RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :.....

T H È S E

Présentée en vue de l'obtention du diplôme de
Doctorat LMD en Informatique

Option : Intelligence artificielle

Titre

Approche de composition de services web dans le Cloud Computing basée sur la coopération des agents

Présentée par
Abdelhak Merizig

Soutenu le : 20/02/2018

Devant le jury composé de :

Président :	**Pr.** Mokhtar Sellami	Université d'Annaba
Rapporteur :	**Pr.** Okba Kazar	Université de Biskra
Examinateurs :	**Pr.** Rachida Benabdelaziz	Université de Biskra
	Dr. Maite Lopez Sanchez	Université de Barcelone, Espagne
	Dr. Sadek Labib Terrissa	Université de Biskra
Invité :	**Dr.** Hamza Saouli	Université de Biskra

Année universitaire : **2017 – 2018**

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Department

Order number :.....

THESIS

Submitted in fulfilment of the requirements for the degree Doctor (3^{rd} Cycle)
in Computer Science

Option : Artificial Intelligence

Title

Web service composition approach in Cloud Computing based on agent's cooperation

---

Presented by
Abdelhak Merizig

Defended in : 20/02/2018

Defended before a jury composed of :

President :	**Pr.** Mokhtar Sellami	University of Annaba
Supervisor :	**Pr.** Okba Kazar	University of Biskra
Members :	**Pr.** Rachida Benabdelaziz	University of Biskra
	Dr. Maite Lopez Sanchez	University of Barcelona - Spain
	Dr. Sadek Labib Terrissa	University of Biskra
Invited :	**Dr.** Hamza Saouli	University of Biskra

Remerciements

En premier lieu, je remercie le bon Dieu de m'avoir donné la force et la patience nécessaire pour achever ce travail de thèse.

Je tiens à remercier mon directeur de thèse monsieur **Pr. Kazar Okba** de m'avoir accueilli au sein de son équipe pour réaliser ma thèse ainsi que pour sa disponibilité et son soutien.

Je remercie également **Dr. Maite Lopez Sanchez** d'avoir suivre mon travail et aussi pour leurs précieux conseils.

Je veux également remercier tous les membres de jury :

Pr. Sellami Mokhtar , Professeur à l'Université de Annaba, pour m'avoir fait l'honneur de présider mon jury de thèse, ainsi que **Pr. Benabdelaziz Rachida** , Professeur à l'Université de Biskra, **Dr. Maite Lopez Sanchez** , Maître de conférences à Université de Barcelone - Espagne, **Dr. Terrissa Sadek Labib** , Maître de conférences à l'Université de Biskra, **Dr. Saouli Hamza** , Maître de conférences à l'Université de Biskra, pour m'avoir fait l'honneur d'accepter de juger ce travail.

Je tiens aussi à remercier toute ma promotion de doctorat et tous mes amis.

Enfin je tiens à remercier tous ceux et celles qui m'ont aidé et soutenu de près ou de loin pour l'accomplissement de cette thèse.

Abdelhak Merizig

Acknowledgements

Above all, thank Allah the most graceful and merciful for helping me finishing this modest work which I hope it has a good importance.

First and foremost I want to thank **Pr. Okba Kazar** my supervisor, for his support and his encouragement. He dedicated to me a lot of time and guided me during my Thesis. I felt so lucky to work with him.

Special thanks to **Dr. Maite Lopez Sanchez**, Thank you for all the guidance, patience, immense knowledge, and the support that you gave me.

My sincere thanks also to the committee members : **Pr. Mokhtar Sellami**, **Pr. Rachida Benabdelaziz**, **Dr. Maite Lopez Sanchez**, **Dr. Sadek Labib Terrissa**, and **Dr. Hamza Saouli**, for accepting to evaluate this work. I would like to take this opportunity to express my deepest gratitude to members of *WAI research group* at University of Barcelona for all the help and the support that gave me during my scholarship. Thank you !

Finally, I would like to thank all my family. In particular my parents, and my sisters. I really could not have done most of the things I have enjoyed doing. Thanks again.

Last but not least, all my friends that are really a lot. They always supported me during these years and gave me the right strength to live in happiness.

Abdelhak Merizig

Dedicated to my precious parents and sisters

ملخص

يعتبر في وقتنا الحالي تركيب الخدمات واحدة من المشاكل الرئيسية في الخدمات السحابية نظرا للقيم الاستثنائية لنمو الخدمات والتي نشرها منتجو الخدمات. كما أن الخدمات البسيطة لم تتمكن مؤخرا من تلبية جميع متطلبات الزبائن، حيث تكوين الخدمة التقليدي يوفر للعملاء خدمة مركبة فقط دون إسناد عامل جودة الخدمة. من أجل الرد على الإعدادات الوظيفية وغير الوظيفية، نحن بحاجة إلى تركيب مجموعة من الخدمات. وباعتبار أن خدمات الويب غير قادرة على التواصل فيما بينها أو المشاركة ديناميكيا في تركيب الخدمة، أدت هذه المشكلة إلى استخدام كيان ديناميكي يمثله وكيل الذي يستند إلى بنية ديناميكية.

يقترح هذا العمل تصميم يستند على الوكلاء عن طريق اضافة بروتوكول جديد للتعاون يمكن أن يوفر تكوين خدمة بطريقة أوتوماتيكية وقابل للتكيف من خلال توفير خدمة مركبة مع ضمان جودة الخدمة (QoS). ولأن مقدم الخدمة يستخدم السحابة لنشر خدماته، لضمان السير الحسن لهذه العملية ومن أجل تشغيل الخدمة في أفضل الظروف وفقا لقيم الجودة الخدمة (QoS). بغرض نشر الخدمات في السحابة، كل واحدة من هذه الخدمات تحتاج إلى نموذج موحد. عند الاشارة إلى مشكلة التجميع، يمكن اعتبار تركيب الخدمة حلا لهذه المشكلة. بالإضافة إلى ذلك، يبحث كل مزود خدمة عن أفضل نشر لها والذي يمنح الخدمات أفضل الظروف بفضل نوعية الخدمة التي يتميز بها. تحول هذه الأطروحة مشكلة نشر الخدمة إلى مشكلة التحسين، كما تقترح هذه المذكرة نموذجا لوصف الخدمة المساعدة لعملية البحث عن الخدمة.

الكلمات المفتاحية:

تكوين الخدمة، الخدمات البسيطة، خدمة ويب، وكيل، نشر الخدمة، الحوسبة السحابية، الخدمة السحابية، نظام متعدد الوكلاء، جودة الخدمة، مشكلة التحسين

Abstract

Nowadays, service composition is one of the major problems in the Cloud due to the exceptional growth number of service deployed by providers. Recently, atomic services cannot deal with all client requirements. Traditional service composition gives the clients a composite service without non-functional parameters. To respond to both functional and non-functional parameters, we need a service composition. Since, web services cannot communicate with each other or participate dynamically in service composition, this issue led us to use a dynamic entity represented by an agent. This work propose an agent-based architecture with a new cooperation protocol that can offer an automatic and adaptable service composition by giving a composite service with maximum number of quality of service (QoS). As the provider uses the cloud to deploy their services, which this process needs to be handled well in order to make the service operate in best conditions according to QoS values. In order to deploy services in the Cloud, each one needs some combination of services model. When we mentioned combination problems, service composition represents the solution of this issue. In addition, each service provider looks for the best service deployment that gives his service best conditions through the quality of service. The present thesis transforms the service deployment problem into an optimization problem. Moreover, in this thesis we proposed service description model that assist service discovery process.

Keywords : *Service Composition, Atomic Services, Web Service, Agent, Service Deployment, Cloud Computing, Cloud Service, Multi-Agent System, QoS, Optimization Problem.*

Résumé

De nos jours, la composition de services est l'un des problèmes majeurs dans l'environnement du *Cloud Computing* à cause de la croissance exceptionnelle du nombre de services déployés par les fournisseurs. Les méthodes traditionnelles qui sont basées sur les services atomiques ne peuvent pas répondre à toutes les exigences des clients. L'un des solutions pour résoudre ce problème est la composition des services. Tandis que la composition des services proposent une solution lorsqu'ils existent un nombre gérable des clients. Néanmoins, les services composites ne prennent pas en considération l'aspect non-fonctionnel. Donc, pour répondre aux besoins fonctionnels et non fonctionnels, nous avons besoin d'une composition de services efficaces. Pendant plusieurs décennies, de nombreux travaux ont utilisé la composition pour les services Web, dans lequel les services Web connu par l'absence de l'aspect de communication entre eux et le manque de participation de manière dynamique. La raison principale de ces problèmes est le manque d'un langage unifié, donc pour obtenir ce langage, nous avons besoin d'une entité avec une propriété dynamique connue dans les approches basées sur les Système Multi-Agents (SMA). Dans le contexte de cette thèse, nous allons proposer une nouvelle architecture basée sur les SMAs avec une nouveauté de création d'un protocole de coopération qui peut offrir une composition de services automatique et adaptable en fournissant un service composite avec une haute qualité de service (QoS).

Le processus du déploiement des services dans l'environnement du *Cloud Computing* doit être traité de manière spécifique pour rendre les services capable de fonctionner dans les bonnes conditions selon les valeurs de QoS. Afin de déployer ces services dans le *Cloud Computing*, chacun d'eux à besoin d'une collection d'autres services. Pour répondre à cette question, premièrement nous avons défini un modèle de description pour réduire le nombre de services candidats, la deuxième étape est la composition de ces services et assuré la QoS en utilisant une méthode adaptable aux problèmes de type NP-complet comme les algorithmes évolutionnaires. Dans notre cas, nous avons utilisé l'algorithme de Niched Pareto Genetic Algorithm (NPGA).

Mots clés : *Composition de services, Services Atomiques, Service Web, Agent, Déploiement de service, Cloud Computing, Service Cloud, Système Multi-Agent, QoS, Problème d'optimisation.*

Table des figures

Liste des tableaux

Table des matières

Chapitre 1

Introduction Générale

1.1 Contexte du travail

Actuellement, les services atomiques ne peuvent plus traiter toutes les exigences des clients en termes de complexité de la requête elle-même. C'est pourquoi la composition de services web est considérée comme un processus très important dans plusieurs opérations dans le web. Ces opérations se diffèrent d'une utilisation à une autre. En effet les approches proposées dans la littérature concernant la composition de services sont basées sur trois composants principaux :

1. La requête du client : elle représente le moyen de communication entre le client et le système. Les requêtes sont présentées sous forme textuelle ou sous forme d'un formulaire rempli par l'utilisateur afin d'introduire les exigences désirées.
2. Le système intermédiaire : joue un rôle très important puisqu'il est responsable de toutes les opérations de réception, d'analyse de la requête, de recherche et de composition les services conviviales aux exigences de l'utilisateur selon les caractéristiques mentionnées dans la demande.
3. Le fournisseur de service : offres des services accessibles via l'Internet.

La nécessité de composition de nombreux services est reliée au nombre important des requêtes des utilisateurs qui varie de manière exponentielle et qui comportent généralement une tâche complexe. Dans ce cas les fournisseurs s'orientent vers un déploiement des services via le Cloud pour profiter des services notamment de type "application" ou "infrastructurel" et pour exploiter les serveurs puissants qui sont distribués géographiquement.

1.2 Problématique et objectifs

Actuellement, la composition de services web représente un problème majeur dans le Cloud à cause de la diversité et la croissance exponentielle des services déployés d'après les fournisseurs. Alors les approches de composition classiques ont essayé de donner aux clients des services composites. Néanmoins, l'aspect non-fonctionnel de ces services (composites) n'est pas pris en considération. Cet aspect est relié principalement aux valeurs de la qualité de services. De plus, la représentation statique d'un service Web peut représenter un problème dans le processus de composition de services web surtout dans le cas où les fournisseurs a besoin de changer les paramètres des services déployés au cours du processus de la composition. Ceci peut conduire à donner un service incomplet ou adresser un service inexistant à la fin de ce processus. Tous ces problèmes sont traités dans la littérature de manière séparée. Ainsi, dans notre travail nous avions un challenge relatif à la résolution des problèmes de changement des paramètres durant le processus de composition des services. Pour cela, il était nécessaire de proposer un modèle de représentation dynamique des services Web qui prend en charge l'ensemble des problèmes identifiés. Ce modèle est basé sur l'utilisation d'agents coopératifs intégrés dans les couches existantes du Cloud.

1.3 Concepts généraux

Dans cette section, nous résumons les concepts de base utilisés pour proposer une architecture de composition de services à base d'agents dans le Cloud. Ce concept d'agent est utilisé dans la littérature dans plusieurs domaines tels que : l'industrie, la médecine, l'éducation ... afin de modéliser des systèmes complexes en décentralisant la gestion et l'exécution des tâches en créant des sous-systèmes représentés par des agents spécifiques.

1.3.1 notions sur le concept d'agents

1) Définition d'agent

Il existe plusieurs définitions de la notion d'agent. Parmi ces définitions lesquelles nous citons :

- un agent est défini par D'Inverno et Luck dans [25], comme une entité autonome qui s'efforce d'atteindre les objectifs qu'on conçue.
- Selon Maes [81], un agent est un système informatique qui se trouve dans un environnement complexe et dynamique, et qui aperçoit et réagit de façon autonome, afin de réaliser les buts pour lesquels il a été créé.
- Selon [42], un agent intelligent est un logiciel qui effectue un ensemble de tâches prédéterminées, avec un certain degré d'indépendance et d'autonomie, en se basant sur un ensemble de connaissances et une représentation d'objectifs prédéterminés.

En outre, on peut voir le concept d'agent en plusieurs formes selon les besoins des applications. Parmi ces formes nous citons [91] :

- L'autonomie : un agent peut agir sans revenir à son intervention extérieure afin de prendre des décisions.
- Le comportement social : un agent à la possibilité de communiquer et d'interagir avec les autres éléments de son environnement.
- La réactivité : un agent peut apercevoir et ensuite réagir aux différents évènements sur son environnement.
- La proactivité : un agent capable de prendre des initiatives afin de s'adapter au changement de son environnement.
- La persistance : un agent doit suivre son but sans interruption jusqu'à l'achèvement de ce dernier.
- Le raisonnement et la rationalité : un agent capable de raisonner rationnellement afin de choisir les meilleures actions à entreprendre pour optimiser sa productivité.
- La mobilité : un cas particulier d'agent logiciel est l'agent mobile qui a la possibilité de se déplacer d'une machine à une autre, afin d'exécuter des tâches de natures distribuées.

2) Système multi-agent

Un système multi-agent (SMA) est un système composé par une collection d'agents qui coopère entre eux afin d'arriver à résoudre un problème commun. Ces agents interagissent entre eux indirectement (en agissant sur l'environnement) ou bien directement (à travers la communication et la négociation). Les agents peuvent décider de coopérer pour un objectif commun ou bien travailler séparément pour servir leurs propres objectifs [10].

Généralement, les SMAs sont conçues pour modéliser les systèmes complexes correspondant à des ensembles constitués d'un grand nombre d'entités en interaction et situées dans un certain environnement. Ces SMAs sont utilisés de manière performante dans plusieurs domaines ; ces performances sont représentées selon les caractéristiques suivantes [118] :

- Fiabilité : correspond à la distribution des tâches sur les agents qui peuvent faciliter la résolution des problèmes d'un système de façon rapide et efficace.
- Extensibilité : est reliée à l'indépendance des agents les uns vis à vis des autres et qui rend possible la modification de leurs comportements.
- Robustesse : la coopération entre agents permet au système de satisfaire le critère de tolérance aux fautes et d'affronter les situations d'échec d'un agent.
- Maintenabilité : suite à l'inter-indépendance entre les agents, cette caractéristique permet également de maintenir chaque agent séparément des autres sans affecter le fonctionnement global du système.
- Evolutivité et flexibilité : la possibilité d'auto-adaptabilité permet aux développeurs d'ajouter ou de supprimer de nouvelles contraintes (ou même de nouveaux agents) au SMA sans pour autant d'altérer le mécanisme global du système.
- Efficacité : la capacité de communication entre agents permet de développer des systèmes de calcul distribués très puissants et très efficaces.
- Réduction des coûts : un système centralisé est toujours gourmand en temps de développement et de maintenance. Avec les SMA, on peut décentraliser la gestion et l'exécution des tâches en créant des sous-systèmes représentés par des agents spécifiques, ce qui réduit les coûts d'extension et de maintenance.

3) Agent mobile

Un agent mobile est un logiciel autonome et auto-adaptable ayant la possibilité de migrer d'un environnement à un autre et de suspendre ou changer son comportement selon les évènements et les circonstances qu'il rencontre [42].
Généralement, un agent mobile est composé : d'un code (statique), d'un état et de données. Lors de la migration d'un agent mobile d'une plate-forme à une autre, ses données et son état peuvent changer. On distingue deux types de mobilités : une forte mobilité, qui consiste à transférer le code, les données ainsi que l'état de l'agent, et une faible mobilité qui consiste à trans-

férer le code et les données seulement [63].

1.3.2 Définition de Service Web

Un service web est définit comme étant une application accessible avec des autres applications via le web [2],[72], dont les services web permettent des intéractions d'application à la demande sur les réseaux [72]. Ils ont des caractéristiques sémantiques, fonctionnelles, comportementales et non-fonctionnelles. Ainsi, l'utilisation des protocoles de communications entre les clients et les fournisseurs de services via le SOA et le SOAP. Ces services sont déployés au niveau d'UDDI sous forme des fichiers WSDL [62]. Généralement, un service Web est la logique d'application programmable disponible sur Internet. Il peut être consulté et invoqué via des normes telles que XML, WSDL, UDDI et SOAP. Avec le développement rapide du commerce électronique sur Internet, les services Web ont attiré beaucoup d'attention ces dernières années [43].

1.4 Contributions

Les contribution de cette thèse se divise en trois champs : (i) la première contribution consiste à résoudre le problème de composition de services web dans l'environnement du Cloud Computing en prenant en considération l'aspect non-fonctionnel (ou les qualités de services). Ainsi pour prendre en charge le changement des paramètres au niveau des services durant le processus de composition, nous avons proposé une architecture à base d'agents dans le Cloud Computing. Notre solution est basée sur la proposition d'un nouveau protocole pour la coopération entre les différents agents. cette interopérabilité permet de fournir aux utilisateurs les meilleurs services composites en termes de qualité de service.

(ii) La seconde contribution concerne l'assurance du bon fonctionnement de la tâche de la découverte dans le Cloud Computing. Pour cela, nous avons proposé un nouveau modèle de description de services pour réduire le nombre de services candidats tout en assurant une haute qualité de service.

(iii) La troisième contribution concerne le problème de déploiement de service. pour cela nous avons proposé d'intégrer une nouvelle couche dans l'architecture standard du Cloud Comuting. Cette solution transforme le problème du déploiement du service à un problème d'op-

timisation (en termes de qualité de service). De cette manière, notre approche donne au fournisseur de service une configuration qui permet de déployer les services avec une haute qualité de service.

1.5 Structure de la thèse

La présente thèse est organisée en huit chapitres dont les thèmes sont donnés ci-dessous :

Le premier chapitre présente le contexte et la problématique de la recherche. Ce chapitre identifie également les objectifs de la thèse et présente brièvement les principales contributions.

Le deuxième chapitre présente l'état de l'art sur la technologie du *Cloud Computing* en commençant par un historique, ensuite, nous donnons la définition du Cloud Computing ainsi que les différents modèles de service et de déploiements. De plus, il présente les composants essentiels pour un contrat de Cloud avec les majeurs fournisseurs du Cloud. Ce chapitre se termine par la présentation des avantages et des inconvénients de cette technologie avec une conclusion.

