaux connexes

Plusieurs approches ont adressé le problème de la tolérance aux fautes. Différents outils fournissent des mécanismes de réplication pour la fiabilisation.

Hagg [HAG ,1996] propose d'utiliser des agents sentinelles pour superviser la communication entre agents, pour construire un modèle des autres agents et pour entreprendre des actions de reprise sur erreurs. Les sentinelles analysent toutes les communications dans le système pour

détecter les fautes. Cette approche reste trop coûteuse en matière de calculs et de communications. De plus, les sentinelles sont elles-mêmes sources de défaillance.

Decker et al. [Decket et al,1997] décrivent différents niveaux d'adaptation, mais se concentrent sur l'adaptation d'exécution où le clonage d'agent est utilisé comme technique d'équilibrage de charge. Les aspects fondamentaux de la tolérance aux fautes ne sont pas abordés.

Kumar et al. [Kumar et al, 2000] proposent une architecture tolérante aux fautes à base de brokers et le middleware Agent Scape offre un service de réplication pour la tolérance aux fautes [Brazier et al ,2002].

Fedoruk et al. [Fedoruk et al,2002] utilisent aussi la réplication pour la tolérance aux fautes. Leur travail implémente la stratégie de réplication passive d'une manière transparente en utilisant des « proxies ». Tous les messages envoyés et reçus par un groupe de réplication passent pour le « proxy » du groupe.

Kraus et al. [Kraus et al ,2003] définissent le problème de la tolérance aux fautes comme un problème du déploiement et proposent une approche probabiliste pour déployer les agents dans une application multi-agent. Le problème principal de ces deux travaux est que la réplication est décidée de manière statique avant l'exécution de l'application. Cela n'est pas souhaitable dans le cas des applications multi-agents dynamiques et adaptatives puisque la criticité des agents peut évoluer dynamiquement pendant l'exécution.

D'autres infrastructures logicielles pour la tolérance aux fautes adaptatives [Cuckuern et al,1998] [Favarim et al,2003,] [Kalbarczyket al ,1999] existent dans lesquelles des stratégies existantes peuvent être dynamiquement changées. Néanmoins, un tel changement doit être défini au préalable par le concepteur avant l'exécution. Un paramétrage de ces systèmes est également possible mais il reste totalement à la charge du concepteur qui peut l'effectuer de façon interactive avec le système en cours d'exécution.

4. Modèle Proposé

Les systèmes multi-agents fournissent une approche fiable et efficace pour la conception et la construction d'applications distribuées et collaboratives, un SMA peut être vu comme un ensemble d'entités autonomes, appelées agents, qui interagissent et ce cordonnet pour résoudre les problèmes donnés. Les SMA deviennent des solutions courantes pour résoudre les problèmes d'applications distribuées, tels que la tolérance aux fautes dans le Cloud

Computing. C'est pour toutes ces raisons que nous avons opté pour l'utilisation des SMA dans notre travail.

La stratégie de tolérance aux fautes proposée est basée sur le modèle de réplication et les SMA, L'intérêt premier de cette réplication est que, si une donnée n'est plus disponible, le système peut continuer à assurer ses fonctionnalités en utilisant une donnée répliquée, ce qui permet d'augmenter la disponibilité des données et la tolérance aux fautes [Drapeau,2003].

4.1 L'architecture proposée du système

L'architecture de notre système (voir figure 4.1) est basée sur deux types d'agents qui sont :

L'agent contrôleur de fautes (ACF) dont le rôle est de contrôler et d'assurer l'exécution correcte des différentes tâches du Cloud. En cas d'apparition d'une faute au niveau du data center lors de l'exécution, il informe l'agent de réplication afin de régler les fautes.

L'agent de réplication (RA) après la réception d'un message de faute l'agent va traiter les requêtes, puis il crée des répliques selon le protocole de réplication passive et enfin renvoyer les données ou les composants logiciels répliqués à l'agent contrôleur.

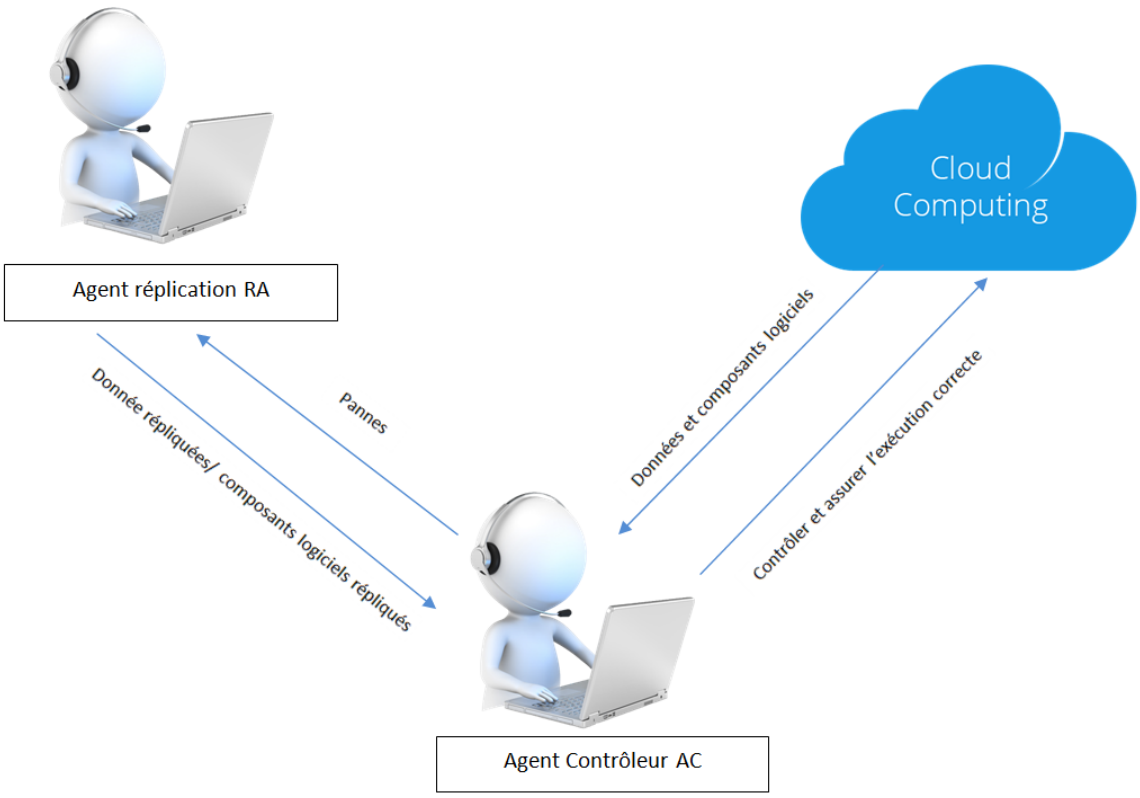

Figure.4.1: Architecture proposé du système

4.2 Fonctionnement de la méthode proposée

Dans cette section, nous présentons comment notre méthode fonctionne et répond aux objectifs fixés au début de ce travail. Ceci est fait à travers des diagrammes de séquences comme montrées dans les Figures. Comme indiqué ci-dessus, nous utilisons la technique de réplication passive basée sur la réplication des données qui est le principal rôle de notre agent AR dans les deux cas état normal et l'état suspect.

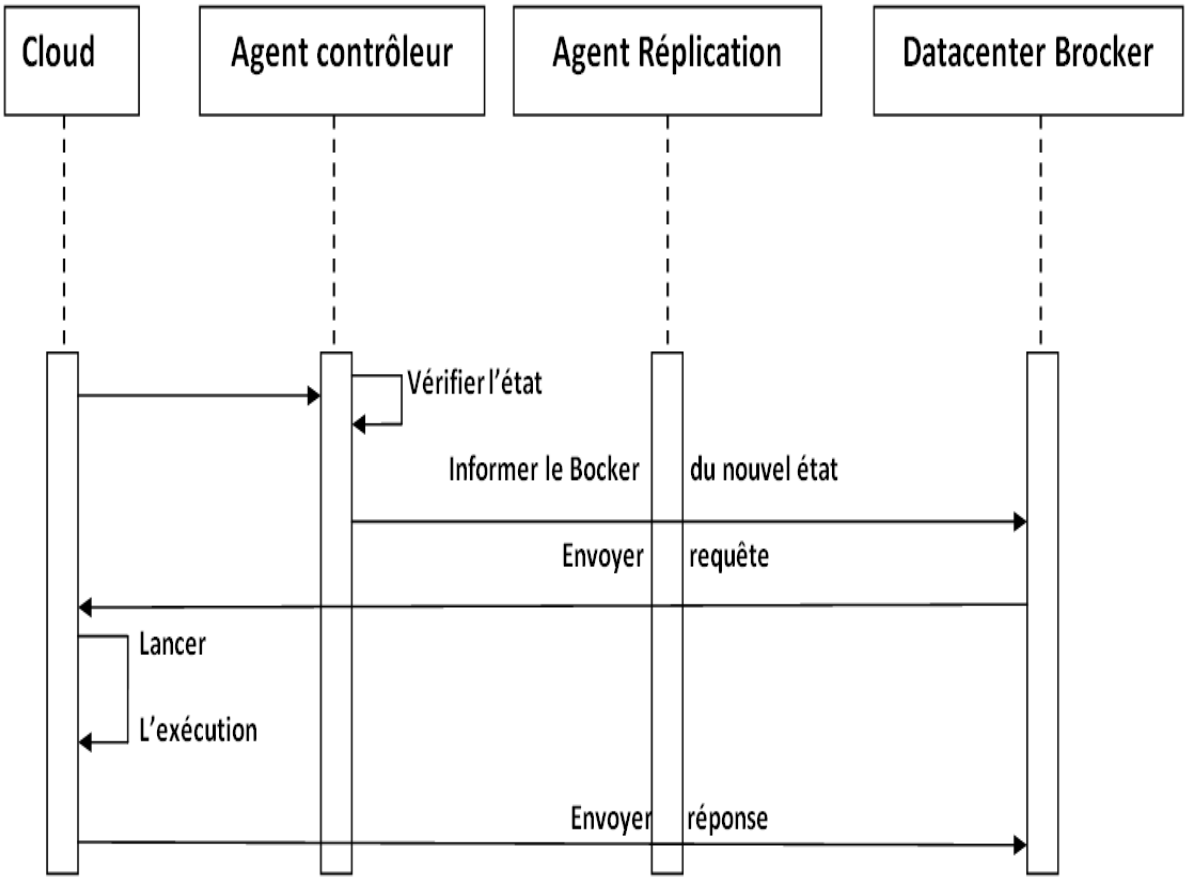

Figure.4.2 :Diagramme de séquence état Normal

Selon le diagramme de séquence de la figure 4.2, qui représente le flux de communication entre les acteurs de CloudSim et les deux agents. L'agent contrôleur vérifie l'état du data center (Cloud), si l'état est en état normal, le broker peut envoyer des requêtes à ce datacenter pour être exécuté.

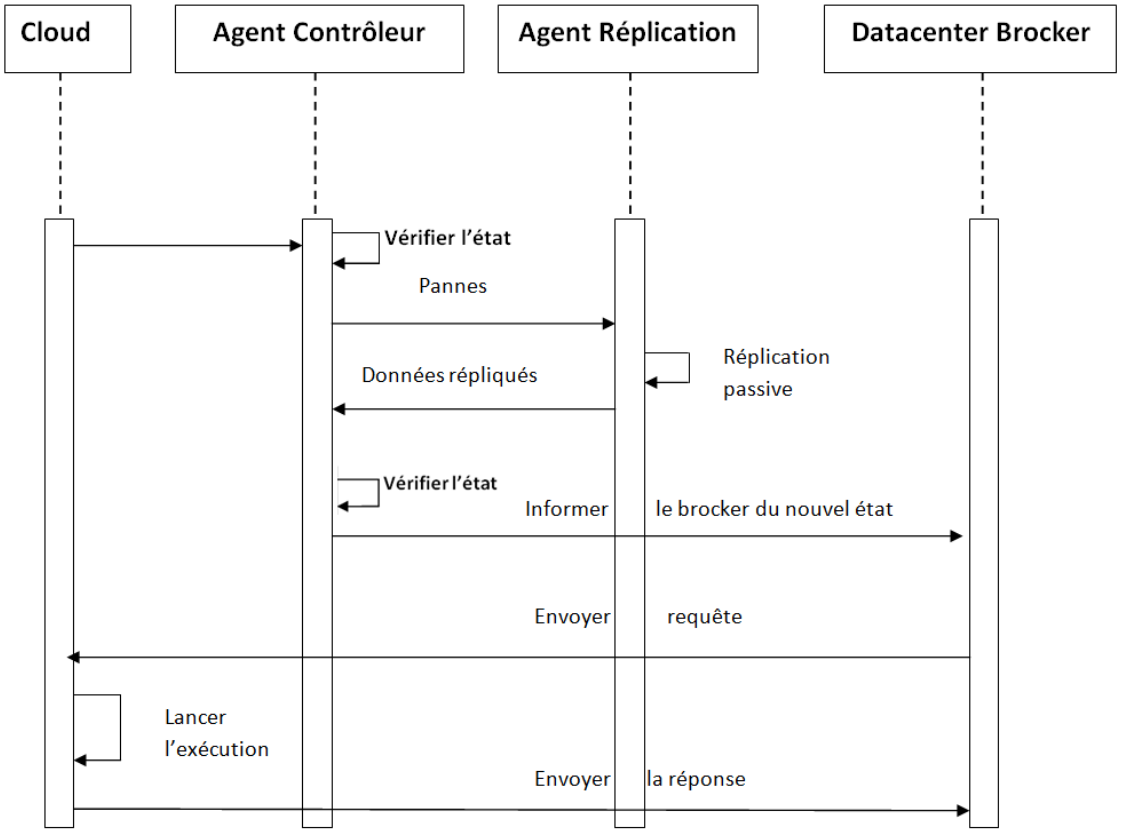

Figure.4.3 : Diagramme de séquence état Suspect

Selon le Diagramme de séquence de la figure 4.3 l'agent contrôleur vérifie l'état du data center (Cloud) lors de la détection d'une faute, l'agent contrôleur envoie les données en échec à l'agent de réplication, ce dernier vas traiter les requêtes, crée des répliques selon le protocole de réplication passive et enfin renvoyé les données ou les composants logiciels répliqués à l'agent contrôleur qui doit revérifier l'état du datacenter et continuer la procédure de l'état normal.

4.3. Comportement de l'agent contrôleur (AC)

Le rôle de l'agent contrôleur est de contrôler et d'assurer l'exécution correcte des différentes tâches du Cloud. En cas d'apparition d'une faute au niveau du data center lors de l'exécution, il doit réagir immédiatement en informant l'AR, et en lui envoyant les informations utiles tels que : à quel niveau la faute est apparue, afin de régler les fautes. (Voir Figure 4.4).

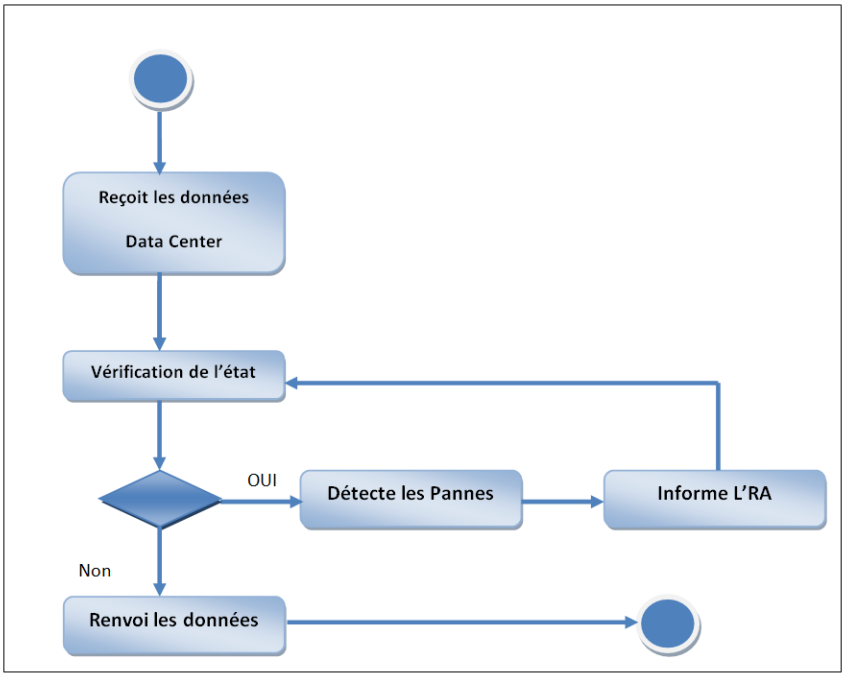

Figure.4.4 : Diagramme d'activité de l'agent controleur

4.4. Comportement de l'agent réplication (AR)

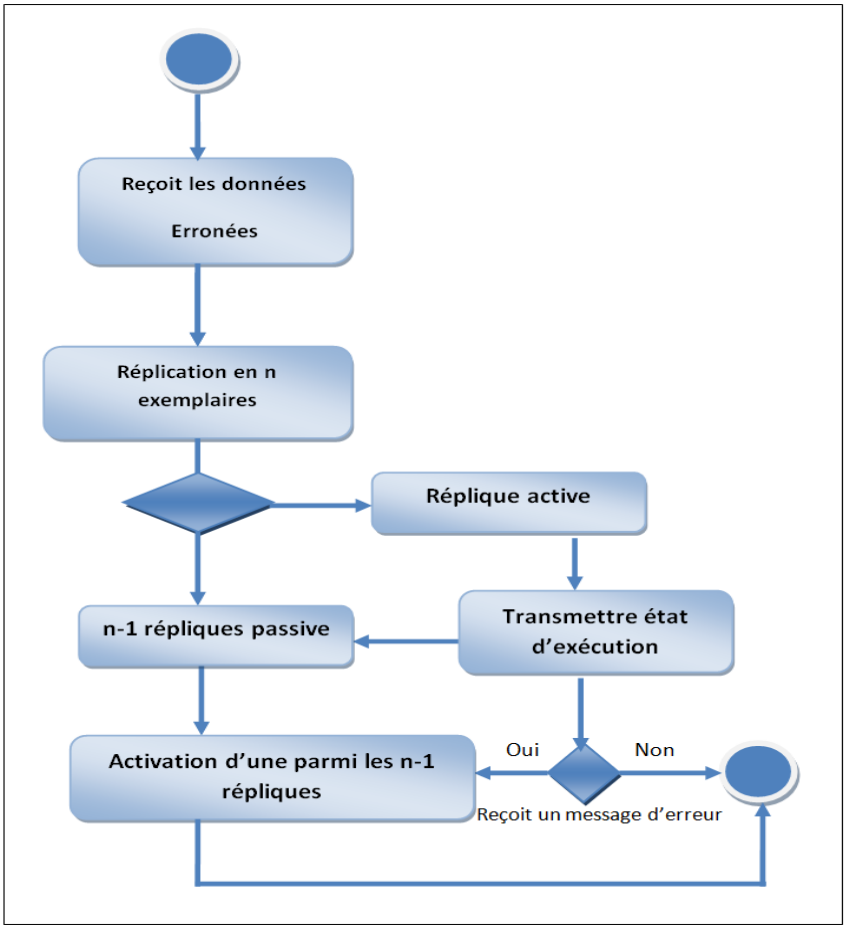

Figure.4.5 : Diagramme d'activité de l'agent réplication

Le rôle principal de l'agent réplication est de réservoir les données erronées et de faire une réplication passive où un composant logiciel est répliqué en n exemplaires, mais une seule des n répliques effectué le calcul. Les n-1 autres répliques sont passives et ne prennent la relève que si la réplique active est défaillante. Pour que cette stratégie fonctionne, il est nécessaire que la réplique active transmette aux répliques passives, à intervalles réguliers son état d'exécution. Cet état d'exécution composé d'une pile de données et de registres et stocké par chacune des répliques passives et constitue un point de reprise. Si la réplique active est défaillante, une des répliques passive est activée et reprend l'exécution du calcul à partir du dernier point de reprise enregistré. (Voir figure 4.5).

4.5. Communication inter agents

Chaque agent possède un module de communication qui gère la réception et l'interprétation des messages provenant de l'autre agent. Cette gestion concerne le médium linguistique de la transmission des messages, c'est-à-dire l'expression et l'interprétation d'un message. Nous avons retenu, comme langage d'expression des messages, le langage de communication ACL de la FIPA.

Un agent envoie un message dans deux cas de figure :

- S'il a reçu un message nécessitant d'être traité, il construit alors un message pour répondre à celui qu'il a reçu et l'envoie à l'émetteur de ce dernier ;
- S'il exécute une action dont les effets consistent en l'envoi d'un ou plusieurs messages, alors il construit ces messages et les envoie aux destinataires spécifiés.

Les messages échangés entre les agents constituant notre architecture sont illustrés dans la figure 4.6. Le contenu du message envoyé par l'agent contrôleur à l'agent réplication sont les données en échec (données erronées) et d'autre informations comme à quel niveau la faute est apparue. En conséquence, les messages envoyés par l'agent réplication à l'agent contrôleur contiennent la liste des données répliquées.