Dans le troisième chapitre nous allons présenter les approches utilisées dans l'internet (architecture orientée service ; SOA, Grid Computing et le Cloud Computing). En effet, ce chapitre commence par un cycle qui montre le problème à traiter. Avoir d'une comparaison de chaque technologie afin de faciliter la tâche de la sélection pour les utilisateurs. Ensuite, nous allons introduire les modèles de description de service et les plateformes de composition de services. On termine par mentionner quelques caractéristiques de ces plateformes de composition.

Le quatrième chapitre donne une synthèse bibliographique sur les travaux réalisés pour résoudre le problème posé. Il présente aussi les méthodes utilisées pour résoudre le problème de composition de services dans et hors le Cloud.

Dans le cinquième chapitre commence notre contribution qui consiste a résoudre le problème de changement des paramètres durant le processus de composition, il commence par un exemple qui montre l'utilité de cette solution. Après, nous allons présenter l'architecture globale du système en détaillant son fonctionnement à l'aide d'un diagramme de séquence UML pour simplifier la compréhension de notre travail. Ensuite, nous allons expliquer l'architecture détaillée de chaque composant (sous-système) et son fonctionnement, nous terminerons ce chapitre par une conclusion.

Le sixième chapitre est cœur de la deuxième contribution qui vise a donne aux fournisseurs du service la possibilité de déployer les services avec une meilleure valeurs de QoS. Ce chapitre commence par un exemple qui montre l'utilité de cette solution. Ensuite nous présentons le modèle de description utilisé dans notre solution. Après, nous allons présenter l'architecture globale du système en détaillant son fonctionnement à l'aide d'un diagramme de séquence UML pour simplifier la compréhension de notre travail.

Le septième chapitre montre la mise en œuvre de notre approche, il commence par la présentation des outils et l'environnement de développement. Ensuite, nous allons présenter quelques interfaces qui montrent les résultats obtenus d'après l'implantation de notre modèle, nous terminons le par une discussion sur les résultats obtenus.

Finalement, le huitième chapitre conclus cette thèse. Il synthétise les contributions globales et met en évidence les perspectives de cette recherche.

Première partie

Etat de l’art

Chapitre 2

Cloud Computing

2.1 Introduction

Actuellemnt, le Cloud Computing représente une révolution dans le monde informatique. En effet le Cloud Computing est émergé dans les dernières années comme une solution universelle utilisée par différents types d'utilisateurs. Bien que l'idée de virtualisation était la première caractéristique reliée à un espace de stockage très important et un moyen de calcul puissant. Ainsi, le concept du Cloud Computing a été proposé par John McCarthy en déclarant : « dans le future, l'informatique devient un outil publique et organisé » [13]. L'arrivé du Cloud représente la solution aux limites des serveurs traditionnels d'utilisation tel que la qualité faible de la. ces limites concernent principalement la bande passante et le contrôle de données [134]. Pour cela le cloud computing permet de pallier à ces limites. La première génération du Cloud Computing a été proposée pour le domaine e-commerce. Pour que les fournisseurs de Cloud offrent ensuite des solutions et des opportunités pour développer les différents services comme une liste de services internet consommés.

A l'arrivé du cloud encouragée par les majeurs corporations comme Amazon Web Services et Google, cette technologie devient le 5ième utilitaire après l'eau, électricité, gaz, et la consommation téléphonique [134]. Ainsi cette technologie est devenu populaire par de nombreux centres de données situés dans différentes localisations géographiques à travers le monde entier. Ces centres de données fournissent des capacités de calcul et de traitement pour le Cloud qui représentent une nouvelle vision de déploiement des services pour donner la possibilité aux clients d'accéder, de travailler, de partager et de stocker des informations en utilisant l'internet [122].

Dans ce chapitre, nous allons présenter la technologie du *Cloud Computing* dont la première définition a été qui était proposée par le *NIST*[1]. pour cela nous présentons les caractéristiques et les modèles de cette technologie ainsi que les caractéristiques de ses services.

Ce chapitre se termine par la présentation des avantages et des inconvénients de cette technologie en présentant les composants essentiels pour un contrat de Cloud.

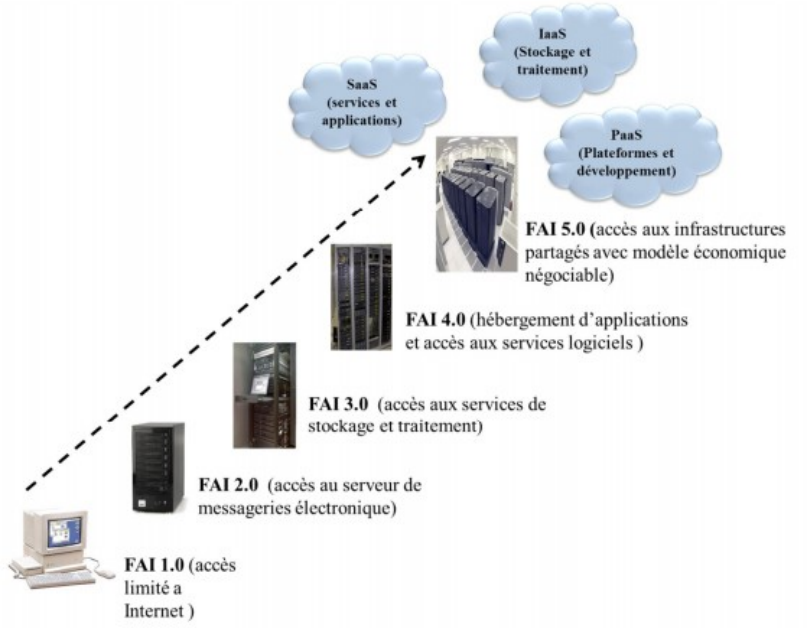

FIGURE 2.1 – Evolution vers le Cloud [42].

2.2 Définition du Cloud Computing

Dans la littérature, il existe plusieurs définitions concernant le concept du Cloud Computing depuis l'année 2007 jusqu'à maintenant. Ces définitions sont basées sur la première qui était proposée par le NIST. Dans cette section, nous allons présenter un ensemble de définitions pour montrer les différentes perspectives du Cloud :

- D'après la définition de NIST [86]« le Cloud Computing est un modèle qui permet d'accéder au réseau, de façon ubiquitaire facile et à la demande, à un ensemble de ressources informatiques partagées et reconfigurables (réseaux, serveurs, stockage, applications et services). Ces ressources peuvent être fournies ou libérées avec un système de gestion minimal et selon

1. National Institute of Standards and Technology

les intéractions avec le fournisseur de services.»

- Selon [102], le Cloud Computing est une technologie de livraison dynamique de ressources informatiques et des capacités technologiques représentées comme étant un service sur internet. De plus, le Cloud est un aspect informatique dans lequel les ressources sont dynamiquement évolutives et souvent virtualisées et fournies en tant que service sur internet.

- Selon les auteurs de [132], *Gartner Research* présente le *Cloud Computing* comme un style de calcul où les grandes capacités de calcul sont fournies en tant que service à travers l'internet pour servir les multiples clients externes.

- Selon [87], le Cloud Computing est défini comme une utilisation de la technologie informatique qui peut exploiter la puissance de traitement de nombreuses machines inter-réseau tout en cachant la structure réelle. Les utilisateurs n'ont pas besoin d'avoir des connaissances en expertise ou de contrôler l'infrastructure technologique dans le *Cloud* qui les soutient.

- Les auteurs de [100] ont défini le *Cloud* comme étant un ensemble de services informatiques offerts par un tiers, disponibles à la demande, pouvant être mis à l'échelle en fonction des besoins changeants. De plus, le *Cloud Computing* représente un écart par rapport à la norme de développement, d'exploitation et de gestion des systèmes informatiques.

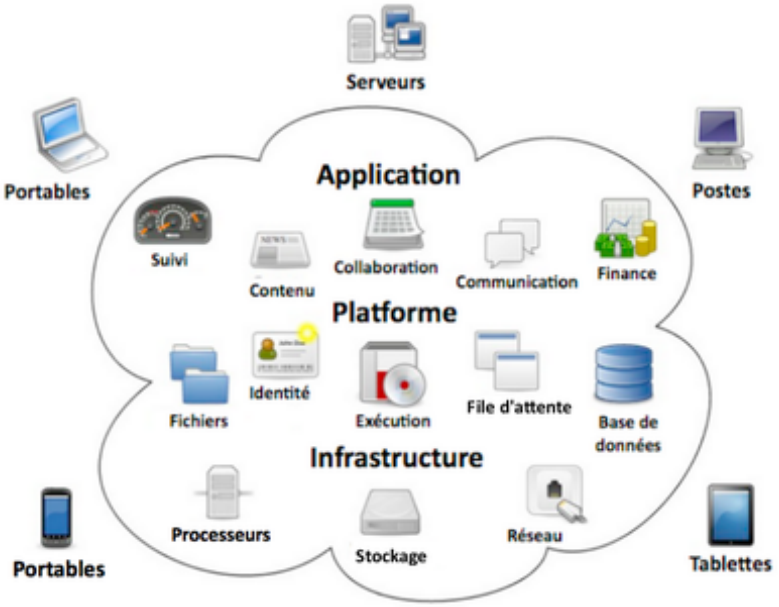

FIGURE 2.2 – Le Cloud Computing et ces applications.

2.3 Les caractéristiques du Cloud Computing

La technologie du *Cloud Computing* a été émergée avec des caractéristiques qui la différencie des autres technologies *(Grid computing, SOA, ...)*. Comme nous avons vu dans la section précédente que le Cloud donne des avantages dans le sens d'un aspect économique [100]. D'après le NIST, sept caractéristiques essentielles correspondantes au *Cloud Computing* qui sont illustrées dans la figure suivante [69] :

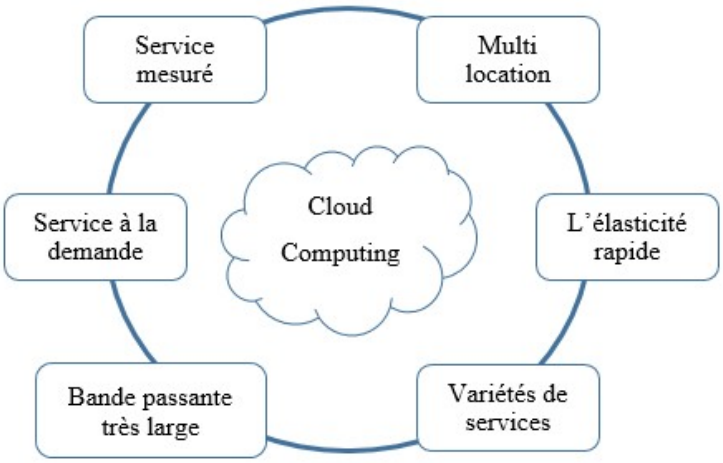

FIGURE 2.3 – Les caractéristiques du Cloud Computing.

2.3.1 Service à la demande

N'importe quel utilisateur (entreprise ou client simple) peut demander un ou plusieurs services. Les clients peuvent demander à chaque fois ce qu'ils ont besoin selon un mode de paiement. Ces services sont fournis aux clients à travers une interface graphique en ligne et sans interactions directe avec les fournisseurs.

2.3.2 Bande passante très large

Parmi les caractéristiques les plus importantes dans le Cloud Computing est d'offrir une bande passante très large. Cette caractéristique est importante pour les demandeurs de services afin obtenir les services dans un délai satisfaisant malgè que les ressources et les services sont localisés dans des localisations géographiques différentes.

2.3.3 Variétés des services

Avec cette caractéristique le *Cloud* offre un ensemble de ressources qui apparaissent comme étant une seule ressource fusionnée [138] de telle sorte que l'utilisateur de services n'aurait pas besoin de connaitre l'emplacement de ces ressources. DE plus cette approche permet aux fournisseurs de services de fournir plusieurs ressources (réel ou virtuel) dans le cloud de façon dynamique.

2.3.4 L'élasticité rapide

Elasticité est connue aussi par le terme extensibilité de sorte que les ressources peuvent être allouées ou libérées selon les besoins d'un service quelconque. Ce qui conduit en général les utilisateurs à demander autant de fois les différents services ou les ressources souhaitées.

2.3.5 Service mesuré

Les différents aspects du Cloud doivent être contrôlé, surveillé, optimisé, et fourni des rapports à plusieurs niveaux pour les ressources des vendeurs et les consommateurs de façon automatique.

2.3.6 Multi location

Celle-ci est suggérée par les alliances de sécurité du Cloud. Ce terme signifie : avoir des modèles pour une politique d'exécution des applications, de segmentation,d'isolation, de gouvernance, de niveau de service et de facturation pour les différentes catégories de consommateurs.

2.3.7 Vérifiabilité et certification

Dans le sens où les utilisateurs peuvent utiliser des services de façon appropriée, il est important de mentionner que les fournisseurs de services doivent préparer des logs afin de donner une possibilité d'évaluer le degré de régulation et les politiques observées.

2.4 Les modèles de livraison des services Cloud

Cette section concerne principalement les différents types de services qui couvrent la majorité de besoins des consommateurs. selon la définition NIST du Cloud (voir section) , on peut

distinguer plusieurs modèles de services*cloud* récents. Ces modèles sont basés sur les trois services essentiels cités comme suit :

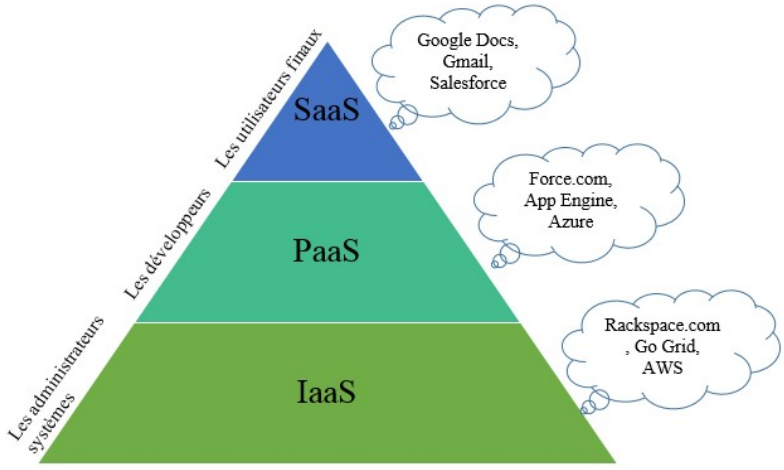

FIGURE 2.4 – Le Cloud Computing et ces applications.

2.4.1 Infrastructure en tant que service

Les services d'infrastructures (*Infrastracture as a Service : IaaS*) offrent des plateformes de virtualisation qui représentent l'évolution de virtualisation des serveurs privés ultérieurs [46]. Les clients de services achètent des ressources au lieu de configurer un ensemble de serveurs, logiciels, ou d'espaces dans les *data center* eux-mêmes. De plus, les utilisateurs de ce type de service doivent être facturés en fonction des ressources consommées [47]. Ce modèle de service permet aux fournisseurs de services de déployer leurs applications dans des machines virtuelles configurables. Ainsi, selon les besoins des utilisateurs, le nombre de machines virtuelles louées peuvent augmenter ou diminuer selon les ressources demandées et les exigences des applications et selon leurs bon fonctionnement. De plus les fournisseurs de services infrastructure offrent des interfaces graphiques afin de faciliter la tâche de la configuration.

2.4.2 Plateforme en tant que service

Les fournisseurs de service PaaS(*platform as a service : PaaS*) proposent une infrastructure logicielle gérée de niveau supérieur, où les clients peuvent construire et déployer des classes d'applications particulières et de services en utilisant des outils, environnements et langages de

programmation qui sont pris en charge par le fournisseur [47]. En outre, cette couche offre l'utilisation des ressources infra-structurelles tels que les serveurs, réseaux, l'espace du stockage ou les systèmes d'exploitations. Mais les clients n'ont aucun moyen de les contrôler car elles sont abstraites [46][85]. De plus, les services plateformes sont destinés à des domaines spécifiques, tel que le développement d'applications web dépendant du langage de programmation. Les clients disposent un environnement séparé pour tester et développer ou déployer de façon permanente leurs applications [39].

2.4.3 Application en tant que service

Les services Applications (*Software as a Service : SaaS*) offrent des applications déjà créées qui s'exécutent dans une infrastructure Cloud. La plupart des services Cloud Computing en tant que logiciels sont des applications web qui peuvent accéder à travers différents périphériques clients via une interface client légère comme un navigateur web afin d'envoyer des données et de recevoir des résultats [69][47]. De plus, les clients qui utilisent ces services n'ont pas le droit de gérer ou de contrôler l'infrastructure sous-jacente du Cloud ou la plateforme d'application sauf quelques configurations bien spécifiques [37].

D'après l'utilisation de différents types de services et les différents besoins de consommateurs, les fournisseurs pensent à définir de nouveaux services à la demande. Ces derniers sont déterminés à base des trois types de services de base définis par NIST et présentés précédemment. De plus, ces services sont connus sous le nom "everything as service" (XaaS) qui comporte les services de réseaux, données, bureau, sécurité, ... [143].

2.5 Les caractéristiques de services cloud

Dans cette section, nous allons présenter les caractéristiques de services Cloud. Les services Cloud ont des caractéristiques communes et d'autres spécifiques pour chaque catégorie.

2.5.1 Les caractéristiques communes

Les caractéristiques partagées entre les services Cloud sont présentés comme suit :

- Type de licence

Dans le Cloud Computing, la majorité des services utilisent des logiciels propriétaires et des licences. Par contre, il existe d'autres fournisseurs de services Cloud qui utilisent l'open source pour les services SaaS et PaaS [47]. De plus, les licences jouent un rôle lors de l'offre de services de types infrastructurel et de plateforme. Lors de l'opération de la location des serveurs virtuels sans système d'exploitation installé, les fournisseurs de service infrastructurel ne souffrent pas de problèmes de licence des logiciels. Par contre, le cas où les fournisseurs incluent un système et des progiciels, un problème potentiel peut être posé (utilisation de service pour une période du temps limitée avec des frais additionnels) [46][6][125].

- Le groupe d'utilisateur ciblé

Dans le cloud computing, on peut citer deux catégories de services : la première pour les entreprises et l'autre pour les utilisateurs (utilisation privé). Par exemple les services de type IaaS et PaaS sont généralement proposés pour les entreprises, tandis que les services de type SaaS sont destinés pour les entreprises et/ou les utilisateurs individus [47] avec l'utilisation éventuelle des services IaaS et PaaS pour les utilisateurs simples.

- La sécurité et la confidentialité

Dans tous les systèmes, la sécurité et la confidentialité jouent un rôle très important, surtout dans le cas où les données sensitives résident dans les serveurs Cloud. La perte ou l'accès illégal sur les données peuvent donner des conséquences mal entendus surtout pour les données sensibles (le cas des banques ou des gouvernements). [47] Pour cette raison-là, les fournisseurs de services cloud doivent prendre en considération ce point par l'introduction des politiques de cryptage et d'authentification très efficaces.

- Les systèmes du paiement

Le paiement dans le Cloud est un paiement dynamique [46][135] qui est totalement différent du type traditionnel. Dans ce cas, le consommateur paie seulement pour les ressources utilisées, ces derniers peuvent être des instances de machine virtuelle, espace de stockage, bande passante, temps du calcul, transactions ou pour la combinaison de ces ressources [47].