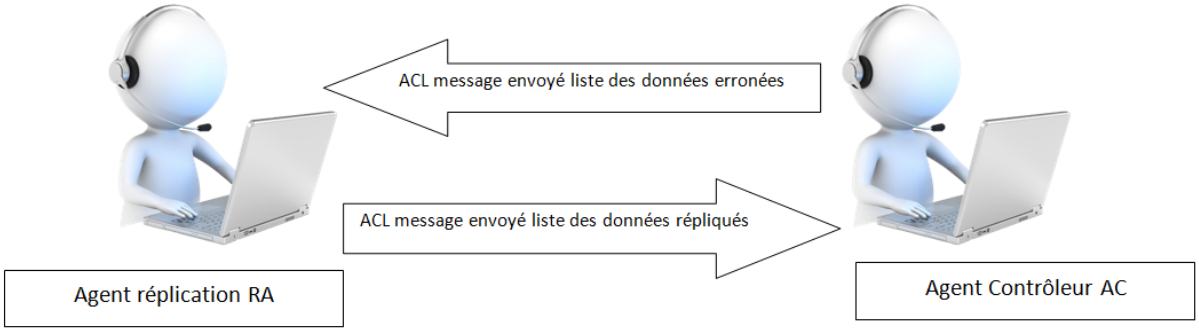

Figure.4.6 : Communication entre agents en utilisant ACLMessage

4.6 Description de la Technique de réplication utilisée

Cette section décrit l'architecture proposée et la mise en œuvre pour supporter la tolérance aux fautes dans le Cloud Computing. Dans ce travail nous proposons une stratégie de réplication passive de données pour résoudre cette problématique.

La réplication **passive** distingue deux comportements d'un composant répliqué : la copie primaire et les copies **secondaires** (backups). La copie primaire c'est la seule qui effectue tous les traitements. Les copies secondaires, oasives, surveillent la copie primaire. En cas de défaillance de la copie primaire, une copie secondaire devient la nouvelle copie primaire.

4.6.1 Principe

La réplication passive est définie ainsi [Wiesmann et al,1999] :

– Réception des requêtes : la copie primaire est la seule à recevoir les requêtes ;
– Traitement des requêtes : la copie primaire est la seule à traiter les requêtes ;
– Émission des réponses : la copie primaire est la seule à émettre les réponses.

La figure 4.7 illustre ce principe. Le client envoie la requête uniquement à la copie primaire S1. Celle-ci traite la requête, construit un point de reprise et l'envoie à l'aide d'un multicast fiable assurant l'ordre FIFO, aux copies secondairesS2 et S3 en précisant le changement de son état (message update). Après la mise à jour de leurs états, les copies secondaires envoient un ack à la copie primaire. La copie primaire envoie la réponse au client après réception des "ack" de toutes les copies secondaires. Le point de reprise permet de synchroniser l'état des copies secondaires avec celui de la copie primaire puisque celle-ci est la seule qui communique avec le reste du système.

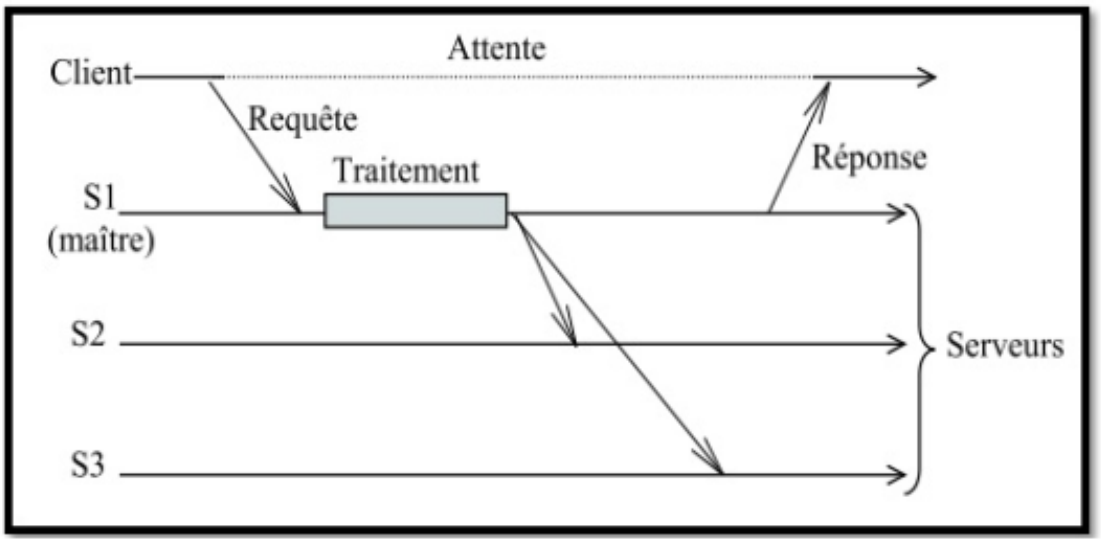

Figure.4.7: Principe de réplication passive

4.6.2 Tolérance aux fautes

La réplication passive où un composant logiciel est répliqué en n exemplaires ,mais une seule des n réplique effectue le calcul. Les n-1 autres répliques sont passives et ne prennent la relève que si la réplique active est défaillante. Pour que cette stratégie fonctionne, il est nécessaire que la réplique active transmette aux répliques passives, à intervalles réguliers son état d'exécution. Cet état d'exécution composé d'une pile, de données et de registres est stocké par chacune des répliques passives et constitue un point de reprise. Si la réplique active est défaillante, une des répliques passive est activée et reprend l'exécution du calcul à partir du dernier point de reprise enregistré.

4.7 La stratégie de réplication de données

4.7.1 Algorithme Création des répliques

L'idée générale est de créer de nouvelles répliques selon une certaine disponibilité exigée par l'utilisateur. Cette création vérifie certaines conditions comme le budget et le coût de réplication (voir figure 4.8).

Algorithme 1 : Algorithme de création des répliques

Entrées : Budget, cout de réplique

Sorite : le nombre des répliques

```
calculer un estimation du temp de réponse
calculer SLA
Si SLA >temp de réponse alors
    autoriser la cloudlet pour accéder au fichier() ;
Sinon //SLA<temp de réponse
  calculer Budget local
Si Budget > coût de réplication
    Créer réplique .
     autoriser la cloudlet pour accéder au fichier
Sinon Ajouter la liste d'attente
Mise à jour de Budget
```

Figure 4.8 : Algorithme Création des répliques.

1. Calcul du temps de réponse moyenne : les requêtes envoyées par l'utilisateur sont exécutées par les machines virtuelles, ces requêtes ont besoin des données pour terminer leurs exécutions. Le temps de réponse de chaque requête est estimé par la formule suivante :

Temps de réponse = T1 + T2 + T3

Où :

T1 : est le temps d'exécution.

T2 : est le temps d'accès.

T3 : est le temps d'attente.

Pour estimer le temps d'exécution d'une requête dans une machine virtuelle, nous utilisons cette formule : $temps\ d'execution = \frac{tailledecloudlet}{lavitesse}$

Où la taille de cloudlet représente le nombre d'instruction de la requête Et vitesse, c'est la vitesse d'exécution de la machine virtuelle donnée par MIPS.

a) Calcul du temps d'attente de la cloudlet :

Pour déterminer le temps d'attente moyenne de la requête req, nous devons déterminer en premier temps le nombre des requêtes qui sont dans la liste d'attente. Puis nous devons calculer le temps restant pour libérer les ressources, c'est-à-dire le temps nécessaire pour terminer les requêtes (temps d'accès à la donnée) qui sont en cours d'exécution. Il faut aussi déterminer le nombre de répliques de la donnée dans ce Datacenter. Le temps d'attente moyenne est la somme de ces deux temps ;

b) Le temps d'accès : Le temps d'accès est le temps nécessaire pour accéder à la donnée

2. Calcul SLA : Les exigences de la qualité de service sont extrêmement importantes pour le Cloud Computing. Elles sont généralement formalisées sous forme de SLA

3. Comparaison temp de réponse et SLA : Si le temps de réponse est inférieur au SLA c'est à dire la qualité de service est vérifiée avec les ressources actuelles, la requête peut accéder à la réplique. Par contre si le temps de réponse de la requête dépasse le SLA, dans ce cas la qualité de service n'est pas assurée avec le nombre des répliques actuelles.

Budget local = B1 + B2 + B3.....

Cette étape consiste à comparer le budget local avec le coût de création du nouvelle réplique. Si le budget dépasse le coût de création, dans ce cas le fournisseur garantie une marge de bénéfice. Donc le Datacenter peut créer des nouvelles répliques pour augmenter la disponibilité des données et de garantir la qualité de service. Dans le cas où le budget est inférieur au coût de création de nouvelle réplique. Le Datacenter informe les utilisateurs que leur requête ne peut pas être exécutée avec la qualité exigée.

4.8 Diagramme d'utilisation

Nous présentons maintenant une vue globale de notre simulateur qui permet de détailler les différentes étapes effectuées pour réaliser une simulation. La Figure 4.9représente les fonctionnalités d'utilisation nécessaires à l'application. Le diagramme des cas d'utilisation, est une modélisation d'une fonctionnalité du système, il se détermine en observant et en précisant acteur par acteur les séquences d'interactions avec le système. Les cas d'utilisation

décrivent les fonctions du système selon le point de vue des utilisateurs. Le cas d'utilisation spécifie une séquence d'actions tel que la configuration de la Cloud et afficher le résultat.

Un acteur : est une entité externe au système, dans notre cas c'est l'utilisateur (user) qui joue un rôle important, On a un type d'acteur.

User : l'acteur qui configure le Cloud et lance la simulation.

DC : Datacenter.

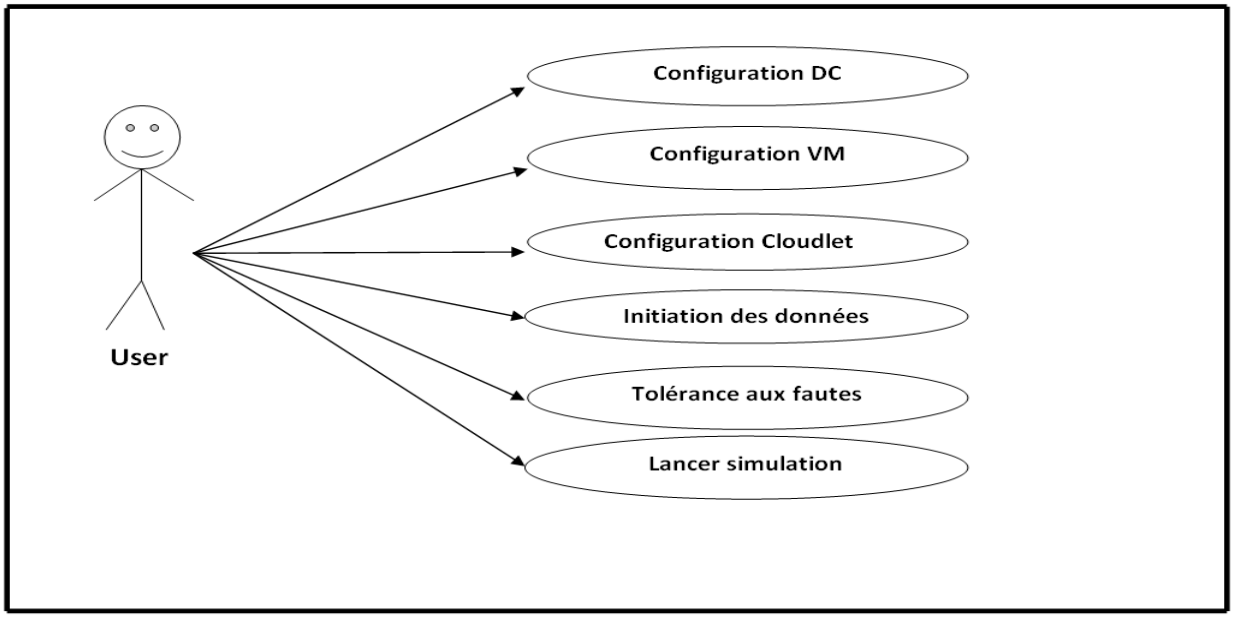

Figure.4.9 : Diagramme cas d'utlisation

4.9 Diagramme de classes

Le diagramme de classes permet de représenter la structure du système comme un ensemble de relations entre classes. Le simulateur CloudSim est composé de plusieurs lasses que nous pouvons classer en deux catégories : les classes que nous avons modifiées sont montrées en bleu dans la Figure 4.10 comme la classe Datacenter, Datacenter Broker, . . . et les classes que nous avons ajoutées sont présentées en rouge dans la même Figure