- Standardisation

La standardisation dans le Cloud signifie l'utilisation d'un *API* commun avec des normes techniques différentes [46] et d'architectures. De plus, les standards peuvent augmenter l'interopérabilité et permettent une personnalisation possible dû à la transparence technique. La standardisation est appliquée dans les architectures Cloud, dans les protocoles, les identifiants et le langage de description de service Cloud et les technologies de virtualisation [47][125].

- Le contrat entre client-fournisseur

Pour assurer le bon fonctionnement du processus de la demande et la réception d'un service de façon parfaite, le Cloud Computing utilise des contrats entre les clients et les fournisseurs de services [157]. De plus, ce dernier est présenté dans le Cloud par le nom du SLA (Service Level Agreement) qui désigne le service contractuel et sa performance. Ce contrat est un document qui spécifie la description du paramètre de niveau de service, l'objectif de niveau de service, le service convenu, les garanties et les actions en cas de violation [18]. En outre, SLA est utilisé pour construire des tâches telles que ; le déploiement et la surveillance les services [18].

2.5.2 Les caractéristiques spécifiques

Après la représentation des caractéristiques communes dans cette section, nous allons présenter les caractéristiques spécifiques pour chaque modèle de service. Ces caractéristiques sont présentées comme suit :

- Les caractéristiques spécifique d'un service IaaS

Les caractéristiques considérées dans ce modèle sont les systèmes d'exploitation pris en charge et les applications/Frameworks. Cela peut donner une importance pour des consommateurs de service spécifique. La majorité des fournisseurs IaaS supportent le système Linux, mais certains supportent le Windows et Open-Solaris. De plus, dans ce modèle, les applications incluent le serveur HTTP, Apache et les logiciels de bases de données MySQL [47]. Les fournisseurs de Cloud offrent aux développeurs des interfaces API ou des outils de commandes spéciales [47].

- Les caractéristiques spécifiques d'un service PaaS

Des caractéristiques importantes pour le niveau plateforme sont liées aux langages de programmations et les environnements. Par exemple : Google's App Engine supporte seulement les environnements Python et Java. Les systèmes d'exploitation et les applications prisent en charge peuvent être également une caractéristique pertinente [47].

- Les caractéristiques spécifiques d'un service SaaS

Les services Cloud de modèle logiciels sont variés et beaucoup. Une caractéristique à considérer est le domaine client/application du service offert. Ce domaine pourrait avoir des relations avec : la clientèle ou d'autres domaines de gestion d'entreprise, les applications bureautiques, les réseaux sociaux et l'échange de données [47].

2.6 Les techniques de déploiements du Cloud Computing

Les services du Cloud Computing existants sont différenciés par rapport à leurs utilisations soit pour des entreprises ou autre type d'utilisateurs. Pour cela, le Cloud offre quatre types de déploiement. De plus, cette opération, met dans le Cloud, est définie comme étant une manière qui désigne l'offre d'un service particulier.

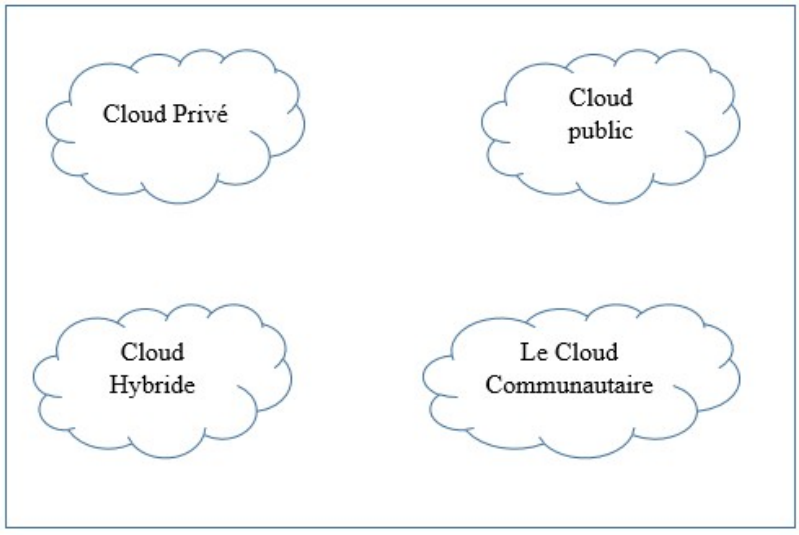

FIGURE 2.5 – Les modèles de déploiement du Cloud.

2.6.1 Le Cloud Public

Dans ce type de déploiement, le fournisseur et le consommateur de service sont des organisations différentes. De plus, ce type implémente toujours des modèles commercialisés. L'infrastructure de Cloud est mise à la disposition du grand public ou d'un grand groupe industriel et est possédée par une organisation qui offre des services Cloud [102]. Le Cloud public est un concept qui réalise le partage des services et l'infrastructure fournie par des fournisseurs de services dans un environnement multi-locataire [140]. Bien que, l'utilisation de ce type de ressources soit auto-approvisionnée dynamiquement via des applications ou des services web mis par le fournisseur de ressources qui partagent les services.

Dans ce type de déploiement, l'infrastructure est la propriété de tierce partie et les utilisateurs n'ont pas l'accès à toutes les fonctionnalités du service. Par conséquent, les consommateurs de ce type de service (Cloud public) perdent le contrôle physique de ses ressources [82].

Pour l'aspect de la sécurité et la confidentialité dans le Cloud public, les utilisateurs de ce genre du déploiement rencontrent des problèmes liés aux partages des données (les données peuvent être utilisées par les fournisseurs de services). De plus, la sécurité et les lois de chaque zone géographique où l'emplacement de données sont situées à travers le monde (lorsque l'emplacement des données est situé dans des différentes régions ; Asie, Europe,... ; les politiques de sécurité seront changées d'une zone à une autre).

2.6.2 Le Cloud Privé

Ce type de déploiement est destiné aux organisations ou aux institutions privés. On peut dire que ce type prend les avantages du Cloud public exclusivement en faveur des organisations privées [69]. Ces modèles de déploiement sont des réseaux propriétaires, souvent des centres de données résident dans l'entreprise pour l'utilisation exclusive de l'organisation. En outre, les entreprises, qui utilisent le Cloud privé, ont pris en charge le contrôle et la gestion des ressources Cloud. De plus, l'entreprise peut mieux contrôler les problèmes de sécurité et de conformité réglementaire [157].

Dans le Cloud privé, on peut distinguer quatre types [115] présentés comme suit :

– Cloud privé typique : l'organisation hébergeant le Cloud dans l'un de ces propres centres de données derrière le pare-feu de l'entreprise. L'architecture est semblable à l'utilisation

de l'intranet par laquelle l'organisation est basée sur les techniques d'internet mais par un accès limité pour les employés internes.

- Cloud privé géré : dans ce cas, l'organisation possède toujours l'infrastructure dans ces propres centres de données, mais la gestion est mise sous le contrôle de tierce (géré par un fournisseur tiers).
- Cloud privé hébergé : les fournisseurs de service Cloud offrent l'infrastructure nécessaire, et sont eux qui prennent la responsabilité de gestion. Les bénéfices pour le consommateur de service, dans ce cas, ne sont que l'évolutivité, l'élasticité et la disponibilité de la demande qui sont garantis par le fournisseur. Ainsi que, les serveurs ne sont pas partagés avec d'autres organisations et les fournisseurs de services offrent un très grand moyen de sécurité.
- Cloud privé virtuel (VPC) : ce type de Cloud est proposé par les fournisseurs de services Cloud dans un environnement multi locataire.

2.6.3 Le Cloud Hybride

Ce type du Cloud connu, aussi, par le Cloud de fédération [116] est une composition entre deux modèles de service définis précédemment (Cloud public et privé). Le Cloud hybride utilise les ressources de calcule propriétaires gérées directement par l'organisation et le Cloud public pour quelque besoin de calcule [14]. Le Cloud hybride typique utilise l'infrastructure en tant que service pour construire la partie du Cloud public de Cloud hybride. De plus, ce type du Cloud est approprié pour les grandes ou les moyennes entreprises, car il offre un système intensif de calcul ainsi que les avantages de service Cloud public tel que : la disponibilité, l'évolutivité et le modèle de paiement -payez ce que vous utilisez- [127]. Le but de Cloud hybride est de fournir un modèle qui offre des tolérances aux pannes et haute disponibilité [101]. Maintenant, les majeurs fournisseurs du Cloud tel que : Amazon, VMware et HP fournissent des services Cloud hybride. Dans ce cas, on peut trouver certaines applications hébergées dans deux environnements. En cas où l'un de ces environnements tombe en panne, le consommateur de service peut toujours accéder à l'application. Le Cloud hybride soufre de quelques problèmes, par ce qu'il représente l'environnement le plus complexe. En tant qu'il est composé par deux types différents du Cloud (public et privé), les lois et les opérations ne peuvent pas les appliquées dans tous les environ-

nements. Afin de résoudre ce problème, il faut développer un ensemble de lois et opérations pour chaque environnement.

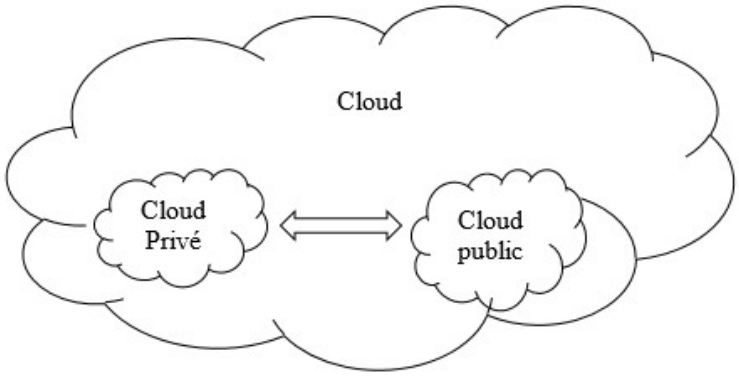

FIGURE 2.6 – Le Cloud hybride.

2.6.4 Le Cloud Communautaire

Après avoir présenté les deux types de Cloud extrêmes (public et privé) et la combinaison entre les deux (hybride). Maintenant, nous allons présenter le dernier type, le Cloud communautaire ou appelé aussi le Cloud collective. En effet, ce type du Cloud est semblable à celui qui est public où le cas d'infrastructure multi locataires. Néanmoins, ce type diverge dans le cas où les membres de la communauté sont présentés par une partie de groupes spécifiques avec une préoccupation de calcul dépend aux niveaux de services, sécurité et de la conformité réglementaire...etc. [75]. En outre, dans ce modèle, on peut distinguer deux types basics : fédérer et modèle basé sur un courtier [115]. Dans le modèle fédérer, les entreprises, qui appartiennent au même secteur, participant au Cloud communautaire où toute ressource de calcule inutile dans une organisation peut être utilisée par une autre organisation membre à la demande (voir la Figure 2.7). Le deuxième modèle, le tiers de confiance, sert d'un broker et des interfaces avec les différents membres de la communauté. Le broker est responsable de l'acquisition des différents services essentiels au secteur industriel et les met à la disposition de tous les membres (voir la Figure 2.8).

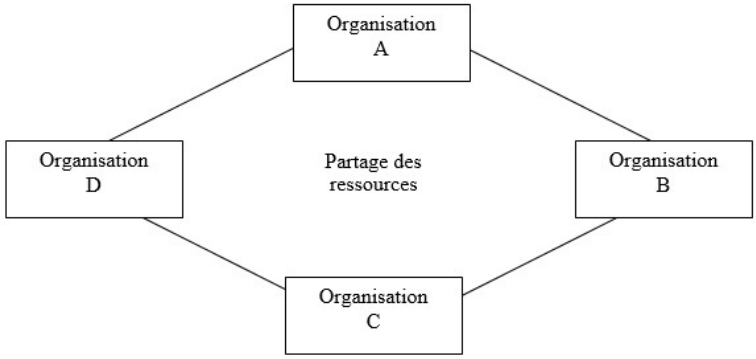

FIGURE 2.7 – Le Cloud communautaire (le modèle fédérer).

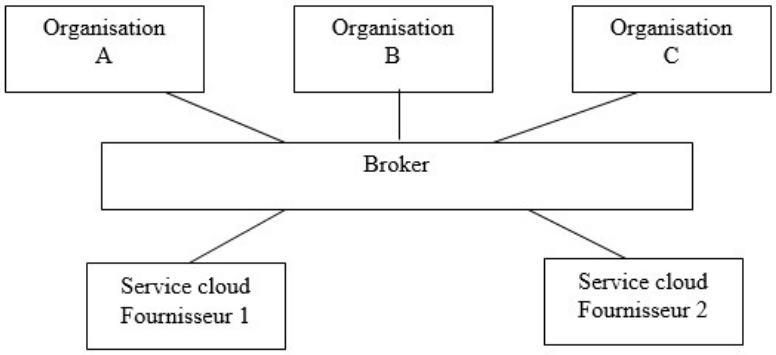

FIGURE 2.8 – Le Cloud communautaire (le modèle à base de broker).

2.7 Les composants essentiels d'un contrat de Cloud

Comme nous avons présenté dans la section précédente (section 2.6), la définition d'un contrat de Cloud (SLA). Il s'agit d'un document entre le fournisseur et le client. Afin de bien comprendre ce concept, dans cette section, nous allons présenter les composants essentiels de ce dernier. Le contenu typique d'un SLA est présenté dans le travail de [70], et est présenté comme suit :

2.7.1 Définition de service

La première chose très importante dans un contrat est de décrire les besoins de l'utilisateur pour fournir un service qui répond au ses besoins. La définition des informations d'un service est une tâche sensible. Pour cela, le contrat doit prendre des spécifications détaillées de service à délivré.

2.7.2 La gestion du rendement

Un point très important dans un contrat SLA est : assurer un moyen efficace pour mesurer le rendement. De plus, chaque service doit être mesuré et les résultats doivent être analysés et rapportés. Afin de garantir le niveau de performance du service, ce contrat doit être régulièrement examiné par les deux parties (fournisseur et le client).

2.7.3 La gestion des problèmes

Le but de cet point est de minimiser l'impact négatif des incidents et des problèmes. En effet, les problèmes consistent en ceux du type imprévu et les propositions ou le plan à suivre pour les résoudre.

2.7.4 Les responsabilités et les obligations du client

Le client doit connaitre qu'il peut prendre des responsabilités sur l'opération de livraison du service, dans lequel il doit faciliter les opérations d'accès, d'installation et de fournir des ressources nécessaires pour les employés du fournisseur.

2.7.5 Garanties et remèdes

Ce point comporte trois sujets : indemnités de qualité du service, réclamations de la troisième partie, remèdes pour infractions, exclusions, et la force majeure.

2.7.6 Sécurité

La sécurité dans n'importe quel domaine représente une caractéristique critique. Chaque client doit fournir un accès physique et logique contrôlé à ses informations. De même façon, le fournisseur doit respecter les politiques de sécurité du client.

2.7.7 Résiliation

Ce composant, dans un contrat SLA, comporte les points suivants :

- Résiliation à la fin de la période initiale.
- Résiliation pour plus de commodité.
- Résiliation pour cause.

2.8 Les avantages et les inconvénients du Cloud Computing

Le Cloud Computing représente la révolution dans l'informatique dans ces dernières années, ce paradigme-là arrive avec des avantages qui peuvent corriger les limites des autres technologies existées. Dans cette section, nous allons présenter quelques points forts de ce paradigme. Lorsqu'on parle d'un nouveau paradigme, les chercheurs ou les clients peuvent rencontrer toujours quelques limites [87]. Les avantages et les limites de ce paradigme seront présentés comme suit :

- Flexibilité : le Cloud utilise multi-locataire pour ces ressources au cours de l'exécution, une application peut utiliser plusieurs ressources. Cela offre une possibilité de demander d'autres ressources s'il est besoins [87].
- Optimisation des coûts : réduction des effectifs informatiques et fixation du prix en fonction de la durée d'utilisation des ressources informatiques sans investissement initiale lourd.
- Évolutivité : ça fait référence à un utilisateur Cloud (entreprise) qui peut utiliser d'énormes calculs de données dans un temps spécifique. A la fin du processus, le système peut retourner aux normes, tous sans nécessiter ces serveurs lourds [87].
- Portabilité : les organisations peuvent utiliser leurs puissances informatiques partout où les utilisateurs peuvent avoir un accès dans n'importe quelle localisation géographique [87].
- Sécurité : dans le Cloud, les fournisseurs utilisent des stratégies afin de garantir la vie privée de chaque utilisateur par : la réplication des données, plan de reprise d'activité, etc.
- Simplicité d'utilisation : le Cloud offre des applications et des services installés et faciles à utiliser à travers des pages web.

Comme nous avons mentionné, chaque nouvelle technologie arrivée porte des avantages et malheureusement suivi par quelques limites rencontrées par des utilisateurs du Cloud. Ces limites sont présentées comme suit :

- La gestion d'énergie : afin de définir un plan d'utilisation des ressources, le fournisseur doit définir une stratégie pour la gestion d'énergie (consommation d'électricité) [87].
- Confidentialité et sécurité : dans n'importe quelle technologie, la sécurité pose toujours

des problèmes. Le problème concerne les attaques lors des opérations du transfert de données. Dans le cloud, ce problème est posé dans les cas des cloud publiques [87].

– Gestion de ressources : ce problème représente toujours les limites de chaque technologie. A cause de la nature multidimensionnelle des machines virtuelles, la gestion des ressources sera compliquée [15].
– Dépendance : en cas ou l'entreprise (ou client final) souhaite des fonctionnalités très spécifiques, il est peut être difficile de convaincre le fournisseur de proposer ces fonctionnalités. Le client final doit choisir un fournisseur en qu'il a la confiance.
– Migration vers une autre offre difficile : il n'existe pas pour l'instant un standard entre les différents acteurs du domaine, donc, le risque d'incompatibilité du transfert de données.

2.9 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur la technologie du Cloud Computing. Comme nous avons vu que le Cloud pense à résoudre les problèmes posés dans les autres technologies. De plus, le Cloud représente une révolution car il offre des services de plusieurs types. Grâce aux bénéfices offerts par le fournisseur du Cloud, les utilisateurs finaux et les entreprises trouvent que le Cloud est un bon choix pour l'utilisation de ces services.

Toutes ces solutions et les avantages produits dans le Cloud, il existe toujours des limites. Quand on parle de la sécurité par ce qu'elle représente la vie privée des clients. Mais tout le monde maintenant utilise le *Cloud Computing* que ce soit personnel ou partagé avec les membres des équipes de la société.

Dans le prochain chapitre, nous allons expliquer les différentes technologies avant l'arrivée du Cloud Computing. Ensuite, nous allons présenter quelques comparaisons entre ces technologies pour bien comprendre la différence entre eux.

Chapitre 3

Technologies des services Web

3.1 Introduction

Les chercheurs proposent des modèles et des technologies qui offres des bénéfices aux entreprises et aux utilisateurs. Ces technologies sont diverses en termes d'architecture ou de stratégie de fonctionnement. De plus, on ne peut pas dire qu'une technologie est meilleure qu'une autre parce ce que cela dépend du problème posé et la nature de la requête. Par contre, les utilisateurs sollicites toujours des solutions de plus en plus compétitives et rentable.