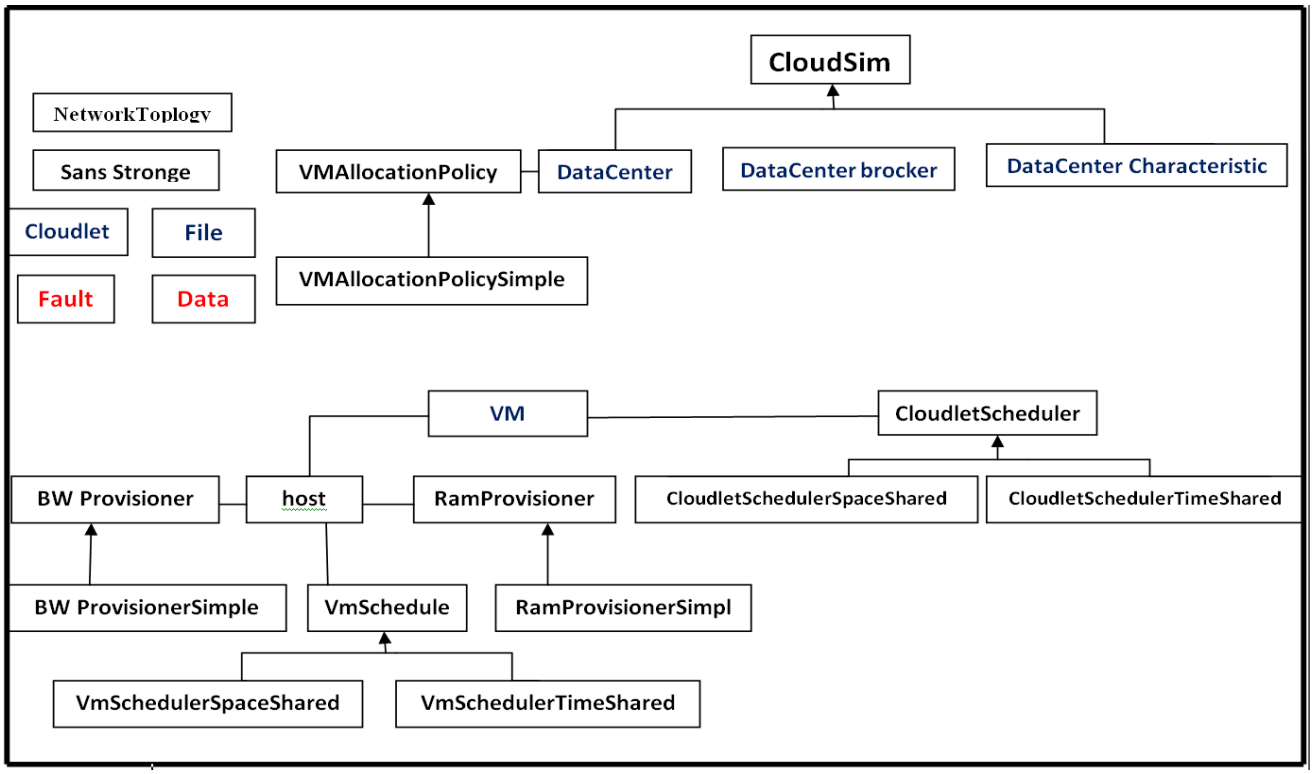

Figure.4.10: Diagramme de classe de la conception de simulateur CloudSim.

5. Conclusion

Dans ce chapitre, nous avons présenté la description et la conception de l'approche proposée ; l'approche est basée sur la réplication et les systèmes multi agents pour faire face aux fautes, Nous avons présenté l'algorithme proposé pour décrire le fonctionnement de notre mécanisme de tolérance aux fautes. Ainsi quelques diagrammes qui ont été formalisées à l'aide du langage UML pour décrire les différentes étapes de conception. Le prochain chapitre sera consacré à la partie implémentation et évaluation de notre proposition de tolérance aux fautes.

Chapitre V :
Implémentation

1. Introduction

Ce chapitre est consacré à la réalisation et la concrétisation de notre approche proposée, qui consiste à répliquer les données et tolérer les fautes dans les environnements de Cloud computing en utilisant les systèmes multi agents.

Dans un premier temps, nous présentons l'environnement de notre travail, puis nous définissons les différents services de simulateur CloudSim, et finalement nous présentons une série de simulation et leurs interprétations pour mettre en évidence notre proposition.

2. Langage de programmation Java :

Nous nous sommes orientés sur le langage JAVA qui présente néanmoins beaucoup d'avantages. Ce langage est apparu vers la fin de 1995 et obtient rapidement un énorme succès. Il s'agit d'un langage de conception très performant qui a été adopté par la majorité des fournisseurs. Ses caractéristiques intégrées de sécurité offrent un sentiment de confiance aux programmeurs comme aux utilisateurs des applications. De plus, Java incorpore des fonctionnalités qui facilitent grandement certaines tâches de programmation avancées comme la gestion des réseaux, la connectivité des bases de données ou le développement d'applications multitâches, etc.

3. Environnements de développement :

Eclipse est un environnement de développement intégré, libre, extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java, grâce à des bibliothèques spécifiques, ce langage est également utilisé pour écrire des extensions. La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de Plug-in (en conformité avec la norme OSGi), toutes les fonctionnalités de cet atelier logiciel sont développées en tant que Plug-in. Plusieurs logiciels commerciaux sont basés sur ce logiciel libre.

4. CloudSim

L'objectif principal de simulateur CloudSim est de fournir un cadre de simulation généralisée et extensible qui permet la modélisation, la simulation et l'expérimentation des nouvelles infrastructures du Cloud Computing et les services d'application, permettant aux utilisateurs

de se concentrer sur des questions de conception du système qu'ils veulent étudier, sans être préoccupé aux détails relatifs aux services et infrastructures Cloud.

4.1 Architecture générale de CloudSim :

La Figure 5.1 montre une architecture multicouche de la structure logicielle CloudSim et ses composantes.

La couche CloudSim fournit un support pour la modélisation et la simulation des Data Centres dans l'environnement des Cloudcomputing y compris les interfaces de gestion dédiées aux machines virtuelles, la mémoire, le stockage et la bande passante. L'affectation des VMs à des hôtes, la gestion d'exécution des applications et le suivi de l'état du système dynamique sont traités par cette couche. Un fournisseur Cloud doit implémenter ses stratégies dans cette couche afin d'étudier l'efficacité des différentes politiques qui permettent l'attribution des VMs à ses hôtes.

Au niveau supérieur de la couche CloudSim, nous avons le code utilisateur (user code) fournissant les entités de base pour les hôtes (nombre de machines, leur spécification et ainsi de suite), les applications (nombre de tâches et leurs exigences), VMs, nombre d'utilisateurs,types d'application et les politiques d'ordonnancement du Broker. Un développeur d'applications Cloud peut générer plusieurs activités parmi lesquelles nous citons : Des Scénarios de disponibilité des modèles Cloud, effectuer des tests robustes basés sur les configurations personnalisées supportées par le CloudSim et implémenter des techniques de provisionnement des applications personnalisées dans les Cloud.

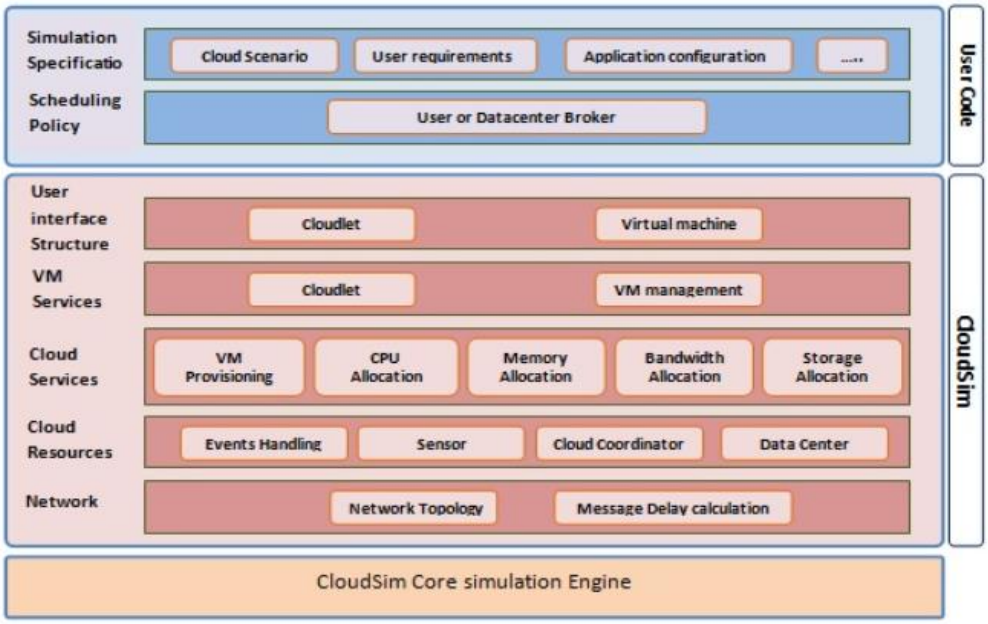

Figure 5.1 : Architecture générale de CloudSim.

4.2 Les classes de CloudSim :

Le simulateur CloudSim est composé de plusieurs classes que nous pouvons les classer en deux catégories : des classes qui modélisent les entités comme le Data Center, le Broker, etc. et des classes modélisant les politiques d'allocation

- **Datacenter :** cette classe modélise les services au niveau des infrastructures de base (matériel) qui sont offerts par les fournisseurs de Cloud (Amazon, Azure, App Engine). Elle encapsule un ensemble d'hôtes qui peuvent être homogènes ou hétérogènes par rapport à leurs configurations matérielles (mémoire, noyaux, capacité et stockage).
- **Datacenter Broker :** cette classe modélise le broker, qui est responsable de la médiation des négociations entre les utilisateurs et les prestataires de services selon les conditions de QoS des utilisateurs. Il déploie aussi les taches de service à travers les Clouds. Le Broker, au nom des utilisateurs, agit sur les prestataires du service approprié du Cloud par le service d'information du Cloud CIS (Cloud Information Services) et négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs.
- **VM :** cette classe représente une instance de machines virtuelle (VM) qui est gérée et hébergée par une machine physique (hôte). Chaque composant VM a accès à un composant qui stocke les caractéristiques suivantes liées à une VM telles que : mémoire accessible, le processeur, capacité de stockage et les politiques de provisionnement internes de la machine virtuelle. Dans le but de réduire l'énergie consommée par les data centres.
- **Cloudlet :** cette classe modélise les services d'application du Cloud (comme la livraison, réseaux sociaux et le workflow d'affaires). CloudSim représente la complexité d'une application en fonction de ses besoins informatiques.
- Chaque service d'application à une taille d'instruction pré-assigne et la quantité de flux de transfert de données qu'il doit entreprendre au cours de son cycle de vie. Cette classe peut également être étendue pour supporter la modélisation de la performance et d'autres paramètres de composition pour les applications telles que les transactions dans les applications orientées basent de données.
- **Injection des fautes :** Cette classe permet de modéliser l'inter-arrivée des fautes afin de créer les scenarios nécessaires pour évaluer le bon fonctionnement de nos approches de tolérance aux fautes.

5. Outil d'observation et de visualisation des SMA : la plateforme JADE

Les premiers développements logiciels, qui devinrent la plate-forme JADE, furent lancés par Telecom Italie (anciennement CSELT) fin 1998, motivé par la nécessité de valider les premières spécifications FIPA. [John et al ,2007]

JADE est une plateforme logicielle qui fournit des fonctionnalités de couche middleware de base indépendante de l'application spécifique et qui simplifient la réalisation des applications distribuées qui exploitent l'abstraction de l'argent logiciel (Woolbridge et jennings ,1995) un mérite important de JADE et qu'il implémente cette abstraction sur un langage orienté Object bien connu, java, fournissant une API simple te conviviale, les choix de conception simples ont été influencé par abstraction d'argent.[John et al ,2007]

JADE comparent trois composantes de base : un environnement d'exécution dans lequel les agents peuvent (vivre) : une bibliothèque de classes que les programmeurs peuvent utilisé pour développer leurs agents ;est une suite d'outils graphique permettant la gestion et l'administration des agents actifs, de plus, JADE peut être distribuée sur plusieurs hôtes et est composée de trois agents de base : AMIS(Agent Management Système) qui gère le cycle de vie des agents, maintient une liste une liste de tous les agents et contrôle l'accès et l'utilisation du canal de communication entre agents ;DF (Directory facilitateur) qui enregistre les descriptions des agents ainsi que les services qu'ils offrent, et ACC(Agent Communication Channel) qui gère les communications entre agents . [Bonjean,2009]

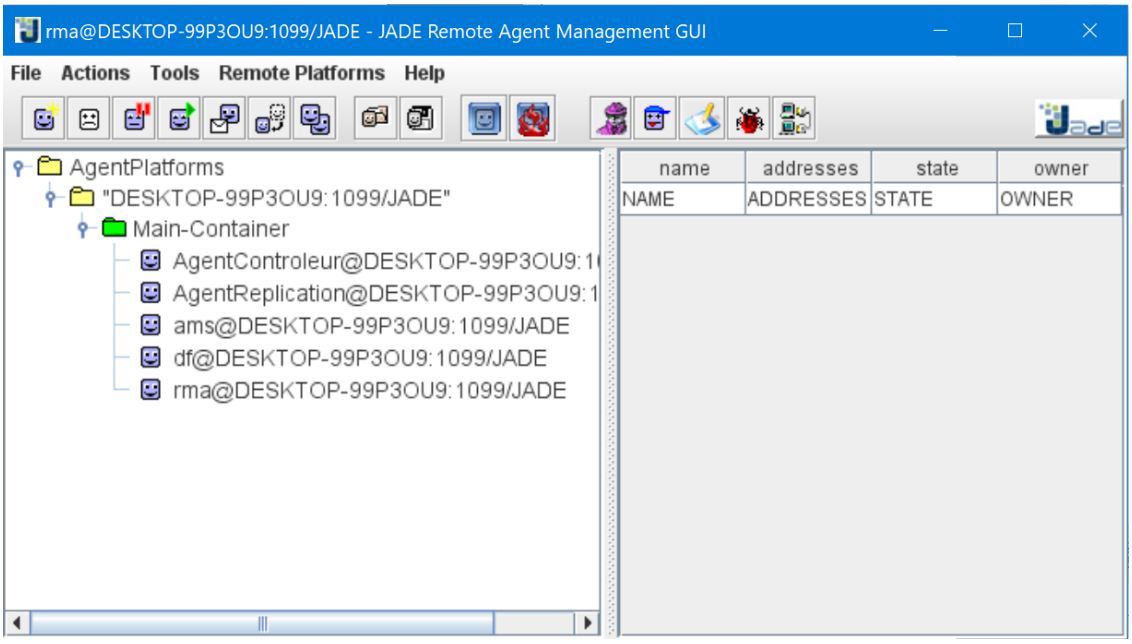

Figure 5.2 : Environnement de développement d'agent Java.

Notre système SMA est Composé de deux agents (l'agent contrôleur et l'agent réplication) et comme ces classes héritent de la classe Agent de bibliothèque (jade.jar), on doit configurer chaque agent puis ajouter une configuration de coordination entre les deux agents comme il est décrit ci-dessous (voir Figure 5.3).

Pour compiler et lancer l'agent il faut aller à Run→Run configuration puis java application. Dans l'onglet « main » et dans la zone de saisie Main class, il faut taper le code suivant : jade.Boot.

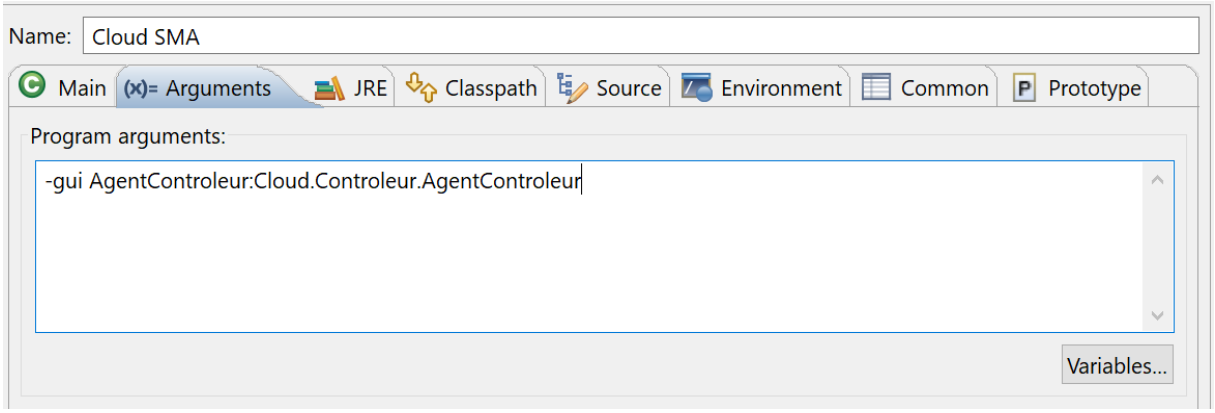

Figure 5.3 : Configuration d'exécution de l'agent contrôleur.

Dans l'onglet argument : on doit taper le code suivant :

-Gui NomDeL'agent :LeNomDuPackage.LeNomDeLaClasse

- Gui veut dire graphique user interface

La même chose pour l'agent réplication (le deuxième agent)

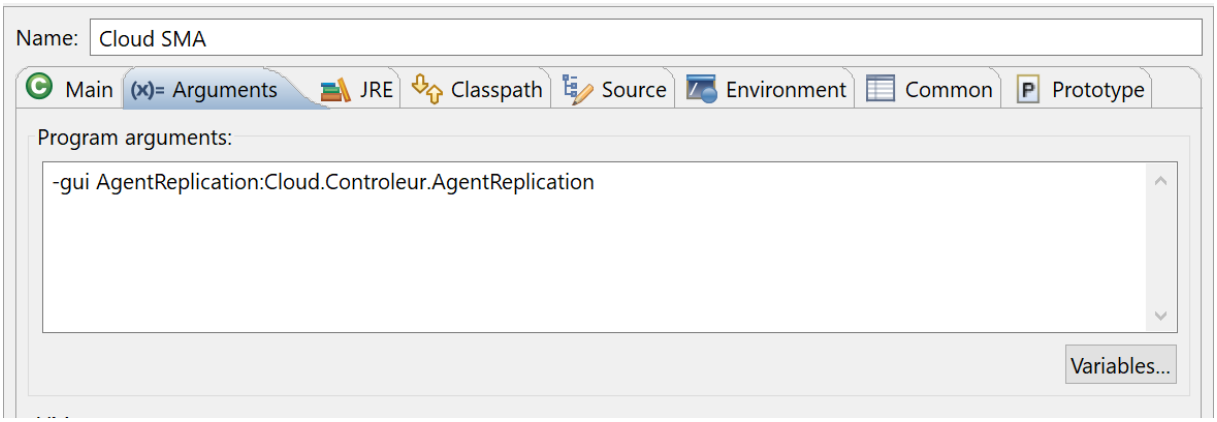

Figure 5.4 : Configuration d'exécution de l'agent réplication.

Ensuite, la configuration d'exécution pour les deux agents aux mêmes temps se fait comme suit : (voir Figure 5.5).

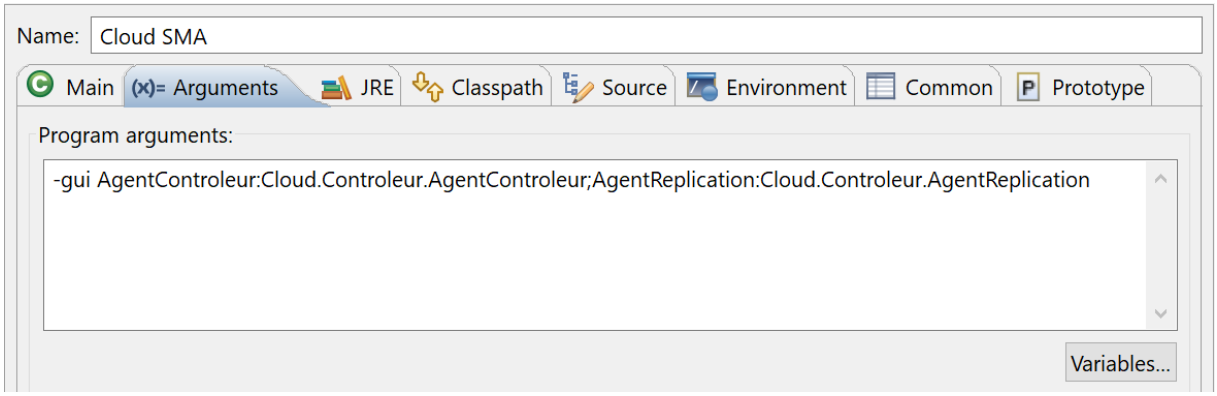

Figure 5.5 : Configuration d'exécution de deux agents.