Dans ce chapitre, nous présentent les principaux technologies qui environnent les services web, a savoir : architecture orientée service ; *SOA, Grid Computing* et le *Cloud Computing*. Il introduit également une comparaison entre ces technologies afin de faciliter la tâche de sélection pour les utilisateurs. Ensuite, nous allons introduire les modèles de description de service et les plateformes de composition de services. On termine par mentionner quelques caractéristiques de ces plateformes de composition.

3.2 Le cycle de vie de composition de services

Nous allons utiliser le processus défini dans [110] afin d'illustrer les étapes nécessaires de composition de services. La composition de services est basée sur un ensemble d'étapes présentées comme suit :

3.2.1 Phase de définition

Dans cette étape, l'utilisateur introduit la demande qui comporte les informations sur le service voulu. Ensuite, les demandes seront décomposées (requête comporte plusieurs information) de façon automatique en utilisant des ontologies ou d'autres modèles en tenant en compte les qualités de services suggérés par l'utilisateur.

3.2.2 Phase de sélection

Dans cette phase, à l'intérieur de chaque activité dans le service composite, on cherche dans un registre de services en ceux qui concernent le type de service convenables à chaque activité à partir de la description du service déployé. Dans cette étape, on peut trouver des milliers de service qui rencontrent les demandes de l'utilisateur. Pour choisir le service adéquat à une certaine activité, on doit utiliser un algorithme de matching. A la fin de cette étape, on peut composer un ensemble de services qui répondent aux besoins de l'utilisateur.

3.2.3 Phase de déploiement

Dans cette phase, le service composite va être déployé afin d'être utilisé à partir des opérations d'instanciation et d'invocation d'après l'utilisateur final. A la fin de cette phase, nous obtenons un service composite exécutable.

3.2.4 Phase d'exécution

Dans cette étape, une instance d'un service composite sera créée et exécutée par le moteur d'exécution qui est responsable à l'invocation des composants du service individuel. Au cours de l'exécution, plusieurs opérations seront réalisées telle que : les taches de surveillance, y'compris l'enregistrement, suivi de l'exécution, mesure du rendement et gestion des exceptions.

On peut trouver des méthodes de composition automatique qui fusionnent les deux premières étapes dans une seule étape [110]. De telle façon, le service composite produit est directement généré selon les conditions de composition sans créer le service composite abstrait.

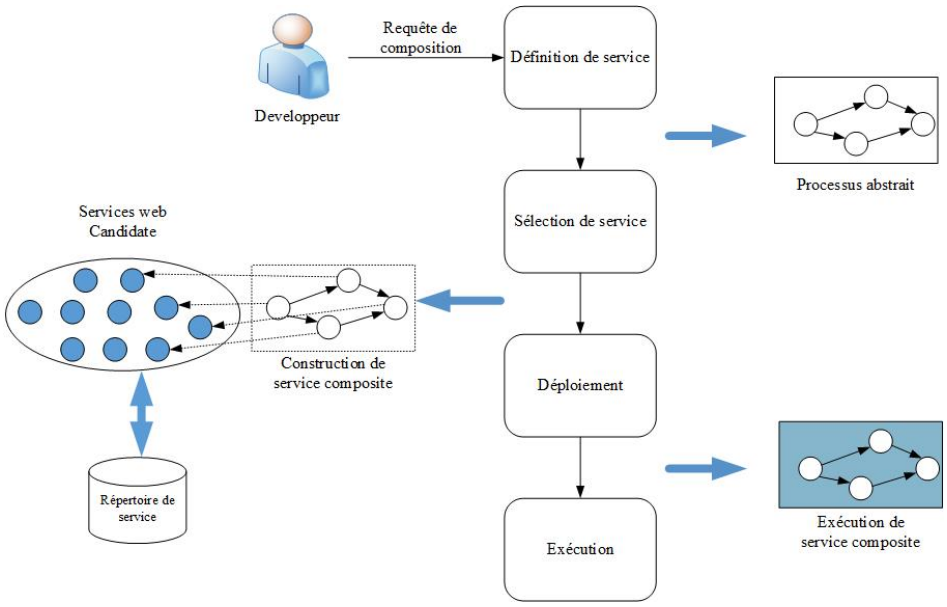

FIGURE 3.1 – Le cycle de vie de composition de service [110].

3.3 La relation entre un service Cloud et un service web

Après l'émergence du Cloud Computing, plusieurs chercheurs pensent que les services Cloud et les services Web sont semblables. Par contre, les services Web sont utilisés pour les connexions entre les services [8]. D'après le diagramme proposé par Venn [8], le *Cloud Computing* utilise les services Web pour les connexions. Il existe d'autre relation entre les deux concepts, mais la majorité d'utilisation ayant pour but de se connecter. Les services Web utilisent des langages de description afin de décrire le service et pour être facile à interpréter. Mais pour un service Cloud, on trouve des difficultés afin de trouver un modèle unifié à cause de la diversité des services et la nature des architectures de service Cloud.

3.4 Grid Computing versus Cloud Computing

Dans cette section, nous allons citer quelques différences entre les deux technologies. Avant de citer la différence entre les deux, le *Grid Computing* est une technologie qui utilise des ordinateurs et des ressources interconnectés répartis collectivement pour améliorer les performances de calcul et le partage des ressources [137]. D'après la littérature, il existe plusieurs in-

tersections entre les deux, mais ils se distinguent en quelque points, ces différences sont présentées comme suit [17][37][44] :

TABLE 3.1 – Comparaison entre le *Grid Computing* et le *Cloud Computing*.

	Technologie	
Caractéristique	**Cloud Computing**	**Grid Computing**
L'objectif des applications	Les applications sont proposées pour les cas du business.	Les applications sont destinées à la collaboration des projets scientifiques et de recherche.
La gestion d'infrastructure	La gestion est centralisée avec un accès unique.	La gestion est décentralisés et qu'elles s'étendent sur des sites répartis géographiquement.
Le modèle économique	Le paiement est basé sur l'utilisation *"Pay as you go"*.	On ne trouve pas un modèle économique fondamental.
Modèles d'utilisation des ressources	Les ressources sont virtualisées et forment une couche homogène.	Les ressources sont hétérogènes.
L'interopérabilité	Le verrouillage du fournisseur et l'intégration sont des problèmes.	L'interopérabilité entre les fournisseurs ne pose pas un problème.

3.5 SOA versus Cloud computing

Il existe plusieurs entreprises qui travaillent toujours avec l'architecture SOA malgré l'apparition du Cloud Computing. Dans la littérature, les chercheurs trouvent qu'il existe une relation entre les deux technologies, cette dernière est représentée par l'utilisation du SOA dans le Cloud [8]. Avant de citer la différence entre les deux technologies, un SOA est définit comme étant une approche destinée à favoriser la flexibilité grâce à l'encapsulation et le faiblement couplé [100].

Mais ces deux technologies se diffèrent en quelques points présentés comme suit [95] :

TABLE 3.2 – Comparaison entre *SOA* et le *Cloud Computing*.

	Technologie	
Caractéristique	**SOA**	**Cloud Computing**
L'intérêt de la technologie	Se concentre sur les compositions et qui inclut l'aspect physique dans chaque couche.	Reste éloigné de l'implémentation et du déploiement physiques.
Les acteurs dans les technologies	Le matériel et les logiciels qui sont utilises comme un rôles de service consommateur et fournisseur de services.	Clairement identifié les différents rôles tels que fournisseurs, brokers et les utilisateurs des services.
Le modèle économique	On ne trouve pas un modèle économique fondamental.	Le paiement est basé sur l'utilisation *"Pay as you go"*.
L'interopérabilité	Les applications sont utilisables par différentes plateformes.	On ne peut pas voir une application qui fonctionne dans un autre fournisseur.

3.6 Les languages de description de service

Dans la littérature, il existe plusieurs travaux proposés pour décrire les services web. Ces modèles sont utilisés pour présenter les services de façon compréhensible aux systèmes utilisés. Les modèles de description de service sont repartis en deux catégories qui sont : description à base syntaxique et sémantique. Dans cette section nous allons présenter quelques modèles.

3.6.1 WSDL

Un WSDL (Web Service Description Language) est un fichier XML pour décrire les services et leur lien avec les adresses réseaux spécifiques. Les fichiers WSDL contiennent les définitions et les opérations. Généralement, les définitions exprimés en XML incluent à la fois les définitions

de type de données et les définitions de messages qui utilisent les définitions de type de données [8].

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/saop/" xmlns:="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" wmlns:s0="http://www.myapp.org"
    xmlns:SOAP-ENC="http://schemas.xmlspoap.org/soap/encoding/"
    xmlns:mime="http//schemas.xmlspoap.org/wsdl/mime/" targetNamespace='http://www.myapp.org'
    xmlns:chead="http://www.intersystems.com/SOAPheaders"
    <types>
     <s:schema elementFromDefault="qualified" targetNamespace="http://www.myapp.org"
       <s:element name="GetCustomerInfo">
          <s:complexType>
            <s:sequence>
               <s:element name="ID" type="s:decimal" minOccurs="0" />
            </s:sequence>
          <s:complexType>
          <s:element>
```

FIGURE 3.2 – Exemple d'un fichier WSDL.

3.6.2 USDL

Le langage de description du service unifié (USDL ; Unified Service Description Language) est utilisé pour décrire les paramètres commerciaux, opérationnels et techniques des services. Les descriptions de service incluent des informations telles que le prix, le service juridique, le fournisseur de services, les méthodes d'interaction, les accords de niveau de service [57].