Puis, on clique sur ***apply*** et enfin on clique sur ***run*** pour voir le résultat.

6. Interface principale

Cloud Sim n'a pas une interface graphique, son exécution se fait sur la console, donc nous avons créé une interface qui facilite l'accès à la configuration du simulateur.

6.1 Configuration des paramètres de simulation

Pour lancer la simulation d'un Cloud avec notre version étendue du simulateur CloudSim, nous devons configurer les paramètres de simulation dans un premier temps. Les Figures 5.6 au-dessous montrent deux fenêtres de configuration de simulation qui contient des parties principales :

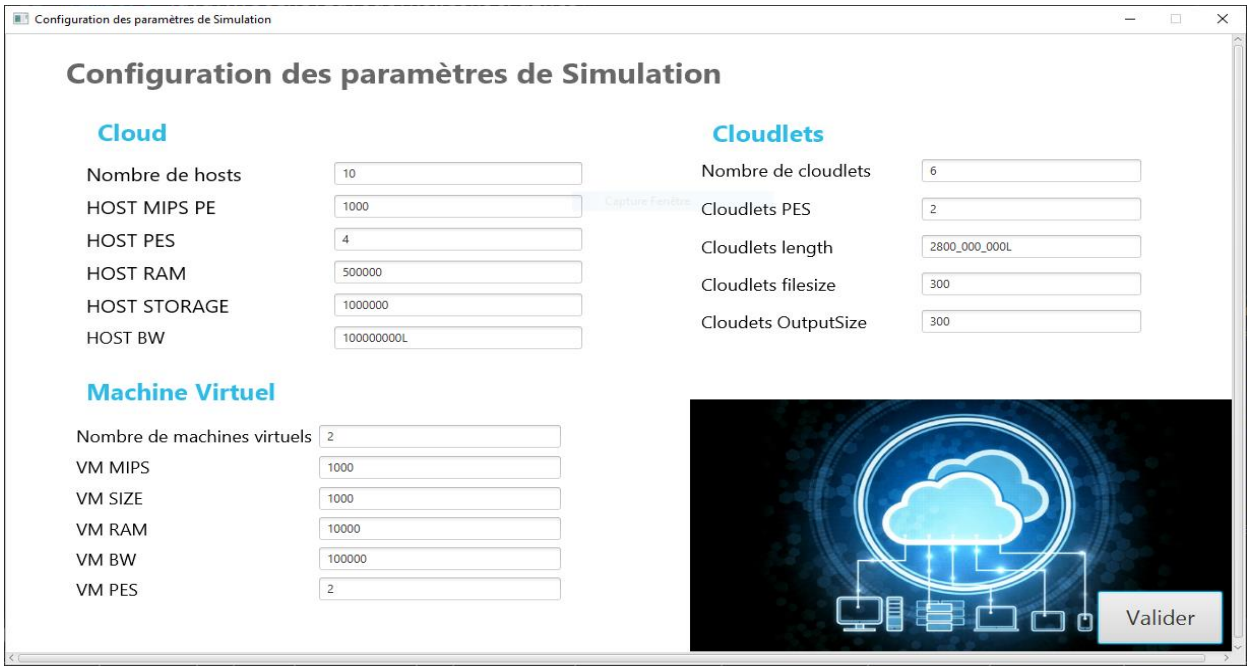

Figure 5.6 : Configuration des paramètres de simulation

6.1.1 Datacenter :

Cette étape consiste à faire entrer les paramètres nécessaires propres à la topologie du réseau comme : le nombre de hosts, host bw... (Voir Figure5.7)

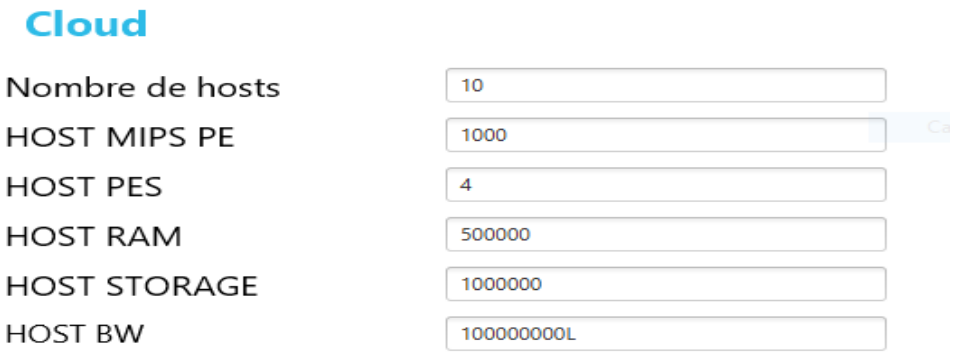

Figure 5.7 : Configuration de Datacenter

a) Code de création de Host

```
public Host createHost() {
    final List<Pe> pesList = createPeList(HOST_PES, HOST_MIPS_BY_PE);
    final ResourceProvisioner ramProvisioner = new ResourceProvisionerSimple();
    final ResourceProvisioner bwProvisioner = new ResourceProvisionerSimple();
    final VmScheduler vmScheduler = new VmSchedulerTimeShared();
    return new HostSimple(HOST_RAM, HOST_BW, HOST_STORAGE, pesList)
            .setRamProvisioner(ramProvisioner)
            .setBwProvisioner(bwProvisioner)
            .setVmScheduler(vmScheduler);
}
```

b) Code de création de Datacenter

```
private Datacenter createDatacenter() {
    hostList = new ArrayList<>();
    for (int i = 0; i < HOSTS; i++) {
        hostList.add(createHost());
    }
    System.out.println();

    Datacenter dc = new DatacenterSimple(simulation, hostList, new VmAllocationPolicySimple());
    dc.setSchedulingInterval(SCHEDULE_TIME_TO_PROCESS_DATACENTER_EVENTS);
    return dc;
}
```

c) Virtual Machines :

Ce code permet de configurer les machines virtuelles et leurs caractéristiques qui sont : le nombre de CPU dans une VM, la taille de la mémoire, la taille du disque dur et la bande passante. (Voir Figure5.8)

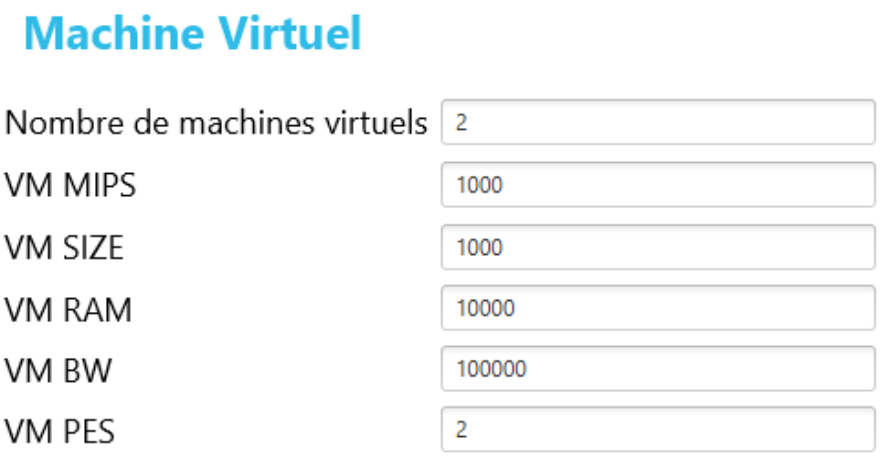

Figure 5.8 : Configuration de Machine virtuelle

d) Code de création d'une machine virtuelle

```
public Vm createVm() {
    Vm vm = new VmSimple(vmList.size()+1, VM_MIPS, VM_PES);
    vm
        .setRam(VM_RAM).setBw(VM_BW).setSize(VM_SIZE)
        .setCloudletScheduler(new CloudletSchedulerTimeShared());
    return vm;
}
```

e) Cloudlet :

Permet de configurer les propriétés de la Cloudltet et leurs caractéristiques qui sont : Nombre de Cloudlets, la taille de la Cloudlet… (voir Figure5.9)

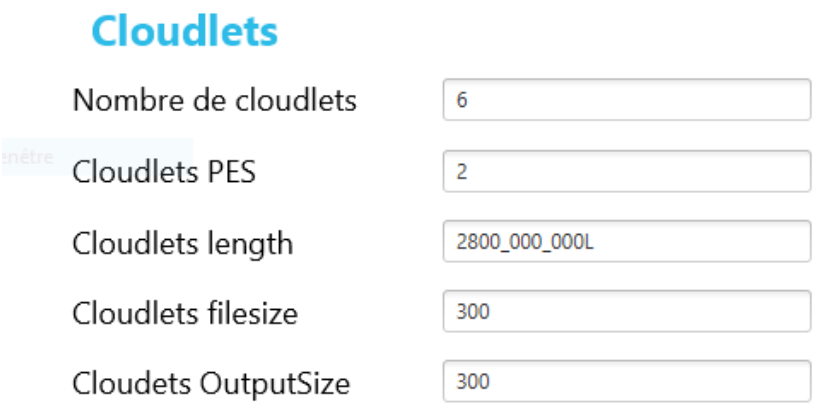

Figure 5.9 : Configuration de Cloudlets

f) Code de création d'une Cloudlet

```
public void createAndSubmitCloudlets() {
    UtilizationModel utilizationModelDynamic = new UtilizationModelDynamic(0.1);
    UtilizationModel utilizationModelFull = new UtilizationModelFull();
    for (int i = 0; i < CLOUDLETS; i++) {
        Cloudlet c
            = new CloudletSimple(cloudletList.size()+1, CLOUDLET_LENGHT, CLOUDLET_PES)
                    .setFileSize(CLOUDLET_FILESIZE)
                    .setOutputSize(CLOUDLET_OUTPUTSIZE)
                    .setUtilizationModelCpu(utilizationModelFull)
                    .setUtilizationModelBw(utilizationModelDynamic)
                    .setUtilizationModelBw(utilizationModelDynamic);
        cloudletList.add(c);
    }

    broker.submitCloudletList(cloudletList);
}
```

g) Tolérance aux fautes

Cette fenêtre permet à l'utilisateur de sélectionner le mécanisme de tolérance aux fautes. Sans Réplication représente l'exécution sans aucune approche de tolérance aux fautes. Réplication est l'approche de tolérance aux fautes basée sur la réplication de données et les systèmes multi agent et la réplication avec l'injection des fautes (détection et correction) (voir Figure5.10)