```
<http://linked-usdl.org/ns/usdl> a owl:Ontology;
  dc:title "Linked-USDL Core";
  dc:description """<p>This vocabulary provides ...most of the original
USDL specification with some useful simplifications.""";
  dc:modified "2012-09-20"^^xsd:date;
  vann:preferredNamespaceUri "http://www.linked-usdl.org/ns/usdl#";
  vann:preferredNamespacePrefix "usdl";
  foaf:page <http://linked-usdl.org/ns/usdl.html>;
  dc:creator
    <http://linked-usdl.org/ns/usdl#cpedrinaci>,
    <http://linked-usdl.org/ns/usdl#jcardoso>,
    <http://linked-usdl.org/ns/usdl#tleidig> .
```

FIGURE 3.3 – Exemple d'un fichier USDL.

3.6.3 RDF

C'est le cadre de description des ressources (Ressource Description Framework) est proposé comme une méthode générale pour la description conceptuelle ou la modélisation de service Web à travers des graphes basé sur les modèles de données [20]. De plus, il est représenté comme une méthode générale pour décomposer n'importe quel type de connaissance en petit morceaux. Le graphe d'un *RDF* contient trois composants principales qui sont : l'objet, l'attribut et la valeur [45].

3.6.4 DAML-S

C'est un langage de description de service en utilisant l'aspect sémantique à travers des ontologies. DAML-S offre des moyens afin de créer une description d'un service web qui va être interprété par un programme. DAML-S est représenté comme les autres langages de description. De plus, il facilite la description sémantique de services, leurs interfaces et leurs comportements [60].

3.6.5 OWL-S

OWL-S fournit aux fournisseurs de services Web un ensemble essentiel de constructions de langage de balisage pour décrire les propriétés et les capacités de leurs services Web sous une forme non-ambiguë et intelligente. Le balisage OWL-S des services Web facilitera l'automatisation des tâches du service Web, y compris la détection, l'exécution, l'interopérabilité, la composition et la surveillance de l'exécution automatisée du service Web. Suite à l'approche en couches du développement du langage de balisage, la version actuelle d'OWL-S s'appuie sur OWL [19].

3.6.6 WSMO

Web Service Modeling Ontology (WSMO) définit les principaux aspects liés aux services web sémantiques tels que : les ontologies, les services Web, les buts et les médiateurs. Les ontologies WSMO sont exprimées à l'aide du langage de modélisation des services Web (WSML), qui se base sur les différents formalismes logiques : logique de description, logique de premier ordre et la programmation logique [1].

3.6.7 WSDL-S

Ce langage est une extension du WSDL traditionnel : WSDL-S c'est un langage de description des services web qui ajoutes des aspects sémantiques [36]. De plus, l'aspect sémantique relié à ce type de fichier (WSDL-S) est contrôlé de l'extérieur d'un fichier WSDL de les éléments extensible des fichiers WSDL [117].

3.6.8 SAWSDL

Ce langage est définit comme un standard pour spécifier comment les liaisons de données du service Web peuvent être mises en correspondance avec les modèles formels. De plus, SAWSDL offre un moyen répétitif de connecter RDF ou OWL aux services Web sémantiques avec les liaisons de données fixes, facilitant la recherche par programme de données de service répondant aux besoins de l'application [111].

3.7 La qualité de services (QoS)

Afin de quantifier un service ou un service composite, il existe des mesures. Ces dernières sont présentées par la qualité de service. De plus, les QoS représentent l'aspect non-fonctionnel d'un service. Ces valeurs permettent la sélection des services pertinents aux demandes des utilisateurs. Dans cette section nous allons présenter quelques QoS.

3.7.1 Le coût

Il se réfère aux frais d'accès et d'utilisation d'un service que le demandeur de service doit payer. En outre, cela dépend du nombre de tâches qu'un utilisateur de service doit exécuter [71] [96].

3.7.2 Temps de réponse

Le plus important dans la délivrance de services est de fournir un service au consommateur dans un délai raisonnable. Le temps de réponse est mesuré en fonction de certains sous-facteurs tels que le temps de réponse moyen et le temps de réponse maximal promis par le fournisseur de services [61]. Le temps de réponse est calculé selon la formule suivante :

$$Temps\,de\,réponse = \sum_{i}^{N} T_i / n \tag{3.1}$$

T_i représente le temps où l'utilisateur i a demandé un service disponible et n est le nombre total de demandes de service.

3.7.3 Fiabilité

Il se réfère à la capacité du service à fonctionner correctement et de manière cohérente selon les conditions définies dans l'accord de niveau de service (SLA). Il est mesuré en termes de défaillances de transaction par année ou par mois [96][61][92]. La fiabilité est calculée selon l'équation (3.2) :

$$Fiabilité = \left(1 - \frac{Nombre\,d'\,échec}{n}\right) \times P_{mttf} \tag{3.2}$$

Nombre d'échec est le nombre d'utilisateurs qui ont subi un échec dans un intervalle de temps inférieur à celui prévu par le fournisseur du Cloud, n est le nombre d'utilisateur, Où P_{mttp} représente le temps d'échec estimé qui est défini par le fournisseur de services.

3.7.4 Réputation

La réputation est le niveau de confiance à partir duquel le service est accepté par les utilisateurs ; il mesure la confiance gagnée par le service en fonction d'expériences antérieures [71][96]. Il est calculé selon l'équation suivante :

$$Réputation = \frac{\sum_{r=1}^{N} F_r}{N} \tag{3.3}$$

F_r est le rang donné par le $r^{ième}$ utilisateur pour un service, N est le nombre d'utilisateurs qui ont classé le service.

3.7.5 Disponibilité

Cela représente la valeur qui mesure si le service est accessible aux utilisateurs ; il est défini par le pourcentage de temps que le client consomme pour accéder au service. En outre, cela représente la mesure dans laquelle le service est opérationnel [96][61][92][**?**]. La disponibilité est calculée par l'équation suivante :

$$Disponibilité = \frac{MTBF}{MTBF + MTTR} \tag{3.4}$$

MTBF se réfère au temps moyen de service avant l'état d'échec, et *MTTR* est le temps moyen de réparation.

3.7.6 Pertinence

Cela signifie le degré d'intersection entre les exigences du client et le fournisseur du service [61]. Pour calculer la pertinence, nous pouvons utiliser deux cas proposés dans [61] :

Le cas où nous avons plus d'un fournisseur de services à la fin du processus de découverte du service qui satisfont toutes les exigences essentielles et non essentielles du client.

$$Pertinence = \frac{Nombre\ de\ caractéristique\ non\_essentielles\ fournies\ par\ le\ service}{Nombre\ de\ caractéristique\ non\_essentielles\ requises\ par\ le\ client} \tag{3.5}$$

Si le filtrage aboutit à une liste de fournisseur de services vide, les opérateurs qui satisfont aux caractéristiques essentielles sont choisis.

$$Pertinence = \begin{cases} 1, & Si\ toutes\ les\ caractéristiques\ sont\ satisfaites \\ 0, & dans\ le\ cas\ inverse \end{cases} \tag{3.6}$$

3.8 Les plateformes de composition de services

La composition de services web est un ancien problème, les chercheurs proposent plusieurs méthodes et approches pour les résoudre. Et parmi ces méthodes, les entreprises proposent des plateformes ayant comme but de résoudre ce problème de composition de services web. Dans cette section, nous allons présenter quelques plateformes.

3.8.1 IBM Business Process Manager

C'est une plateforme de gestion des processus métier (BPM) qui offre des fonctionnalités complètes, prête pour une utilisation immédiate, conçue pour accélérer le déploiement. Ce logiciel permet aux clients d'utiliser les outils et l'environnement d'exécution appropriés pour la

conception, l'exécution, la surveillance et l'optimisation des processus métier [50].

3.8.2 Oracle BPEL Process Manager

C'est une plateforme qui offre une infrastructure complète et facile à utiliser pour créer, déployer et gérer les processus métier BPEL. De plus, cette plateforme permet la conception et le déploiement d'une orchestration axée sur les métadonnées en utilisant des outils de développement et de gestion simplifiés et productifs. Il exécute aussi des déploiements stratégiques avec une grande disponibilité et une fiabilité [51].

3.8.3 Microsoft BizTalk Server

BizTalk Server permet de connecter différents logiciels, puis de créer graphiquement et modifier la logique de processus qui utilise ce logiciel. De plus, il permet aux agents de l'information de surveiller les processus en cours, d'interagir avec les partenaires commerciaux et d'effectuer d'autres tâches commerciales [52].

3.8.4 SAP NetWeaver Process Integration

Cette plateforme s'adresse aux experts en développement et en intégration qui doivent développer et configurer des scénarios d'intégration. Cependant, il fournit également un point d'entrée à la documentation pour les administrateurs qui souhaitent opérer et surveiller les scénarios d'intégration [53].

3.8.5 ActiveVOS

ActiveVOS est une plateforme d'automatisation des processus orientés services spécialement conçue pour répondre aux besoins des membres des équipes de projet informatique. Avec cette plateforme, on peut créer rapidement des modèles de processus conformes BPMN2.0 qui intègrent parfaitement les personnes, les processus et les systèmes, ce qui augmente l'efficacité et la visibilité de l'entreprise [54].

3.8.6 Apache ODE

Le logiciel Apache ODE (Orchestration Director Engine) exécute les processus métier écrits selon la norme WS-BPEL. Il concerne des services Web, de l'envoi et de la réception de messages,

en manipulant les données et le recouvrement des erreurs, comme décrit dans la définition du processus. Il prend en charge les exécutions de processus vivantes longues et courtes pour organiser tous les services qui font partie de l'application [55].

3.8.7 JBoss jBPM

Cette plateforme est une suite flexible de Business Process Management (BPM). Il constitue le pont entre les analystes commerciaux et les développeurs. Les moteurs traditionnels de BPM ont une portée limitée aux personnes non techniques seulement. L'objectif de JBPM est d'offrir des fonctionnalités de gestion de processus de manière à ce que les utilisateurs professionnels et les développeurs l'apprécie [56].
Tous les derniers plateformes proposent une solution pour le problème de composition, néanmoins, l'absence d'un modèle de description va affecter le processus en terme de QoS. L'intégration de l'aspect agent peut améliorer la tâche de composition lorsqu'on parle de l'auto-adaptation.

3.9 Les caractéristiques des plateformes de composition de services

Après la présentation de la liste de quelques plateformes de composition de services web, dans cette section, nous allons présenter les caractéristiques nécessaires pour une plateforme de composition. Ces derniers seront cités comme suit d'après le travail de [110] :

3.9.1 Le standard ouvert supporté

Le standard ouvert supporté est la condition préalable de base pour une interopérabilité et une compatibilité élevées avec lequel une plateforme de composition de services peut choisir des services fournis par des tiers pour obtenir une plus grande réutilisation, cela consiste aussi à collaborer avec des partenaires pour améliorer la portabilité des données. L'information peut être échangée de manière directe.

3.9.2 La facilité d'utilisation

Les plateformes de composition des services devraient être faciles à utiliser, ce qui signifie que des outils essentiels sont nécessaires pour réduire la complexité et améliorer l'efficacité du développement de la composition des services. Par exemple, le processus du service composite pourrait être modélisé plus facilement à l'aide d'un éditeur de glisser-déposer.

3.9.3 La simulation

La simulation est un moyen de créer un service composite virtuel afin d'évaluer ses performances avant de l'exécuter dans un environnement réel. Cela permet aux développeurs de vérifier dynamiquement les flux de processus, de valider le modèle et de recueillir des informations de synchronisation et de ressource sur le service composite en vue de l'amélioration des processus métier.

3.9.4 L'administration et suivi

L'administration de l'exécution, y compris le démarrage, la suspension, la reprise et la fin des instances du service composite, devrait être prise en charge par une plateforme de composition de services. La surveillance comprend le suivi des états d'exécution des cas et l'analyse des caractéristiques de performance pour l'amélioration des processus.

3.9.5 L'adaptabilité

Pour assurer l'aspect dynamique de l'environnement orienté service, une plateforme de composition de services devrait pouvoir apporter une adaptabilité aux services composites, tels que la liaison de service de façon dynamique, la gestion de la version du processus, la mise à jour de la définition du processus et la mise à jour de l'instance. La modification du processus de service composite nécessite une mise à jour dynamique sans suspendre ou mettre fin aux instances d'exécution connexes.

3.9.6 Optimisation

L'optimisation est un moyen important pour réduire les coûts et de donner une valeur commerciale considérable. L'objectif de l'optimisation est de récupérer les informations sur les performances des processus à partir de la phase de modélisation ou de surveillance, promouvoir

l'utilisation à grande capacité des services en réduisant les problèmes potentiels ou réels, pour améliorer l'efficacité en organisant les activités dans le meilleur ordre, et d'identifier les opportunités économiques potentielles de coûts ou d'autres améliorations.

3.9.7 Sécurité

Il est important d'assurer la sécurité dans les applications de service Web. Les techniques de sécurité de la composition des services comprennent la protection des messages par l'intégrité du message, le cryptage et l'authentification, et la prévention de l'accès non autorisé des services par le biais de politiques de contrôle d'accès.

3.10 Conclusion

La rapidité d'évolution des technologies orientées service offre des méthodes qui permettent aux utilisateurs de les exploitées de manière plus en plus facile. Afin de mettre que le choix soit facile entre les technologies existantes, ce chapitre représente une comparaison entre eux. De plus, il présente les modèles de description de services qui représentent le noyau de n'importe quel type de technologie. En outre, afin de choisir un service, nous allons citer quelques qualités de service qui vont évaluer le service final.

De point de vue d'un concepteur, on ne peut pas dire qu'une technologie est meilleure par rapport à une autre, cela dépend des problèmes rencontrés et selon les besoins des utilisateurs.

Dans le prochain chapitre, nous allons terminer la partie de l'état de l'art par la présentation de quelques travaux en relation avec le problème posé. De plus, nous allons présenter les différentes approches utilisées pour résoudre le problème de composition de services dans les deux cas ; dans ou hors du Cloud.

Chapitre 4

Travaux connexes et synthèses bibliographiques

4.1 Introduction

D'après les avantages et les offres fournis et proposés par les fournisseurs de service dans le Cloud Computing, outre, l'énorme service déployer dans cette technologie encourage les utilisateurs tels qu'un simple client ou une entreprise de l'utiliser. Bien qu'il existe plusieurs services qui peuvent donner à l'utilisateur final la solution qu'il veut, et n'empêche pas de mettre ce dernier confus lors de la sélection du meilleur service ou meilleur service composite, les services publiés peuvent donner des résultats de façon fonctionnelle sans prendre en considération l'aspect non-fonctionnels. En outre, les requêtes des clients sont complexes car ils peuvent être composés par différents services, dans ce cas nous avons besoin d'une collection de service qui répond aux besoins du client. Afin de résoudre ce problème, nous avons besoin d'une composition de services parmi les services existants en prenant en compte l'aspect fonctionnel et non-fonctionnels lors du processus de composition.

Après avoir présenté les différences entre les technologies dans le chapitre précédent, le présent chapitre donne une synthèse bibliographique sur les travaux réalisés pour résoudre le problème posé.Dans ce chapitre nous présentons les méthodes utilisées pour résoudre le problème de composition de services dans et hors du Cloud.

4.2 Présentation du problème de composition de services

La diversité et le nombre important de service déployé dans le Cloud rendent les clients confus en posant la question suivante : « quelles sont les services qui peuvent répondre à mes besoins ». Le développement rapide de la nouvelle technologie (Cloud Computing) offre des solutions qui rendent les clients satisfaisant à cause des avantages économiques et technologiques [97]. Lorsqu'un service atomique ne sert pas les clients finaux à cause de la complexité de ça requêtes et l'absence d'un modèle unifié pour la description de services [105], la distribution des services dans les différentes localisations géographiques [106], force les chercheurs à trouver des solutions à ce genre du problème. Parmi les solutions, nous avons la combinaison d'un ensemble de services afin de répondre à ces besoins de façon raisonnable en terme de qualité de service (temps de réponse acceptable et d'un coût réduit) [**?**]. Il existe plusieurs travaux réalisés pour résoudre ce problème. Ce chapitre, nous allons le diviser en deux sections (dans et hors le Cloud), et chaque section comporte un ensemble de catégories.

4.3 Les méthodes de composition de services hors le Cloud Computing

Plusieurs travaux sont réalisés pour résoudre le problème de composition de services. Cette problématique est ancienne avant l'arrivée du Cloud Computing. Dans cette section, nous avons collecté quelques travaux dans la littérature qui voulaient résoudre ce genre du problème. Nous avons divisé cette section en quatre catégories représentées comme suit :

4.3.1 Composition de services semi-automatiques

eFlow [35] est l'une des premières plateformes conçues pour spécifier et surveiller les services composites. Cette plateforme offre aussi de nombreuses caractéristiques pour l'aide au provisionnement adaptatif de services dans les environnements dynamiques. eFlow offre également certaines opérations pour les utiliser dans le processus de composition. Ces opérations sont la sélection dynamique de services, sélection de conversation dynamique et génération des nœuds.

Self-Serv (compoSing wEb accessibLe inFormation and buSiness sERVices) [131] Ce Frame-

work propose une solution pour le problème de composition de services web. Cette solution est basée sur l'approvisionnement de bout en bout dynamique de composition de services web. Dans ce Framework, les services sont composés et les services composites sont exécutés de façon décentralisée. Self-Serv distingue trois types de services : services élémentaires, services composites et services communautaires. Dans cette proposition, un service est associé à un coordinateur qui s'occupe de l'opération de contrôle et le suivi de l'exécution de service correspondant. En outre, Self-Serv prend en considération la qualité de service afin de spécifier les aspects non-fonctionnels de service web [129]. Cette approche assure que la qualité de service reste optimale dans l'exécution de service délivré (service composite).

WISE [3] est un projet ayant la volonté de fournir une application pour le processus basé sur business-to-business dans le domaine du commerce électronique. WISE possède un environnement de développement et un composant d'exécution. Ce dernier est organisé en trois couches : service de base de données, service de traitement et service des interfaces. Cette plateforme offre aux utilisateurs la spécificité du traitement via un outil formel nommé StructWare.

ServiceGlobe [84] est une plateforme de service extensible et distribuée qui offre de nouvelles technologies pour le déploiement et l'exécution des services web dans les environnements dynamiques. Afin de déployer des services, cette plateforme propose : la sélection dynamique de services et un dispatching du service. En outre, cette plateforme utilise le contexte pour faciliter le déploiement des services.

Orriëns et al. dans[9] les auteurs présentent le BCDF (Business Collaboration Development Framework), qui est un framework basé sur des règles de business pour la composition de services où le processus de composition est divisé en quatre phases. Ces phases sont : définition abstraite, ordonnancement, construction et exécution. Ce framework est composé d'un gestionnaire de composition de services qui est responsable de toute opération de composition. De plus, il existe un répertoire de composition de services qui facilite le maintien des éléments de composition et des règles qui sont utilisés pour la composition.

Berardi et al. [26] proposent une technique permettant l'implémentation d'une composition de services et le traitement des évènements qui peuvent se produire lors du processus de composition. De plus, la composition de services peut être synthétiser en temps réel « just-in-time ». Cette méthode était conçue pour répondre au problème de vérification de l'existence de

relation de simulation entre la cible et les composants de service vérifié.

SOA4All (Service Oriented Architectures for All) [76] est un projet européen. Ses objectifs sont : la fourniture de l'énergie, la flexibilité et la simplicité nécessaire aux services technologiques orientés. De plus, ce projet considère plusieurs opérations comprenant l'aspect sémantique dans le processus de composition de service.

METEOR-S (Managing End To End OpeRation for semantic Web) [66] est un système de gestion du flux de travail basé sur le modèle formel, l'ordonnancement central et distribué et l'exécution du flux de travail. L'architecture proposée dans ce système est composée de trois parties : la spécification et l'annotation, la configuration dynamique et l'adaptation de processus. La troisième partie traite le problème d'adaptation des services composites aux événements et aux défauts d'exécution.

El Haddad et al. [67] proposent une méthode de sélection transactionnelle basée sur la qualité de service afin de résoudre le problème de composition de services. En outre, le travail prend en considération cinq qualités de service (le cout d'exécution, le temps d'exécution, réputation, taux d'exécution réussi et la disponibilité). Dans l'étape de sélection, les auteurs utilisent une méthode d'optimisation locale.

Discorso (Distributed Information System for Coordinated Service-Oriented Interoperability) [27] est un framework qui fournit une solution compréhensive afin de spécifier et gérer une composition de services flexible. De plus, ce framework offre un ensemble d'outils pour aider la spécification des informations requises des services composites en temps réel. La méthode de sélection utilisée dans cette approche est basée sur le modèle de programmation linéaire. En outre, Discorso donne des règles de surveillance pour surveiller l'execution de services ainsi le déclenchement des actions correctives si c'est nécessaire.

Michlmayr et al. [5] proposent un environnement nommé VRESCo (Vienna Runtime Environment for Service-Oriented Computing) pour la composition de services web. Il consiste en un serveur d'application composé de : moteur de requête, moteur de notification, service de publication/métadonnées, service de gestion, et moteur de composition. VRESCo permet la découverte et la composition de services en se basant sur la qualité de service.

Fujii et al. [73] proposent un framework de composition de service contextuelle permettant la composition des applications en langage naturel. En outre, ce framework est composé de mo-

dèle de service de composants à base sémantique, environnement d'exécution des composants et composition de services basé sur des graphes sémantiques.

SeSCo (Seamless Services Composition) [119] ce framework fournit un mécanisme de composition de services dans des environnements omniprésents. SeSCo utilise une plateforme middleware orientée événement nommée Pervasive Information Communities Organization (PSCO) afin de réaliser les interactions entre les services de manière efficace. Le mécanisme de composition de services dans ce framework modélise les services comme des graphes orientés attribués. De plus, un on trouve mécanisme hiérarchique de composition de services basé sur un recouvrement de dispositif formé à travers le protocole de verrouillage. Ce dernier fournit un soutien essentiel aux services pour les périphériques à ressources limitées.

Yu et al. [65] proposent une approche nommé PerCAS (Personalized Context-aware Services) pour implémenter une composition de services personnalisé qui pourrait s'adapter aux exigences spécifiques de l'utilisateur. En outre, ce travail utilise un langage de règles, ces règles peuvent s'adapter en cours d'exécution. Pour exécuter les services, cet environnement intègre moteur BPEL et un moteur de règles pour l'exécution de service.