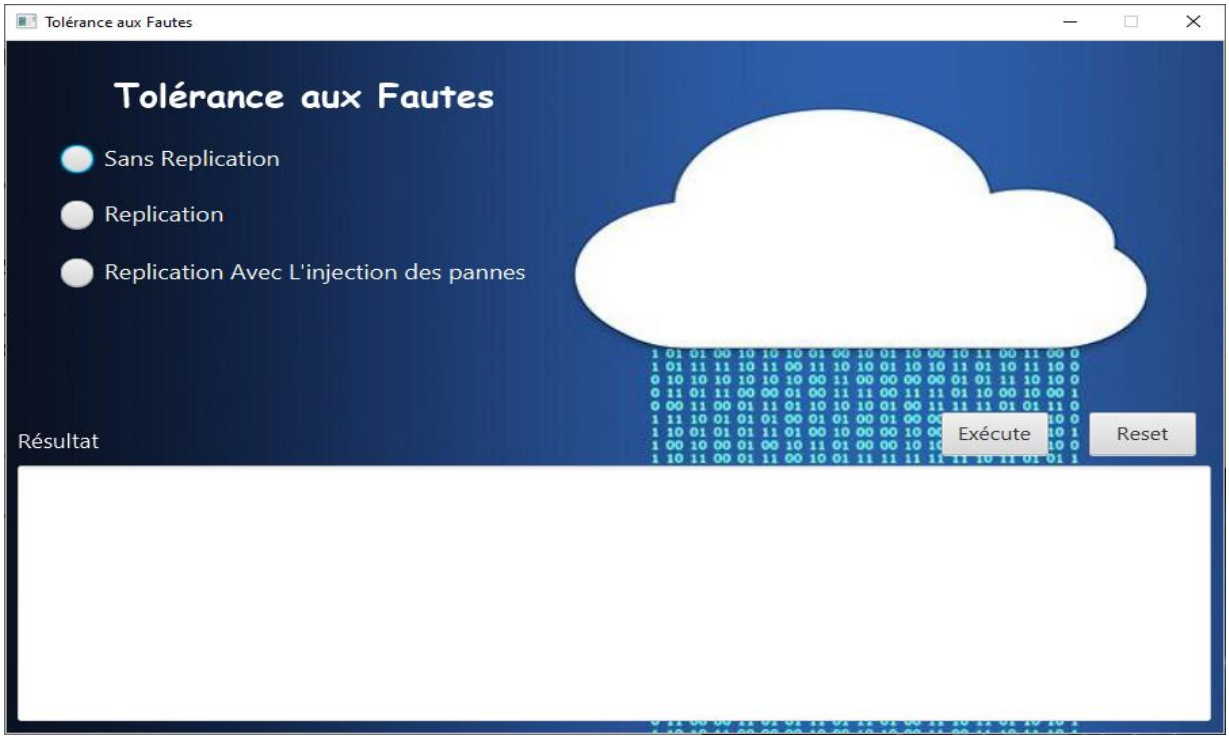

Figure 5.10 : Tolérance aux fautes

h) Code de création de Réplique pour les VM

```
private Vm cloneVm(Vm vm) {
    Vm clone = new VmSimple(vm.getMips(), (int) vm.getNumberOfPes());

    clone.setId(vm.getId() * 10);
    clone.setDescription("Clone of VM " + vm.getId());
    clone
        .setSize(vm.getStorage().getCapacity())
        .setBw(vm.getBw().getCapacity())
        .setRam(vm.getBw().getCapacity())
        .setCloudletScheduler(new CloudletSchedulerTimeShared());
    System.out.printf("%n%n# Cloning %s - MIPS %.2f Number of Pes: %d%n", vm, clone.getMips(), clone.getNumberOfPes());

    return clone;
}
```

i) Code de création de Réplique pour les cloudlet des Machines Virtuelles

```
private List<Cloudlet> cloneCloudlets(Vm sourceVm) {
    final List<Cloudlet> sourceVmCloudlets = sourceVm.getCloudletScheduler().getCloudletList();
    final List<Cloudlet> clonedCloudlets = new ArrayList<>(sourceVmCloudlets.size());
    for (Cloudlet cl : sourceVmCloudlets) {
        Cloudlet clone = cloneCloudlet(cl);
        clonedCloudlets.add(clone);
        System.out.printf("# Created Cloudlet Clone for %s (Cloned Cloudlet Id: %d)%n", sourceVm, clone.getId());
    }

    return clonedCloudlets;
}
```

7. Lancement de la simulation

Après avoir personnalisé les paramètres de simulation, l'utilisateur peut lancer la simulation en cliquant sur le bouton Valider pour valider les paramètres de configuration ensuite l'utilisateur doit choisir le mécanisme de tolérance aux fautes ensuite sur le bouton exécute. (Voir Figure 5.11) .

Figure 5 : 11 : Lancement de la simulation

8. Résultats

- ✓ **Approche sans réplication :** Représente la première stratégie dans laquelle aucune stratégie de réplication n'est implémentée...
- ✓ **Approche basée sur la réplication seulement (Réplication) :** Représente notre stratégie dans laquelle la réplication des données est effectuée mais sans gestion des fautes.
- ✓ **Approche basée sur la réplication avec la gestion des fautes :** c'est une amélioration de notre approche dans laquelle la réplication des données est effectuée et la gestion des fautes est considérée.

Nous avons effectués plusieurs séries de simulation. Dans cette partie,nous présentons un exemple illustrant unrésultat parmiplusieurs expérimentationsque nous avons obtenus.

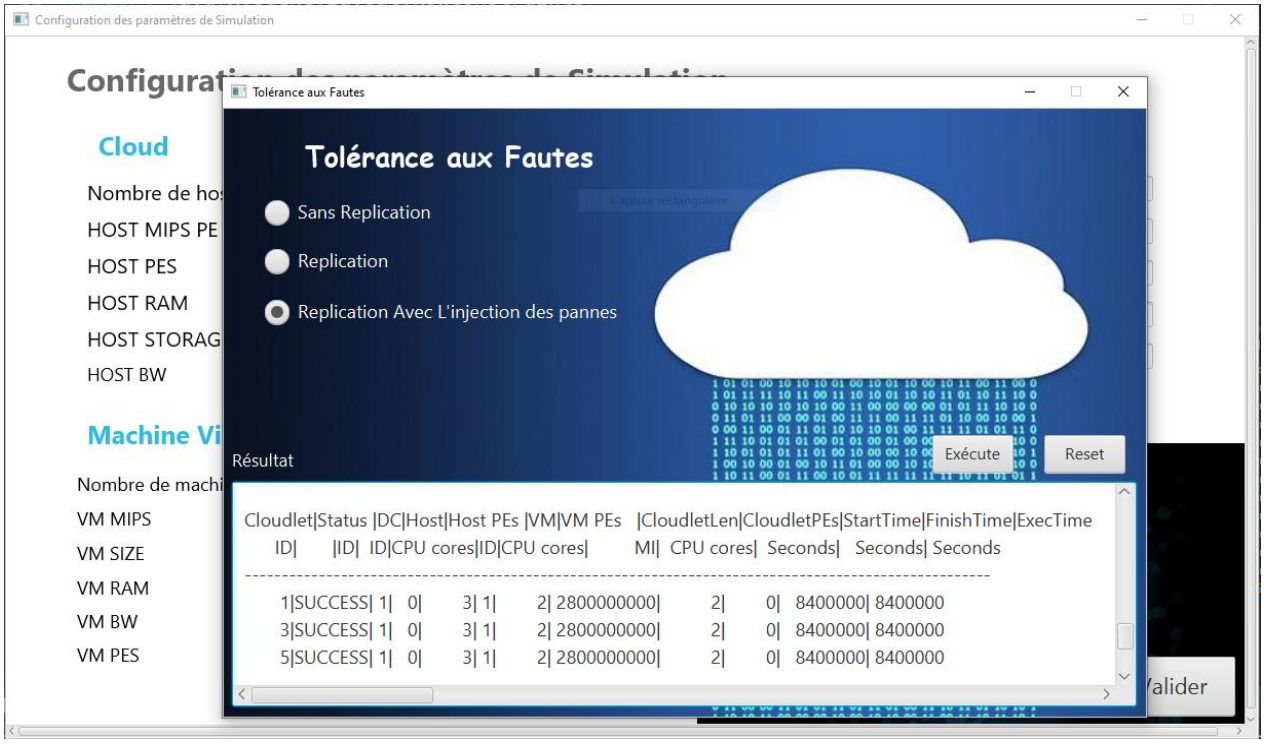

Figure 5.12 : Résultat de simulation sur l'interface

```
[0;39m[34mINFO  CloudInformationService0: Notify all CloudSim Plus entities to shutdown.

[0;39m[34mINFO  8983560,04: DatacenterSimple1 is shutting down...
[0;39m[34mINFO
================== Simulation finished at time 8983560,04 ==================

[0;39m[39mDEBUG DeferredQueue >> max size: 5 added to middle: 0 added to tail: 59
[0;39m

                                    SIMULATION RESULTS

Cloudlet|Status |DC|Host|Host PEs |VM|VM PEs   |CloudletLen|CloudletPEs|StartTime|FinishTime|ExecTime
      ID|       |ID|  ID|CPU cores|ID|CPU cores|         MI|  CPU cores|  Seconds|   Seconds| Seconds
-------------------------------------------------------------------------------------------------------
       1|SUCCESS| 1|   0|        3| 1|        2| 2800000000|          2|        0|   8400000| 8400000
       3|SUCCESS| 1|   0|        3| 1|        2| 2800000000|          2|        0|   8400000| 8400000
       5|SUCCESS| 1|   0|        3| 1|        2| 2800000000|          2|        0|   8400000| 8400000
-------------------------------------------------------------------------------------------------------

# Mean Number of Failures per Hour: 0,010 (1 failure expected at each 100,00 hours).
# Number of Host faults: 28
# Number of VM faults (VMs destroyed): 0
# Time the simulations finished: 2495,4333 hours
# Mean Time To Repair Failures of VMs in minutes (MTTR): 0,00 minute
# Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0,00 minutes
# Hosts MTBF: 5201,00 minutes
# Availability: 100,00%

Host_Fault_Injection finished!
```

Figure 5.13 : Résultat de simulation

9. Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre application. Nous avons réalisé plusieurs simulations en jouant sur différents paramètres comme : le nombre de cloudlets, le nombre de fautes. Nous avons utilisé pour cela plusieurs outils tels que le CloudSim et l'Eclipse et la plateforme Jade. Cette plateforme permet de simplifier le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA. Les résultats obtenus montrent que l'utilisation de notre méthode permet d'améliorer la performance du système, elle minimise le pourcentage d'erreur et réduit le temps perdu causé par les fautes.