4.3.2 Composition de services automatiques

Fusion [30] est un framework pour un portail de services qui permet de spécifier les services des clients et d'optimiser l'exécution des plans de services de façon automatique. De plus, ce framework contient six parties : spécification des utilisateurs, générateur des plans dynamiques pour les services web, exécution des plans, vérification, récupération et un générateur de réponse à l'utilisateur.

SWORD [108] offre un ensemble des outils qui permet aux développeurs de composer des services web existants afin de fournir un nouveau service composite. SWORD n'utilise pas les nouvelles normes standards comme WSDL et OWL-S, il utilise le modèle relation-entité [64] pour spécifier les services Web. Cette solution modélise chaque service par des entrées et des sorties qui consistent en une entité et des relations entre les entités. Afin de créer un service composite, le développeur a besoin de spécifier les états initiaux et finals de service composite.

McIlraith et al. [109] présentent une approche qui adresse la résolution de problème de composition automatique de services web avec l'exécution de web sémantique. En outre, les auteurs

adaptent un langage de programmation logique (GOLOG) pour offrir le traitement générique, personnalisé, et utilisable dans le contexte de service web. Dans cette solution, la composition de services Web est conçue comme un problème de planification, de façon que le service composite est représenté par un ensemble de composants connecté par le GOLOG.

Sirin et al. [34] adoptent une planification du réseau de tâches hiérarchiques pour adresser le problème de composition automatique de services web. Les auteurs pensent à traduire les spécifications OWL-S dans cette approche, ensuite la soumettre dans le planificateur SHOP2 afin de construire un service composite. De plus, un convertisseur de plan est construit pour convertir les plans de SHOP2 dans OWL-S pour l'exécution en tant qu'un processus d'OWL-S.

McDermott et al [32], dans ce travail, ont proposé une extension de planification de la langue de définition de domaine pour formaliser les services web. Les auteurs ont appliqué une planification technique afin d'implémenter la composition de services web. Dans cette approche, les descriptions (DAML-S) sont traduites afin de les utiliser dans la composition de services web. Le problème de composition est vu comme un problème de planification.

Kona et al. [114] utilisent un langage de description sémantique de services unifiés (USDL) pour la spécification formelle sémantique de services. Ce dernier, utilise WordNet pour comprendre les sens de services. Le service désireux est traduit en OWL-S à partir d'une présentation basée sur les graphes acycliques. Le flux de composition comprend des structures séquentielles, non-séquentielles et conditionnelles.

Les auteurs de Mehandjiev et al. [93] ont proposé une méthode de composition de services coopérative pour la synthèse automatisée des services composites. Les fournisseurs de services sont proactifs dans la synthèse des services composites. Le demandeur de service annonce la nécessité d'un service composite en spécifiant ses états de début et de fin sur un tableau d'affichage spécialisé. Chaque agent fournisseur offre une proposition partielle pour le service composite. Quand le service est fourni, il est servi à l'agent demandeur. Ce travail prend en considération la qualité de service ainsi que la qualité sémantique.

4.3.3 Les méthodes de composition non-heuristiques

Yu. et al. [147][148] ont proposé deux méthodes pour le problème de sélection de service. Le but de ce travail est de minimiser la fonction utilitaire pour assurer la satisfaction des contraintes

rencontrées. Ces contraintes sont les besoins non-fonctionnels de clients qui vont être servis à la fin du processus de composition. Il existe plusieurs formes pour le modèle du processus de business tel que séquentiel, parallèle et en boucle, etc. De plus, les auteurs proposent un algorithme séquentiel pour l'utiliser dans le cas d'une seule contrainte pour la composition de service. Ce travail prend en considération les qualités de services suivantes : temps de réponse, le coût, fiabilité, et la disponibilité. Les auteurs ont utilisé une méthode de sélection qui modélise le problème comme Multi-Choice Knapsack Problem (MCKP).

La deuxième approche [148] est présentée par la théorie des graphes. Les auteurs ont modélisé le problème de composition par un problème de plus court chemin. Dans cette solution, chaque service candidat dans chaque classe de service représente un nœud. L'idée est de transférer les paramètres de QoS depuis le nœud de son arc correspondant pour construire les contraintes de graphe acyclique orienté. Les auteurs résolvaient le problème posé en utilisant les algorithmes suivants : Contraint Bellman-Ford et le plus court chemin.

Zeng et al. [153] ont présenté une plateforme intitulée agFlow qui permet de résoudre le problème de composition de services Web à base de qualité de service. Dans cette solution, les qualités de service sont évaluées. Ensuite, il y'a la sélection de services qui peuvent optimiser les QoS. Les QoS considérées dans ce travail sont le cout d'exécution, le temps d'exécution, la réputation, le rang d'exécution réussie, et la disponibilité. De plus, les auteurs ont proposé deux approches pour la sélection de services dans le processus de composition qui sont la localisation locale et la planification globale.

Dans [149] les auteurs proposent une méthode de sélection et de composition de services en cours d'exécution. Pour résoudre le problème, les auteurs ont présenté une méthode basée du contexte pour la sélection de services intitulée (BCCbSS). L'idée générale de cette solution est que l'algorithme passe par le processus et sélectionne les services étape par étape. Après la sélection de services dans chaque étape, l'algorithme fait un retour en arrière pour confirmer si les services sélectionnés sont les meilleurs pour la composition ou non, si oui, il est invoque. Malheureusement, cet algorithme fonctionne seulement dans le cas séquentiel, il ne peut pas trouver une composition optimale.

4.3.4 Les méthodes de composition méta-heuristiques

Dans [12], les auteurs proposent une méthode heuristique pour résoudre le problème de composition. Dans ce travail, les auteurs utilisent une fonction d'agrégation afin de calculer l'utilité de l'ensemble du service composite. L'idée est d'utiliser un algorithme sous forme séquentielle. De plus, cette approche prend en considération l'aspect non-fonctionnel. Le problème présenté dans ce travail est résolu par l'utilisation du simplex avec l'algorithme de backtraking pour trouver la solution.

Le travail représenté dans [16] les auteurs ont utilisé les algorithmes génétiques pour résoudre le problème de composition. La fonction fitness est utilisée pour comparer les résultats. De plus, la diversité de service en terme de QoS conduit au problème général d'optimisation de la sélection des services Web pour chaque tâche, de sorte que la QoS globale et les exigences de coût de la composition sont satisfaites. Cette approche prend en considération les qualités de services .

Jiuyun et al. [142] ont utilisé l'algorithme immunitaire pour résoudre le problème de composition. L'idée de cette méthode est de coder le problème dans un anticorps. En outre, cette proposition est basée sur deux opérations qui sont : la sélection immunitaire de service et la sélection clonale. Dans l'étape de la sélection immunitaire, les anticorps sont reproduits et supprimés afin de contrôler leur densité dans l'espace de travail. De plus, l'étape précédente assure aussi que les meilleurs éléments ne seront pas détruits. Dans la deuxième opération, l'algorithme utilise les anticorps ayant le meilleur fitness comme une information heuristique afin d'accélérer la convergence. Ils ont pris en considération les qualités de service dans la fonction fitness.

Dans [145], les auteurs proposent un algorithme intitulé ACAGA_WSC. Cet algorithme est composé de deux algorithmes, le plus connue dans les domaines méta-heuristique est celui de colonie de fourmi et les algorithmes génétiques pour résoudre le problème de composition de services. De même, les auteurs modélisent la composition de services par un algorithme de colonie de fourmi, pour des raisons dans des cas où les expérimentations de cet algorithme n'est pas fiable, Yang et al. ont introduit les algorithmes génétiques pour régler les paramètres de l'algorithme de fourmi afin d'obtenir de bons résultats.

Chen et al. [90] ont résolu le problème de composition de services en utilisant l'algorithme

d'optimisation des essaims de particules discrètes. Dans cette méthode, les auteurs considèrent chaque particule comme une solution. Chaque particule a une position et une vitesse, ces particules essayent de changer leurs positions à la base de deux éléments : (i) la meilleure dernière position visitée et (ii) la meilleure position qu'était vue jusqu'à cette position.

4.3.5 Les méthodes de composition à base d'agent

Dans le travail présenté en [94], les auteurs ont proposé un modèle qui utilise le système multi-agents pour résoudre le problème de composition de services web. Les auteurs utilisent les agents comme étant un service web afin d'introduire l'aspect dynamique dans le processus de composition à travers la communication entre les agents. En outre, dans ce travail, les auteurs prennent en considération l'aspect sémantique. Malheureusement, l'aspect non-fonctionnel n'ai pas traité dans cette proposition.

Les auteurs de [7] ont présenté une approche pour résoudre le problème de composition de services de façon automatique. De plus, l'idée proposée dans ce travail est basée sur les agents et intègre les deux points suivants : 1) auto-organisation des agents en graphe de dépendance qui comporte un nombre d'agent représenté par un réseau social d'agents, et 2) un protocole de coopération entre les agents pour la composition optimale de services web. Les auteurs prennent en considération l'aspect non-fonctionnel.

Le travail présenté dans [123] Tong et al. ont proposé un algorithme distribué pour la composition de services web à base des agents. L'idée proposée est d'intégrer les deux technologies service web et l'aspect agent dans une seule entité cohésive. C'est ainsi que, l'algorithme proposé intitulé DPAWSC transforme le problème de composition en un problème de recherche dans un graphe à partir des relations de dépendances entre les agents de service. L'aspect fonctionnel est tenu en compte de ce travail, mais n'a pas mentionné le cas des changements des paramètres de services durant le processus de composition de services.

Dans Qian et al.[130] ont proposé un modèle de composition de services web basé sur les agents mobiles. Le modèle présenté évite le problème de transfert de données par l'approche de peer-to-peer. Dans ce modèle, les clients saisissent la spécification et les paramètres initiaux à travers le portail de composition de services. Ensuite, le système crée des agents mobiles afin de réaliser la tâche de composition et renvoyer les résultats aux clients. De plus, les auteurs

ont défini un planning afin de spécifier les actions logiques de l'agent mobile et exécutent le processus de composition de services de façon automatiques. L'aspect non-fonctionnel n'a pas été traité dans cette approche.

Dans le travail présenté en [77], les auteurs ont proposé une approche de composition de services web par l'utilisation des agents autonomes. Les auteurs ont proposé une extension de Contract Net Protocol (CNP) nommé Agent-centric Contract Net Protocol (ACNP) comme un mécanisme de négociation afin de composer les services web. De plus, la méthode proposée comporte les taches suivantes : (1) un mécanisme de matching est intégré dans un agent ; nommé agent intermédiaire ; pour la découverte de services, (2) un algorithme de sélection intégré dans un agent (agent composeur), et (3) mécanisme de négociation entre deux agents (agent contracteur et gestionnaire). L'aspect non-fonctionnel n'est pas traité dans ce travail.

4.4 Le problème de composition de services dans le Cloud

Avec l'arrivée de la technologie du Cloud Computing et les solutions proposées par la majorité des fournisseurs dans le monde comme Amazon Web Service (AWS) et Google [24], les avantages de cette technologie encouragent les fournisseurs de déployer leurs services. La question toujours posée est : « Comment servir les consommateurs (utilisateur final) dans un délai raisonnable avec des qualités de services qui satisfassent les besoins des clients ? ». A cause de la diversité des services publiés dans le Cloud [89] [61], et la complexité des requêtes du client (requête complexe) cela mènera à un problème de sélection et de composition de services. Dans cette section, nous allons présenter quelques travaux qui tente de résoudre ce problème en utilisant des différentes catégories cité dans [69] et présentés comme suit :

4.4.1 Les algorithmes à base des graphes

Kofler et al. [74] proposent une approche basée sur l'algorithme de Branch and Bound en utilisant le parallélisme afin de maximiser les qualités de services lors du processus de composition de service. L'approche proposée utilise une méthode mathématique basée sur le problème de Knapsack multi démentions et multi choix afin de déterminer les mesures de satisfactions de clients.

Le travail présenté dans [152] introduit un algorithme de matching pour les différents ser-

vices Cloud avec les paramètres entrée/sortie en utilisant l'aspect sémantique. Ce dernier utilise la similarité entre les concepts à base du WordNet. De plus, ce travail propose un algorithme nommé Fast-EP et FastB+-EP prouvé pour résoudre le problème de composition.

Les auteurs de [49] proposent une méthode de composition de services pour manipuler la plateforme du vidéo de surveillance dans le Cloud. En outre, cette plateforme utilise une stratégie d'allocation de ressources pour la composition de services. De plus, dans cette proposition, Houssain et al. utilisent la programmation linéaire afin d'optimiser les QoS.

Worm et al. [139] proposent un algorithme pour résoudre le problème de composition de services dans deux cas ; statique et dynamique. L'algorithme proposé dans ce travail est basé sur les qualités de services pour sélectionner les meilleurs services parmi les services disponibles.

4.4.2 Les algorithmes combinatoires

Dans le travail de Jula et al. [79], les auteurs adressent la préparation de services appropriés pour les clients finals. L'idée de cette approche permet de résoudre le problème de composition de services dans le Cloud. De plus, la méthode proposée essaie de résoudre le problème posé par l'optimisation de composition de services par l'utilisation d'une méthode hybride. Ce travail prend en considération l'aspect non-fonctionnel dans le problème de composition.

Ye et al. [68] offrent une solution afin de résoudre le problème de composition de services dans le Cloud. De même, le problème posé est résolu par l'utilisation des algorithmes génétiques. Les auteurs ont proposé un modèle basé sur les qualités de services afin de résoudre le problème de composition de services.

Dans le travail mentionné dans [155], l'auteur propose un modèle de sélection de service basé sur les algorithmes génétiques afin de résoudre le problème de composition par la sélection des flux de travail. De plus, dans ce travail, Ludwig transforme le problème à un autre d'optimisation en utilisant les qualités de services.

Zhao et al. [146] présentent une nouvelle méthode permettant d'assister la résolution du problème de composition par la sélection de services. Nous notons que ce travail utilise la méthode intitulé l'algorithme de sélection négative et immunitaire. L'algorithme proposé essaie de trouver une solution optimale avec des mesures de QoS multiples pour résoudre le problème.

4.4.3 Les algorithmes à base des machines

Les chercheurs dans [38] ont réalisé une machine à état fini en tenant en compte des corrélations de service et des séquences d'exécution de tâches. Chaque machine est utilisée pour le but d'implémenter un ensemble de services ayant des limites en termes de séquence d'exécution (CWS) en ajoutant une autre machine pour le service composite souhaité appeler dans le processus cible. De plus, la méthode proposée est composée de deux phases. La première phase consiste à créer un graphe pour la composition de services basée sur les CWS et le processus ciblé. La deuxième phase consiste à calculer les valeurs de QoS pour chaque chemin dans le graphe. A la fin, la plus haute valeur de QoS sera la solution finale.

4.4.4 Les méthodes basées sur les Framework

Les auteurs de [128] ont proposé un framework pour la composition de services de tel façon à ce que dernier utilise un agent de composition responsable à la réception des requêtes et fournit une gestion de service. L'agent fait l'analyse de chaque requête et la décompose en des services atomiques. De plus, ce framework utilise une base de connaissances pour stocker les identifications et les informations de chaque service. Ce Framework utilise un moteur pour générer des compositions en utilisant des services existant et il enregistre les résultats dans le catalogue de service.

Dans le travail présenté en Chunqing et al [21], les auteurs ont proposé un framework systématique afin d'automatiser la détection des conflits des services et de fournir des politiques sur les demandes des clients. Pour déterminer le meilleur service composite, les auteurs ont utilisé l'algorithme de retour en arrière.

Dans [141], les auteurs ont présenté un framework pour résoudre le problème de composition de services dans le Cloud. Le travail consiste à définir un ensemble de tâches dans ce framework afin de résoudre le problème posé avec l'introduction du facteur de confiance dans la sélection de service, le processus de composition et lors de génération des plans. De plus, les auteurs introduisent les qualités de services lors du processus de composition de services.

« CloudRecommender » est un framework proposé par Zhang et al [154] qui représente un système de composition de services sur le Cloud. De plus, ce framework est composé de trois couches : i) la gestion de configuration ii) la logique d'application et iii) l'interface graphique.

Dans ce travail, les auteurs utilisent les ontologies de service ainsi que les ontologies de QoS.

4.4.5 Les méthodes de composition à base d'agent

Gutierrez-Garcia et Sim [40] ont proposé une approche basée sur les agents afin de résoudre le problème de composition de services sur le Cloud. Le système multi agents introduit dans ce travail est composé par quatre types d'agent : agent client, agent fournisseur, agent ressource, et agent broker. En outre, les auteurs utilisent un modèle formel (réseau de Petri et réseau de Petri coloré) pour présenter le comportement de l'agent. De plus, les auteurs ont utilisé les services web comme une interface pour manipuler les ressources Cloud. L'aspect contextuel est tenu en compte dans cette approche par l'utilisation des ontologies. Dans ce travail, les auteurs ne prennent pas en considérations l'aspect non-fonctionnel.

L'auteur de [113] propose une approche basée sur les agents dans le Cloud Computing. L'idée de cette approche est d'introduire une architecture qui comporte la résolution des problèmes suivants : i) proposition d'un Cloudlet pour la découverte de services Cloud, ii) indiqué que les agents peut être adopter pour produire un protocole de négociation efficace dans le processus de sélection de service Cloud, et iii) de montrer que les agents peuvent utiliser des techniques de coopération afin d'automatiser le problème de composition de services. Ce travail ne prend pas en compte les qualités de services dans le processus de sélection et de composition de services.

Garcia Coria et al.[23] ont présenté une architecture basée sur les agents dans des organisations virtuelles pour la composition de services en utilisant le standard BPEL (Business Process Execution Language). Dans ce travail, les auteurs introduisent une méthode afin de faciliter la réutilisation semi-automatique des services web sur le Cloud Computing. Les auteurs ont intégré quelque composants dans l'architecture proposée pour maintenir la composition automatique de services web. En outre, le système multi agents est le responsable de l'opération de découverte et la composition de services web à travers des opérations qui sont : analyse, recherche et la composition. Dans ce travail, les auteurs ne mentionnent pas l'aspect non-fonctionnel.

Dans le travail présenté en [41], les auteurs ont proposé une architecture basée sur les agents afin de résoudre le problème de composition de services Cloud. En outre, cette approche présente une composition de services auto-organisée en utilisant les protocoles et les techniques

de coopération entres les agents. De plus, les auteurs ont tenu en compte les informations incomplètes sur les participants du Cloud et sa combinaison avec des mécanismes dynamiques de sélection de services. Ici, les qualités de services ne sont pas tenues en compte dans le processus de composition de services.

Tous les derniers travaux ne traitent pas le cas du changement des paramètres durant l'opération de composition de services.

Le tableau suivant montre les qualités de services utilisés dans notre approche avec quelques travaux existants dans la littérature.

TABLE 4.1 – Les différents travaux proposés pour résoudre le problème de composition de services.

QoS Travaux	Coût	Temps de réponse	Fiabilité	Réputation	Disponibilité	Pertinence
kofler et al. (2009)[74]	✓	✓	x	x	x	x
Zeng et al. (2009)[152]	✓	✓	✓	x	x	x
Ye et al. (2011)[146]	✓	✓	x	✓	✓	x
Hossain et al. (2012)[49]	✓	✓	x	x	x	x
Worm et al. (2012)[139]	✓	✓	x	x	✓	x
Ludwing (2012)[79]	✓	✓	✓	x	✓	x
Bao et Dou (2012)[38]	✓	✓	✓	✓	✓	x
Xiaona et al. (2012)[141]	x	x	x	x	✓	x

Jula et al. (2013)[68]	✓	✓	x	x	x	x
Zhao et al. (2013)[155]	✓	✓	✓	x	✓	x
Liu et al. (2012)[78]	✓	x	x	x	x	x
Notre Approche	✓	✓	✓	✓	✓	✓

4.5 Conclusion

La composition de services présente un challenge pour les chercheurs à cause de la diversité de ces derniers. Une différence très importante entre les problèmes posés et les solutions proposées, en particulier nous signalons qu'il n'y a aucun modèle unifié et adopté par tous les fournisseurs de services pour décrire les services disponibles. En outre, le problème de localisation des services est présent dans tout les cas dans les différents serveurs ou centres de données. Ce problème rend les tâches de découverte et de composition d'un ensemble de services trop complexe pour produire les meilleurs résultats satisfaisant les utilisateurs finaux. En effet, le nombre important de services, ainsi que la diversité et le problème de composition et de sélection est classé comme NP-complet.

Comme nous avons vu, ce chapitre dresse une présentation avec une synthèse concernant les travaux réalisés pour résoudre le problème de composition de services. En outre, ce chapitre présente les solutions du problème dans la littérature à travers deux cas (dans le cas où la composition se fait dans un environnement Cloud, ou bien hors du *Cloud Computing*). Parmi les travaux réalisés, il existe quelque points qui ne sont pas tenus en compte lors de la résolution du problème de compositions tel que l'aspect non-fonctionnel manquons ou le cas où les fournisseurs de services introduisent quelque modifications au niveau de la description de service lors du processus de composition de services. Comme nous avons vu, dans ce chapitre, les agents peuvent apporter des améliorations dans le processus de composition.

Ainsi, dans le prochain chapitre, nous allons présenter notre contribution afin de résoudre le problème posé à travers la proposition d'une architecture à base d'agent avec le protocole utilisé. De plus, le prochain chapitre donnera une description de services Cloud pour le cas de déploiement des applications/services dans le Cloud avec la représentation d'un algorithme de composition proposé afin de répondre aux exigences des clients.

Deuxième partie

Contributions

Chapitre 5

Système de composition de services Web

5.1 Introduction

De nos jours, la composition de services représente l'un des problèmes majeurs dans le Cloud Computing, à cause du nombre énorme services déployés dans les centres de données. En effet, ces services sont sous forme atomiques et ne répondent pas à tous les demandes des clients. Comme déjà mentionné dans les chapitres précédents, la composition de service offre une solution à ce problème (la requête est complexe). Les méthodes de composition de services traditionnelles offrent des solutions (services composites) aux clients sans prendre en considération l'aspect non-fonctionnel. Afin de résoudre le problème de la composition, nous allons proposer une architecture qui offre une solution pour ce problème.

Après avoir présenté l'aspect théorique de notre projet et pour comprendre son fonctionnement, nous allons aborder notre contribution pour la composition dynamique des services web.

Dans ce chapitre, nous allons présenter un scénario qui montre l'utilité de cette solution. Après, nous allons proposer une architecture globale du système en montrant sa fonctionnement à l'aide d'un diagramme de séquence UML pour simplifier le déroulement de notre approche proposée. Ensuite, nous allons expliquer l'architecture détaillée de chaque composant (sous-système) et son fonctionnement, nous allons terminer ce chapitre par une conclusion.

5.2 Étude de cas

Dans ce chapitre, nous allons concentrer sur le problème du changement des paramètres dans le processus de composition de services. Pour bien comprendre ce problème, dans cette section, nous allons présenter un exemple qui explique l'étude de cas de notre projet. Au cours de la composition de services, le processus connu est de prendre la requête du client, ensuite, extraire les mots-clés pertinents qui indiquent à un service bien définie. Après l'étape de décomposition de la requête, le système utilise chaque mot qui représente un domaine (type de service) pour collecter les services candidats. Afin de combiner les services pour construire un service final (service composite), ce processus prend un délai de temps. Au cours de cette opération, un fournisseur veut changer quelque tâches (paramètres) dans le service déployé. Dans le cas où ce service est sélectionné comme un service composite, le service final donne un échec lors de l'exécution parce qu'il est modifié au cours du processus de composition. Afin de résoudre ce problème, dans notre contribution, nous allons exploiter le paradigme de SMA pour proposer une architecture pour éviter le problème mentionné dans cette section. Les SMAs ont la capacité de s'adapter avec les situations de changement de paramètres. De plus, les SMAs sont des collection d'agent cognitive et autonome, et c'est le cas de notre problème qu'est le nombre énorme de services dispatché dans le monde et la solution utilise la notion des agents mobiles pour i) collecter les informations pour sélectionner les services candidats, ii) à travers notre protocole de communication les services composites sont crées.

5.3 La composition adaptative de services

Lorsqu'on parle d'une composition adaptative de services, cela veut dire que les services doivent utiliser leurs caractéristiques pour réagir avec les différentes situations (changement des paramètres). Cependant, les approches traditionnelles proposent des services composites mais ces services ne sont pas adaptables. Ce problème d'adaptabilité de services retourne à la limite des services web, la raison principale de ce problème est le manque de l'aspect auto-adaptation, pour résoudre ce problème, nous allons utiliser les agents, car ce paradigme donne un solution ré-configurable si le système a tombé de le cas de changement de paramètres durant le processus de composition. De plus, l'utilisation de l'aspect agent est considéré comme

une solution efficace qui permet de présenter un service web. Donc, cet aspect nous permet de présenter le système multi-agents comme un service composite. Afin d'achever cette objective, nous allons proposer un protocole entre les agents pour accepter les autres agents candidats comme partenaire de composition et éliminer le conflit entre eux. Le tableau suivant donne une comparaison entre les deux concepts (agent et service web)[58].

TABLE 5.1 – Comparaison entre agent et service web.

Agent	**Service Web**
Un agent est autonome, a une connaissance sur lui-même et de ses accointances.	Un service web a des connaissances sur lui-même, mais pas sur ses utilisateurs, clients.
Les agents sont communicatifs.	Les services Web sont passifs jusqu'à ce qu'ils soient invoqués.
Les agents sont conçus pour utiliser et régénérer des ontologies.	Les services Web ne sont pas conçus pour utiliser et régénérer des ontologies.
Les agents peuvent être conçus pour être coopératifs, ce qui nous permet d'effectuer une composition de services compréhensible en utilisant différents protocoles.	Les services Web ne fournissent pas des protocoles de coopération qui peuvent conduire à une composition de services.
Les agents peuvent s'adapter à de nouveaux environnements.	Les services Web ne prennent pas en charge l'adaptabilité.

5.4 Architecture générale de système

Dans cette section, nous allons présenter l'architecture proposée pour résoudre le problème de composition. En effet, l'architecture proposée est basée sur un ensemble d'agents (situé et mobile), et pour l'avantage offert par ce paradigme et afin de maintenir le problème de changement de paramètre lors de l'opération de composition. Notre architecture est basée sur les trois composants principaux de l'architecture orientée service (SOA) [95], qui correspond d'un client

et d'un fournisseur, ces derniers sont représentés par une interface web. Les centres de données représentent les répertoires de service dans l'architecture classique. En outre, notre architecture utilise le broker qui joue un rôle de coordinateur afin de réaliser les différentes opérations dans le système soit le déploiement de services, soit la composition des services. L'architecture proposée dans notre travail est illustrée dans la figure suivante (Figure 5.1) est constituée des composants suivants :

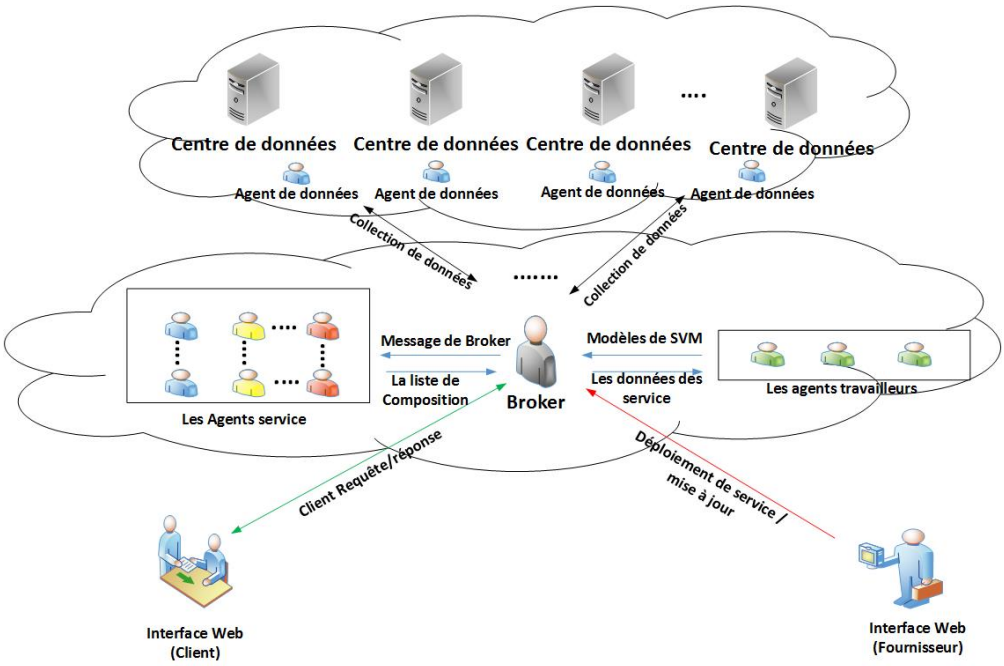

FIGURE 5.1 – Architecture proposée pour la composition de services [88].

5.4.1 Interface web

Ce composant est situé chez les deux types d'utilisateurs : le demandeur de service et le fournisseur de service. Le but principal de cette unité est d'introduire les requêtes du client et d'afficher les résultats. De plus, dans le cas du fournisseur et à travers cette interface, il peut réaliser deux opérations : (i) soit le déploiement du service en présentant la description du service afin de fournir une définition du service (ii) ou bien envoyer des mises à jour sur les services déployés. Ce composant présente un avantage en ce qui facilite l'interaction entre les utilisateurs et le système via l'interface graphique. Dans ce cadre, nous assignons un agent dans les deux côtés (client et fournisseur) pour contrôler les données et normaliser les demandes.

5.4.2 L'agent broker

Ce composant gère le processus de composition du service et gère aussi les changements sur les services déployés. Cet agent est créé en tant qu'un middleware entre le client et le fournisseur, il représente le gestionnaire du système. Ainsi, il constitue le coordinateur dans un système distribué, car son rôle consiste à recevoir des demandes de l'interface client et à rechercher des services dans le *Cloud* en envoyant des agents mobiles. En conséquence, cet agent est situé dans le *Cloud* pour utiliser ses ressources.

5.4.3 Les agents situés

Nous allons utiliser trois types d'agents. Le premier type est utilisé pour assurer la tâche de prévision à travers la création des modèles, ces agents sont des travailleurs. Ces agents utilisent les informations collectées à partir de l'historique d'utilisation de ces services qui sont stockés dans la base de données de l'agent broker. Ces modèles sont présentés par des modèles SVM afin de les utiliser dans la prochaine étape. Le deuxième type est les agents services. Le rôle de ces derniers est d'exécuter l'opération de composition ; ce type d'agents se communiquent entre eux sauf ceux ayant le même type de service (type—hôtel, restaurant...etc.) afin de combiner la meilleure composition par apport aux autres agents (compétition entre les agents). Le dernier type est les agents de données, ces agents sont situés dans les centres de données ; ces agents sont créés dans le but de chercher des services appropriés au requête courante. Après la collection de services, ces agents donnent la liste de services aux agents mobiles pour la rendre à l'agent broker.

5.4.4 Les agents mobiles

Le rôle de ces agents est de se déplacer d'une localisation à une autre dans notre système. Dans cette architecture, nous avons utilisé deux types d'agent. Le premier type est créé pour collecter des informations depuis les interfaces web et l'envoi des résultats. Ces informations sont collectées à partir des demandes des clients et des fournisseurs de services, ces informations correspondent au service qui doit être déployé ou pour introduire une nouvelle information pour les mises à jour de services. Le deuxième type est créé pour se déplacer depuis la localisation de l'agent broker aux différents centres de données dans le *Cloud Computing* afin de collecter les informations sur les services candidats.

5.5 Architecture des agents utilisés

Comme nous avons vu dans la section précédente, notre architecture est composée par un ensemble d'agents. Dans cette section, nous allons présenter les architectures de chaque catégorie. Comme nous le savons d'après la littérature dans les systèmes multi-agent qu'il existe plusieurs types d'agent, les agents utilisés dans notre approche sont de type cognitif et réactif ou mobile.

- **Agent Interface :** l'architecture de cet agent illustrée dans la figure 5.2 est composée de trois éléments qui sont :
  - Base de connaissances : qui comporte les informations sur les demandes des clients pour assurer que les données entrées sont compréhensibles par le système.
  - Le moteur : les tâches réalisé dans cet agent sont la vérification de la requête, la création d'un modèle pour la requête, et afficher les résultats envoyés de la part de l'agent broker.

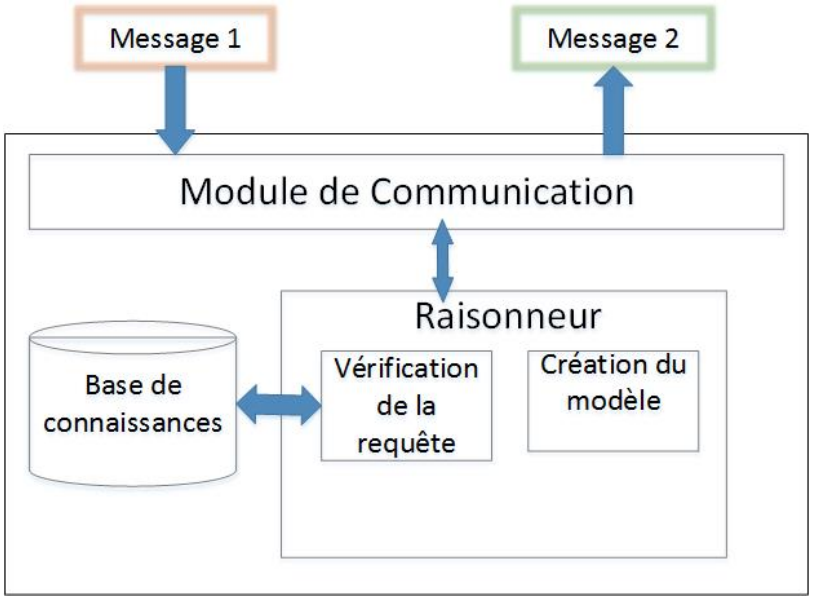

FIGURE 5.2 – Architecture de l'agent interface.

- **Agent broker** : dans la figure suivante (figure 5.3), nous présentons les éléments de l'architecture de l'agent broker. Cette architecture est composée de quatre éléments qui sont les outils de stockage et de connaissance un moteur de raisonnement et un moyen de communication.

- Base de connaissances : pour stocker les connaissances de l'agent représentées par des règles à suivre afin d'exécuter ses tâches.
- Base de modèles : qui contient les modèles SVM de chaque type de services après l'étape d'apprentissage.
- Le raisonneur : qui représente le noyau, cet élément comporte ces opérations : la décomposition de la requête ; en utilisant les informations stockées dans la base de connaissances. Opération de l'affectation : qui fait l'affectation des sous-tâches pour les autres agents. Collecter et sauvegarder : afin de collecter et sauvegarder les modèles construits d'après les agents travailleurs, ou les agents services dans le cas de service composite. L'opération de décision : qui utilise les valeurs obtenues à la fin du processus de composition pour décider quel est le service à utiliser.
- Module de communication : pour échanger les messages entre les autres agents du système.

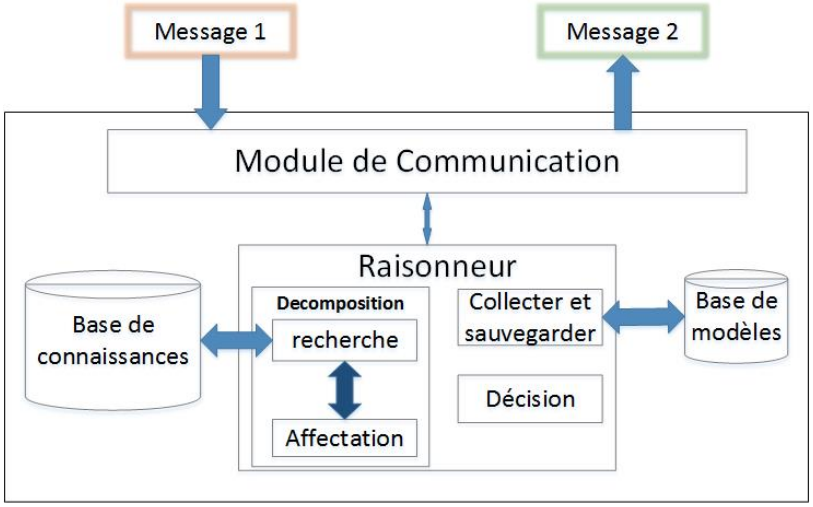

FIGURE 5.3 – Architecture de l'agent broker.

- **Agent travailleur** : l'architecture de cet agent représentée dans la figure 5.4. comporte trois éléments qui sont :
  - Base de connaissances : pour stocker les données et les paramètres utilisés dans l'algorithme d'apprentissage.
  - Module de communication : pour faciliter la communication avec l'agent broker.
  - Le moteur de raisonnement : qui comporte les opérations sur le calcul des valeurs de

QoS, ainsi que l'exécution de l'algorithme SVM afin de construire un modèle

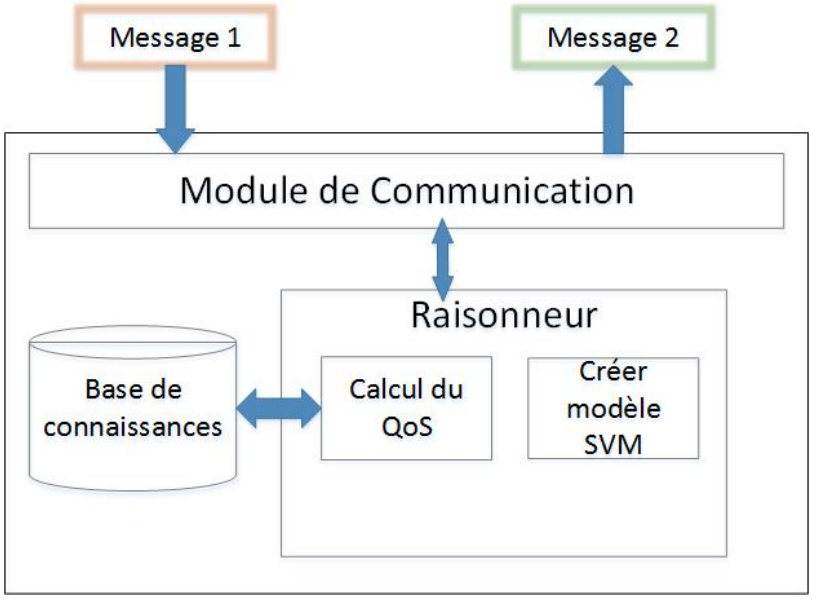

FIGURE 5.4 – Architecture de l'agent travailleur.

- **Agent service** : comme l'agent travailleur, cet agent a trois éléments qui sont :
  - Module de communication : pour communiquer avec l'agent broker si le type de service est élu alors il faut envoyer le service final (service composite), et avec les agents de service par l'échange des offres.
  - Base de connaissances : qui comporte les règles à suivre pour prendre la décision.
  - Le moteur de raisonnement : qui peut réaliser les opérations qui suivent : le calcul de l'intervalle de confiance, sauvegarder les offres à partir des autres agents, et prendre une décision.

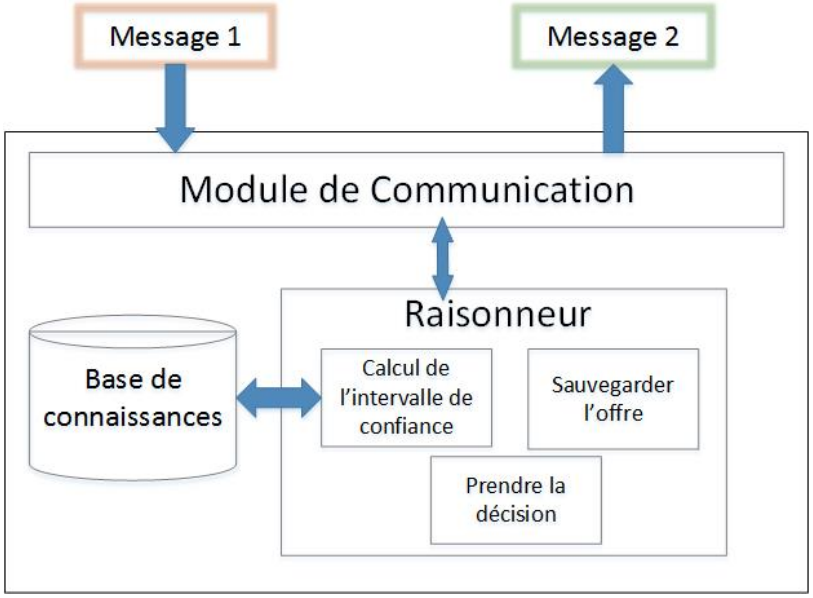

FIGURE 5.5 – Architecture de l'agent service.

5.6 Le modèle de coopération utilisé

Dans cette section, nous allons présenter les méthodes utilisées dans notre système. Le modèle est utilisé entre les agents pour arriver à une solution pertinente. De plus, ce modèle aide à fournir un moyen de compréhension entre les agents pour produire une meilleure combinaison (service composite dans notre cas). Les méthodes utilisées dans notre protocole de coopération sont présentées comme suit [88] :

5.6.1 Facteur de confiance

Dans notre travail, nous avons utiliser une méthode statistique intitulé "intervalle de confiance". Elle est définie comme une gamme calculée en utilisant des statistiques d'échantillonnage pour estimer un paramètre de population inconnu avec un niveau de confiance donné. Ce concept est également appelé la marge d'erreur qui dépend du niveau de confiance et de l'erreur standard [80][112]. L'intervalle de confiance est formulé comme suit :

$$Intervalle\ de\ confiance = \overline{X} \pm t\frac{s}{\sqrt{n}} \tag{5.1}$$

$\overline{X}$ présente la valeur moyenne, t est un multiplicateur généralement entre 1.5 et 2, s la valeur courante, n le nombre de données dans la collection.

5.6.2 La relation de dominance

Ce réfère à extraire le service dominant parmi les autres à partir des valeurs de qualité de service. On dit le service X est meilleur qu'un service Y (X domine Y) dans le cas où tous les paramètres du service X sont meilleur que Y pour les mêmes paramètres et il existe au moins un paramètre de service X qui est strictement meilleur que celui de Y [22][28]. La relation de dominance est formulée par l'équation suivante :

$$\begin{gathered} X(x_1, x_2, ..., x_n) > Y(y_1, y_2, ..., y_n) \\ (i) x_i \geq y_i, \forall i = 1, 2, ..., n \\ (ii) \exists j \in \{1, 2, ..., n\} : x_i > y_i \end{gathered} \tag{5.2}$$

Où '>' représente la relation de dominance, X et Y sont de services, x_i et y_i représentent les valeurs de qualités de service. Dans cet exemple, X domine Y si les deux conditions sont vérifiées.

5.7 Le processus de composition

Dans cette section, nous allons présenter le processus de composition qui est composé par une suite d'opérations [88]. Ces Opérations sont illustrées par la figure suivante (Figure 5.6) :

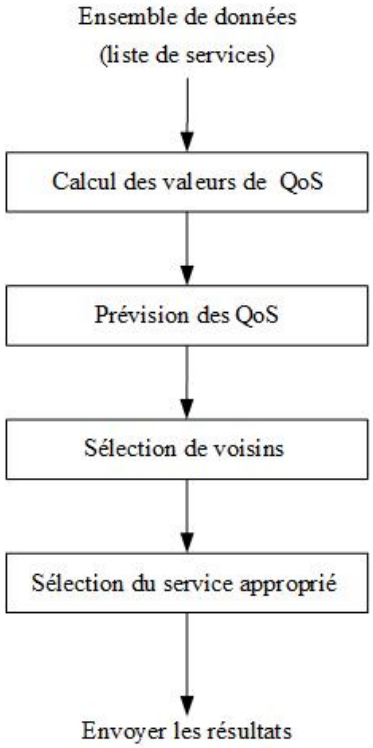

FIGURE 5.6 – Le processus de composition de services [88].

5.7.1 Calcul des valeurs QoS

Après la collection des données sur les services depuis les centres de données qui sont situés dans les différentes régions, l'agent broker donne ces services aux agents travailleurs afin de calculer les QoS pour chaque service en utilisant les équations présentées dans le chapitre 3 (équation 3.1 à 3.5). Lorsque cette opération est réalisée, les agents travailleurs renvoient les résultats à l'agent broker afin de les sauvegarder dans sa propre base de données pour l'étape de prévision. Comme nous l'avons mentionné précédemment, ces valeurs sont utilisées pour évaluer les services et différencier un service d'un autre.

5.7.2 Prévision des QoS

Dans cette étape, les agents travailleurs utilisent les valeurs calculées dans l'étape précédente (QoS) pour l'étape de prévision. Dans cette opération, nous allons construire un modèle de prévision pour chaque sous-requête (sous-requête défini par le type de service depuis la requête courante) pour sélectionner le service préférable d'un service composite. Notre approche proposée utilise les SVM pour la régression (SVR) pour prédire le pourcentage d'efficacité du service en fonction des valeurs de QoS (voir la Figure 5.9). Afin de construire le modèle de prévision, l'agent broker donne l'historique sur les valeurs de QoS qui sont sauvegardés depuis les expériences précédentes. De plus, les SVM utilisent deux étapes importantes avant d'utiliser le modèle. Ces étapes concernent l'apprentissage et la validation du modèle (voir la Figure 5.7 et Figure 5.8). En outre, cette méthode utilise les valeurs de QoS pour définir le degré d'acceptation de chaque service. Chaque enregistrement dans le modèle représente un service tel que les paramètres QoS et les labels défini dans l'équation 5.3 est calculé par le nombre de fois qu'un service a été utilisé dans la composition du service (NTISC) divisé par le nombre de fois que le service est proposé comme participant à la composition du service (NTPSC).

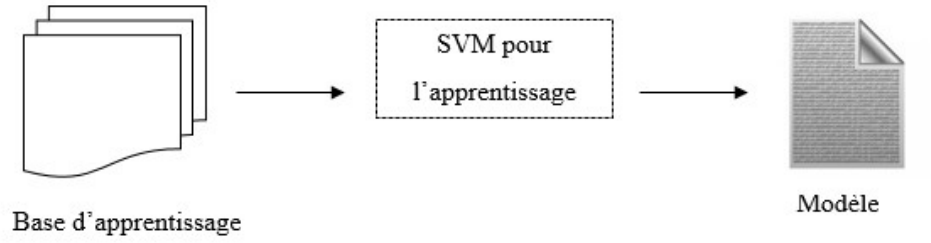

FIGURE 5.7 – La phase d'apprentissage pour les SVMs.

A l'entrée de l'SVM, nous donnons un ensemble de données sous forme de fichiers qui contiennent les caractéristiques du phénomène à traiter. Ces dernières doivent être normalisées pour que la méthode fonctionne bien. Afin d'arriver à un modèle, nous allons exécuter l'algorithme d'apprentissage plusieurs fois afin de minimiser l'erreur et régler les paramètres des SVM qui sont :

- le type de traitement soit régression ou classification ;
- le type de noyau utilisé (RBF, Sigmoïde, Linéaire, Polynomial...) ;
- Une constante C ;
- Gamma, epsilon ... etc. sont des paramètres choisis selon le type de noyau.

Afin de valider le modèle obtenu, nous l'utilisons avec les données pour tester l'efficacité du modèle. La figure suivante montre l'étape de validation du modèle :

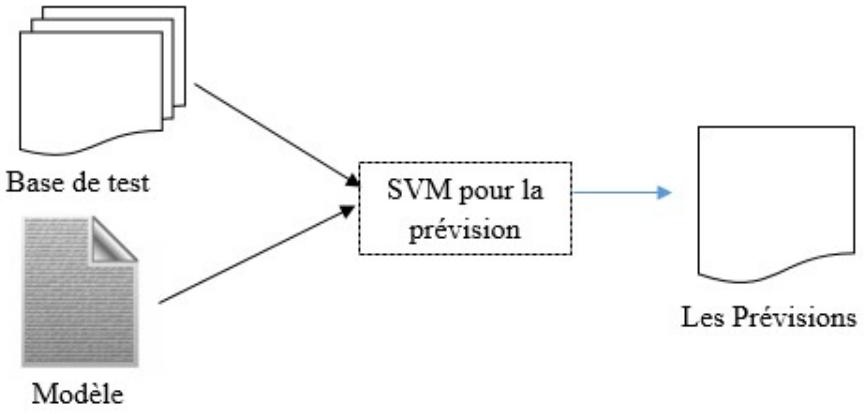

FIGURE 5.8 – Etape de test du modèle.

Après l'exécution de l'étape d'apprentissage plusieurs fois afin de réduire les erreurs, dans notre travail nous avons utilisé l'erreur quadratique moyenne et le pourcentage d'erreur moyen absolue [121] présenté dans les équations 5.4 et 5.5 respectivement, par lequel, les agents travailleurs envoient les modèles obtenus à l'agent broker avec l'erreur calculée. Après la collection des modèles, l'agent broker choisit les meilleurs modèles (le meilleur modèle se réfère à celui ayant le minimum d'erreur dans l'étape d'apprentissage) puis il les sauvegarde pour la prochaine exécution (requête). En outre, le broker lance un ensemble d'agents (agents de service sur la figure 5.1) qui représente les services collectés en leur donnant les modèles, le vecteur du service qui contient les valeurs QoS et la liste des candidats (services) qui ont été créés pour participer dans le processus de composition de service. Le message utilisé est présenté dans l'équation 5.6. La figure 5.9 résume toutes les étapes de l'algorithme *SVM* :

$$Service\ Label = \frac{NTISC}{NTPSC} \tag{5.3}$$

$$Erreur\ quadratique\ moyenne = \sqrt{\frac{1}{N\sum_{t=1}^{N}[X_t - \hat{X}_t]^2}} \tag{5.4}$$

X_t représente la valeur réel et $\hat{X}_t$ représente la valeur prédite.

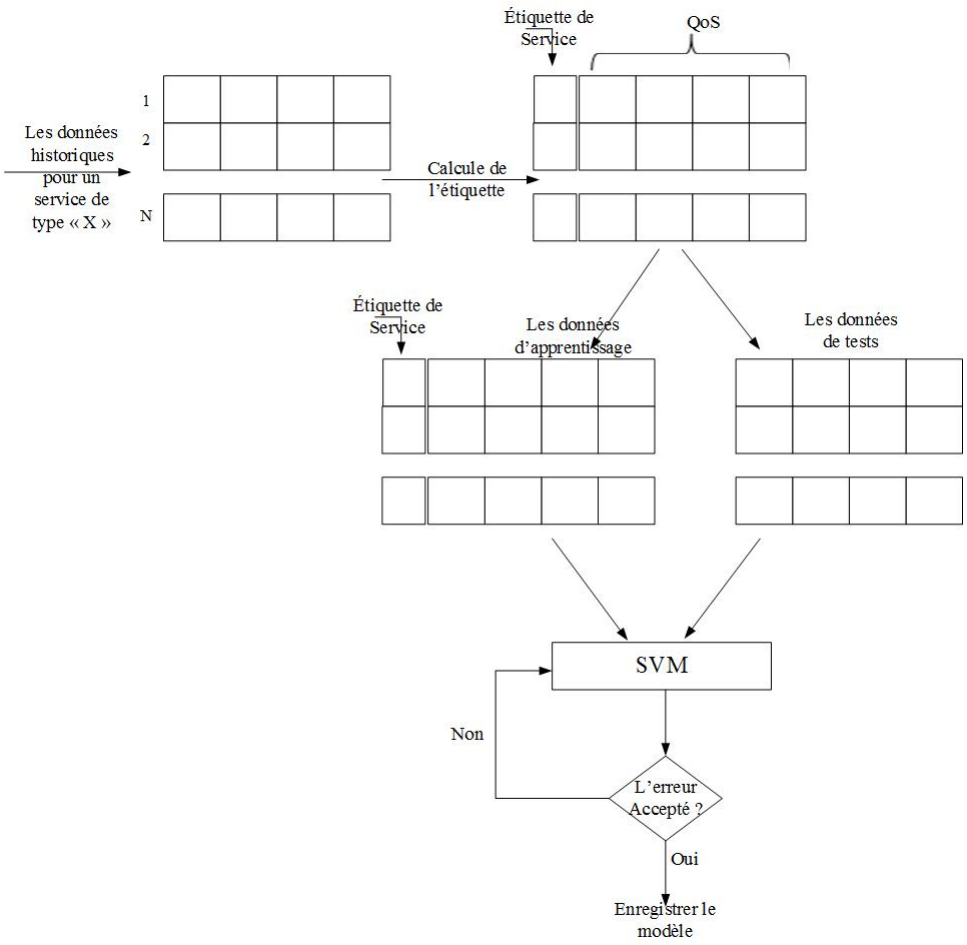

FIGURE 5.9 – Schéma globale de fonctionnement de l'SVM.

$$Pourcentage\ de\ l'erreur\ moyenne\ absolue = \left(\frac{valeur\_approximée - valeur\_réel}{valeur\_réel}\right) \times 100 \tag{5.5}$$

$$Message\_de\_Broker(LS, AQ, M) \tag{5.6}$$

LS représente la liste de services (les services candidats), à l'exclusion du type du même agent, cette liste est définie comme suit :

$$LS = \{\{S_{1,1}, S_{1,2}, \ldots, S_{1,n}\}, \{S_{2,1}, S_{2,2}, \ldots, S_{2,n}\}, \ldots, \{S_{q,1}, S_{q,2}, \ldots, S_{q,n}\}\} \tag{5.7}$$

AQ : est un vecteur qui représente les valeurs de QoS pour chaque service (agent).

$$M = \{M_1, M_2, \ldots, M_n\}\ O\grave{u}\ \ M_i : le\ mod\grave{e}le\ SVM\ pour\ un\ service\ de\ type\ i \tag{5.8}$$

5.7.3 Sélection de voisins

L'agent service utilise l'intervalle de confiance mentionné dans la section précédente (équation 5.1) pour sélectionner la liste d'agents qui peuvent donner la meilleure solution en termes de valeurs de qualité de service. La méthode utilisée est basée sur les données définies pour calculer la valeur moyenne à travers la génération des valeurs proche à celles de QoS. L'idée est de calculer un intervalle de confiance pour chaque valeur de QoS afin de construire une liste de confiance (cette liste est utilisée pour accepter ou éliminer des services). Afin d'évaluer un service, nous avons proposé deux possibilités. La première possibilité est le nombre de QoS. Durant l'opération de calcul de l'intervalle de confiance pour chaque QoS, l'agent compare les résultats avec les services disponibles. Si le service courant a des QoS meilleures que ceux mentionnées alors on considère ce service comme un voisin (Exemple. Si on a deux services S1 (0.1, **0.2**, **0.3**, **0.33**, et 0.2) et S2 (0.1, **0.3**, **0.5**, **0.4**, et 0.2), comme on peut l'observer dans cet exemple S2 a 3 QoS mieux que S1, dans ce cas on garde S2). La deuxième possibilité est d'utiliser la priorité pour choisir les services à partir des valeurs de QoS, dans ce cas l'agent a le choix soit d'utiliser la priorité définie dans sa propre base de connaissance (dans ce cas, cette priorité est définie d'après les expériences déjà réalisées, cela peut donner de bon résultats à la fin du processus de composition) ou la définir de la part du client. En outre, ces agents communiquent entre eux par l'échange de messages, ces messages sont représentés par un vecteur qui porte les valeurs de QoS. De plus, ces derniers sont une sorte d'offres qui peuvent être faites concernant l'acceptation ou le refus de l'utilisateur en fonction du facteur de confiance calculé par chaque agent pour chaque offre reçue.

5.7.4 Sélection du service approprié

Dans cette étape, chaque agent service utilise le modèle SVM qui est calculé dans l'étape précédente afin d'accepter ou refuser l'offre proposée de la part des autres voisins ; l'offre présentée par les valeurs de QoS. Dans le cas où l'agent a plusieurs offres et pour éviter le problème de conflit, nous allons utiliser la relation de dominance [150] qui est présentée dans la section précédente afin de choisir le service approprié à base de la priorité de QoS formulé dans l'équation 5.9. Dans notre approche, nous avons divisé les valeurs de QoS en deux objectifs la maximisation et la minimisation. Les QoS à minimiser sont respectivement : le coût, le temps de

réponse et la maintenabilité, les objectifs à maximiser sont : fiabilité, réputation, disponibilité et pertinence. Quand tous les agents ont reçu et accepté les offres, les résultats seront envoyés à l'agent broker, et sont présentés comme une liste d'agents participants à la tâche de composition. L'agent broker choisit la meilleure composition en termes de QoS, puis, il renvoie les résultats au client à travers l'interface graphique.

$$Relation\ de\ Dominance = (RO, priorité_{QoS}) = QoS* \tag{5.9}$$

$RO = \{QoS_1, QoS_2, \ldots, QoS_n\}$; Où RO représente la liste des vecteurs qui porte les offres reçues, QoS_i est un vecteur qui représente les QoS de l'agent i, QoS* c'est la meilleure offre. Chaque QoS donne les valeurs de qualité de service défini par : $QoS = \{q_1, q_2, \ldots, q_n\}\ q_j$: c'est la valeur de la qualité de service (mentionnée dans le Tableau 4.1).

Le diagramme suivant (figure 5.10) résume l'interaction entre les agents afin de produire un service composite.

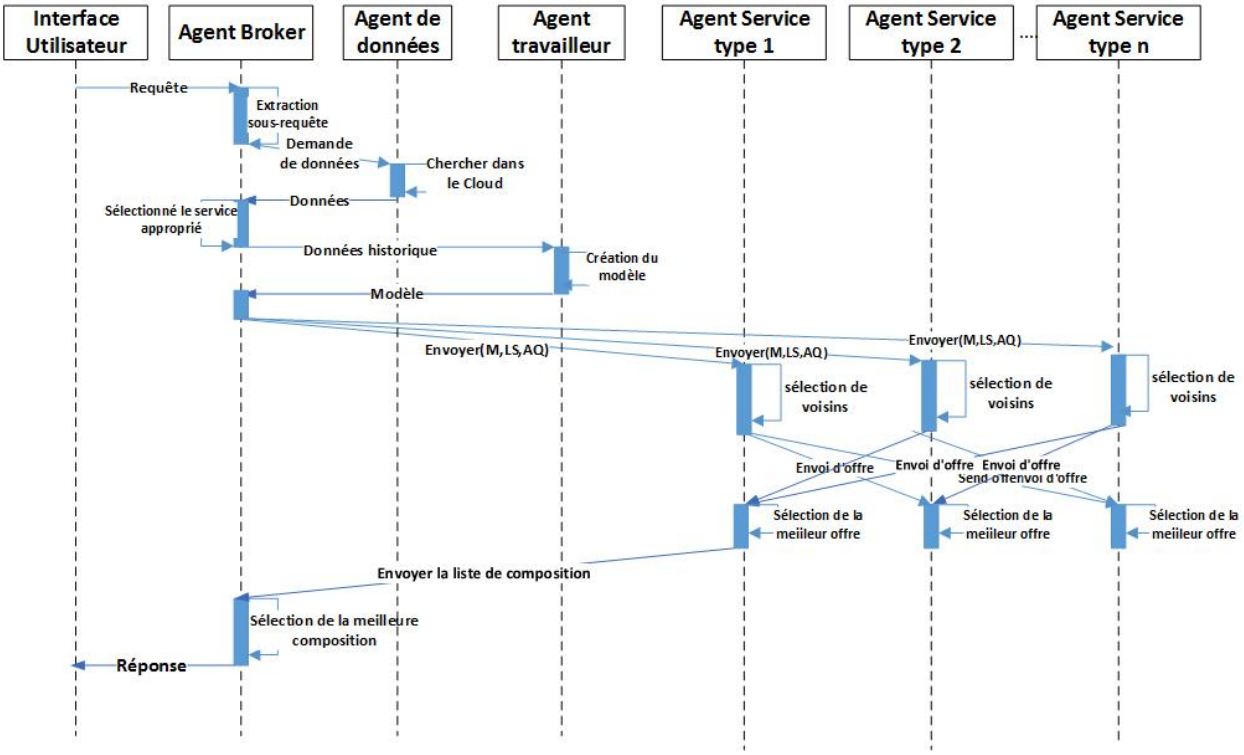

FIGURE 5.10 – Le scénario de la composition de services.

5.8 Algorithme de composition proposé

Après la présentation de l'architecture du système, cette section présente l'algorithme proposé pour résoudre le problème. Chaque agent service exécute cet algorithme. Le résultat final de cet algorithme présenté par une liste qui contient la liste des agents participant dans la composition des services. Premièrement, l'agent service reçoit le message depuis l'agent broker (voir l'équation (5.6)). Après la réception, chaque agent calcule l'intervalle de confiance pour chaque valeur de QoS (voir ligne 1 jusqu'à 3). La prochaine étape est de créer la liste de voisinage en utilisant le facteur de confiance puis, chaque agent envoie son offre et attend une réponse (voir algorithme 1). Après la réception du message, l'agent service utilise le modèle SVM afin de prédire le degré d'acceptation : si l'offre est acceptée, l'agent la garde dans une liste, sinon il rejette l'offre (voir ligne 5 jusqu'à 11). Le reste de l'algorithme (voir ligne 12 jusqu'à 26) calcule la relation de dominance afin de déterminer l'offre acceptée. Quand l'agent termine la tache de sélection des offres (les services sélectionnés), il donne les résultats à l'agent broker comme une liste qui porte les adresses des agents participants à la composition de services. L'algorithme 1 fait la tâche de vérification de l'offre par rapport aux des autres demandes agents.

Algorithme 1 : La création de la liste des voisins

Données : LS : listes des vecteurs qui contiennent des informations sur les agents voisins (qualité de service et type de service),AQ : QoS pour l'agent actuel, C : Vecteur contenant les intervalles de confiance

Résultat : L : liste de voisins

Début