CONCLUSION GENERALE ET PERSPECTIVES

Le Cloud computing est une technique de calcul et de stockage naissante qui se consolide rapidement comme une grande étape dans le développement et le déploiement d'un nombre croissant des applications réparties. L'ordonnancement de tâches et d'allocation de ressources dans les systèmes de type Cloud computing suscite une attention croissante avec l'augmentation de la popularité de Cloud.

Comme tous services informatiques, les services Cloud sont vulnérables aux défaillances. En plus, dans des tels systèmes le taux de fautes croit avec la taille du système lui-même. Ce qui impose l'utilisation d'un mécanisme de tolérance aux fautes pour assurer la fiabilité et la qualité des services. La tolérance aux fautes est un domaine qui a été largement étudié. Dans la littérature, il existe plusieurs mécanismes et manières d'envisager une faute dans un système réparti à grande échelle telles que le Cloud computing. On peut diviser les mécanismes de tolérance aux fautes en trois catégories : les protocoles par points de reprise , les protocoles par journalisation et La tolérance aux fautes par duplication, chacune de ces catégories présente des propriétés différentes en termes de performance, et n'est parfois pas applicable selon le système .

Les SMA offrent de nombreux avantages pour leur conception et leur développement. Ils consistent à distribuer les traitements complexes sur des agents de moindre complexité. Le problème consiste à trouver une bonne décomposition des traitements, puis de mettre en œuvre une bonne coordination qui puisse assurer la coopération des agents.

Nous avons dans ce mémoire exprimé nos recherches sur le sujet, Pour cela, Dans un premier temp, nous avons réalisé une présentation générale du domaine des systèmes multi-agents. Nous l'abordons d'abord sous l'aspect « agent » en présentant les caractéristiques relatives à cette unité de base du . Puis, nous passons à la dimension collective du système. Dans une deuxième on a présente un état de l'art sur le Cloud computing. on donne une introduction sur le Cloud computing, y compris la définition, ses caractéristiques, les types des services Cloud, les modèles de déploiement ainsi que la sécurité, les avantages et les inconvénients

du Cloud et nous terminerons par une conclusion puis la troisième partie décrit les différents concepts et les grandes lignes liés à la tolérance aux fautes ainsi que les diverses techniques de gestion des fautes utilisées. Pour la partie pratique on a commencé par introduisent, tout d'abord, la problématique et les objectifs visés par cette étude. Après avoir survolé quelques travaux existants. Ensuite, nous avons présenté notre méthode. Enfin dans une dernière partie, nous avons implémenté cette dernière.

L'objective de ce mémoire est de proposer une méthode hybride de prédiction des fautes dans le Cloud computing basé sur les systèmes multi-agents et la technique de réplication, pour valider notre travail, nous avons implémenté des Clouds et des services virtuels en utilisant le simulateur Cloud Sim, et on a intégré des agents a travers JADE (JAVA agent développement frame work),
Les résultats montrent que l'approche proposée présente de bonnes performances en présence des fautes, elle permet d'améliorer le temps de réponse et minimiser les surcoûts causés par les fautes.

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

Bibliographies

A. Lee-Post et P. Ram, «Cloud computing: A comprehensive introduction,» *In Security, Trust, and Regulatory Aspects of Cloud Computing in Business Environments, IGI Global*, pp. 1-23, 2014.

Ali KALAKECH. Etalonnage de la sureté de fonctionnement des systèmes ´ d'exploitation { spécifications et mise en œuvre, 2005}.

Brazier F.M.T., van Steen M., Wijngaards N.J.E., « Distributed Shared Agent Representations », In : Marik V., Stepankova O., Krautwurmova H. and Luck M.,
P. Mell et T. Grance, «The NIST Definition of Cloud Computing.,» NIST, [Enligne]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.. [Accès le 07 octobre 2018].

cloud/cloud-kesako, (Consulté Mars 2016). (Cité en pages viii et 17.)

C. N. Höfer et G. Karagiannis, «Cloud computing services : taxonomy and comparison,» *Journal of Internet Services and Applications*, vol. 2, n° %12, pp. 81-94, 2011.

Cuckuern M. et al., « AQuA : An Adaptive Architecture That Provides Dependable Distributed Objects », *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS'98)*, p. 245-253, West Lafayette, Indiana, October 20-23, 1998.

Developing Multi-Agent System with JADE Fabio luigui Bellifemine, Giovanni Caire, Dominic Greenwood, 2007 John Wiley & Sons, Ltd.

Decker K. S., Sycara K., « Intelligent Adaptive Information Agents », *In Journal of Intelligent Information Systems*, vol. 9, p. 239 - 260, 1997.

Favarim F., Siqueira F., Fraga J.S., « Adaptive Fault-Tolerant CORBA Components », *Middleware Workshops 2003*. p. 144-148.

Fedoruk A., Deters R., « Improving fault-tolerance by replicating agents », *Proceedings AAMAS-02*, Bologna, Italy, p. 737-744.

Ferber.J , Les systèmes multi-agents, vers une intelligence collective, InterEditions, 1995.

G. IT, «Gartner IT glossary,» Technology Research, 2013.

Gleizes. MP, Bernon. C, Migeon. F, & Picard. G, (2008), « Méthodes de développement de systèmes multi-agents », Génie logiciel N° 86.

Guillaume Sigui. Cloud computing, quels sont les risques de s´ecurit´e majeurs
du cloud computing ? http ://www.developpez.com/, (Consult´e Mars 2014).(Cit´e en pages ix, 24 et 79.)

Haouati M. S. *Architectures innovantes de systèmes de commandes de vol*. PhD thesis, Universit2 de Toulouse, 2010.

H. D, «Cloud computing : a taxonomy of platform and infrastructure-level offerings,» chez *Tech Rep GIT-CERCS-09-13,CERCS*, Georgia Institute of Technology., 2009.

JC Laprie. Concepts de base de la tol2rance aux fautes, journée de. l'ofta, informatique tolérante aux fautes. Paris, 1994.

Jean-Louis Caire and Willy Munch. *Objectif Cloud : Une démarche pratique orientée services*. EniDatapro, 2014. (Cité en pages 18 et 19.)

Jian J.Z , Chengdu.C et Nan.Z 'Cloud Computing-based Data Storage and Disaster Recovery', Proceediongs of the International Conference on Future computer science and education, Xi'an, China, pp. 629-632,2011.

J. I. et C.-D. B., Aperçu sur les systèmes Multi-Agents, CIRANO: Série Scientifique du centre inter universitaire de recherche en analyse des organisations, 2002.

Joseph F Ruscio, Michael A Heffner, and Srinidhi Varadarajan. Dejavu : Transparent user-level checkpointing, migration, and recovery for distributed systems. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1{10. IEEE, 2007.

J Vial. *Test et Testabilité de Structures Numériques Tolérantes aux Fautes*. PhD thesis, Université de Montpellier, 2009.

Kraus S., Subrahmanian V.S., Cihan N., « Probabilistically Survivable MASs », *In Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*.

Kalbarczyk Z., Bagchi S., Whisnant K., Iyer R.K., « Chameleon : A Software Infrastructure for Adaptive Fault Tolerance », *IEEE Transactions on Parallel and Distributed Systems*, June 1999.

K. Sycara, «Multi-agent system,» *AI Magazine,* vol. 19, n° %12, p. 79–92, 1998.

Kouidri Siham. gestion de la cohérence des répliques tolérante aux fautes dans une grille des données. Master'sthesis, université d'Oran, 2011.
.

LIMAM Said. *Gestion de tolérance aux fautes dans le Cloud Computing avec CloudSim*. PhD thesis, Universit´e Ahmed Ben Bella d'Oran1 Es Senia, 2012.

M. Abdelhak, Approche de composition de services web dans le Cloud Computing basée sur la coopération des agents, Université Mohamed Khider – BISKRA, Département d'informatique, 20/02/2018.

Manetho Elmootazbellah. Elnozahy. *fault tolerance in distributed systems using rollback-recovery and process replication*. PhD thesis, Rice University, Houston, TX, USA, 1993.

Mathur.P , 'Cloud Computing: New challenge to the entire computer industry',Proceediongs of the 1st International Conference on Parallel, Distributed and Grid Computing, Solan, India, pp. 223-228,2010.

MultiAgent-Systems and Applications II, Lecture Notes in Computer Science, Vol. 2322, p.213-220, 2002.

M Wiesmann,F. Pedonne and A .Schipper, A Systematic Classification of Replited Database Protocols based on Atomic Broadcast, In Proceedings of the 3th European Research Seminar on Advances in Distributed Systems(ERSADS99), Madeira Island()Portugal, April 23- 28,1999.BROADCAST Esprit WG22455.

N.Bonjean ; « Ingénierie des systèmes Multi-Agents adaptatifs : vers un guide pour la conception du comportement d'agent coopérative » université Paul Sabatier (UPS) – Toulouse III ; Tlemcen ,2009

N. R. Jennings, K. Sycara et M. Wooldridge, «A Roadmap of Agent Research and Development,» *Journal of Autonomous Agents and Multi-Agent Systems. Boston, USA: Kluwer Academic Publishers.*, 1998.

N. R. Jennings, «On agent-based software engineering,» *Artificial Intelligence Journal*, 2000.

O. Peres. *Construction de topologies auto stabilisante dans les systèmes à grande échelle*. PhD thesis, Université Paris-sud, France, 2008.

o. Khayati, «Modèles formels et outils génériques pour la gestion et la recherche de composants,» chez *Thèse, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE*, 2005

Peter.M and Timothy.G. NIST : The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145. pdf.

P. Mell et T. Grance, «The NIST Definition of Cloud Computing.,» NIST, [Enligne]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.. [Accès le 07 octobre 2018].

P. Guillaume et S. Corina, «Conpaas : A platform for hosting elastic cloud applications,» *IEEE Internet Computing*, p. 88–92, 2012.

Proceedings of the Second Australian Workshop on Distributed AI, Cairns, Australia, August 27, 1996.

S. Drapeau. RS2.7 : Un Canevas Adaptable de Services de Duplication. PhD thesis, Institut National Polytechnique de Grenoble, France, Juin 2003.

S. Srinivasan, «Cloud computing basics,» Springer, 2014, p. 44.
Le Cloud Kesako. Cloud-serveur. http ://www.cloud-serveur.fr/fr/le-

Sylvain Caicoya and Jean-Georges Saury. *CLOUD COMPUTING : Maitrisez les enjeux et solutions de l'informatique dans les nuages*. Micro Application, 2011. (Cité en pages 16, 20 et 21.)

Vercouter. L, (2000), « Conception et mise en oeuvre de systèmes multi-agents ouverts et distribués », Thèse de doctorat, Université de Jean Monnet et Ecole des Mines de Saint-Etienne,France.

W. B. Ling Z. D. Cheng H. Hui, Z. Zhan and Y. Xiao Zong. A two-level application transparent checkpointing scheme in cloud computing environment. 2013.

Y. Crouzet J. Arlat and Y. Deswarte. Encyclopédie de l'informatique et des systèmes d'information. 2006.