```
1 pour chaque Instance d dans LS faire
2     num := 0
3     pour chaque QoS q dans AQ faire
4         si LS_d[q] ≥ (AQ[q] − C[q]) et LS_d[q] ≤ (AQ[q] + C[q]) alors
5             num ← num + 1
6     si num > (NombreDeQoS/2) alors
7         Ajouter(listedesvoisins,d)
```

```
8  pour chaque agent A dans listedesvoisins faire
9      envoyer(A,AQ)
   Fin.
```

Le deuxième algorithme résume toutes les opérations entre les agents afin d'offrir un service composite.

Algorithme 2 : Algorithme de composition des services

Données : M : liste des modèles SVM pour chaque type de service, AQ,LS

Résultat : CL : une liste des vecteurs qui contient des informations de l'agent (agents participants à la composition de services)

Début

```
1  pour chaque QoS q dans AQ faire
2      C[q] ← CalculerIntervaldeConfiance (q,LS[q])
3  listedesvoisins ← CreerListeVoisin(LS,C,AQ)
4  OffreAcceptee ← 0
5  répéter
6      Recevoir(A, AQ_A) [a]
7      si Acceptable(SVM_predict(AQ_A,M_i)) alors
8          Ajouter(T, i , AQ_A)
9      sinon
10         Envoyer(A,refuser)
11 jusqu'à existe d'autres messages;
12 RestMessages ← 0 ;
```

a. AQ_A est la QoS de l'agent A

```
répéter
    pour chaque type i dans T faire
        D[i] ← dominance (extractQoS(i))
    pour chaque donnée d dans D faire
        CL ← CL + A_0
        pour k ← 1 à d.taille() faire
            Envoyer(A_k,refuser)
    RestMessages ← CL.taille() - RestMessages
    pour j ← 1 à RestMessages faire
        Recevoir(A,m,t) ;
        si m = agrée alors
            OffreAcceptee ← OffreAcceptee + 1
        Supprimer(D[j],A)
    RestMessages ← CL.taille() - OffreAcceptee
jusqu'à OffreAcceptee < numbreDeType;
Fin.
```

5.9 Conclusion

Dans ce chapitre, nous avons présenté la conception du système qui consiste à résoudre le problème de composition de services dans le Cloud. Comme nous l'avons déjà mentionné, le problème posé devient très important dans le Cloud à cause de la diversité des services et la distribution dans des différentes localisations géographiques. Quand le Cloud utilise les services web, le problème posé est un *NP-complet* à cause de l'énorme nombre de service déployés dans le Cloud. Avec ce nombre important de services, on a besoin d'une solution qui peut donner une solution à ce problème. Cette solution fournit à l'utilisateur final un service composite selon l'aspect fonctionnel et non-fonctionnel de service.

Afin de résoudre le problème posé, nous avons proposé dans ce chapitre une architecture basée agents. L'idée vise à proposer un protocole de coopération entre les agents afin de résoudre le problème. Comme nous avons vu dans ce chapitre, lorsque nous voulons fournir un service composite avec des mesures de QoS appropriées, des problèmes surviennent en ce qui concerne le conflit entre les services. L'idée est de présenter l'intervalle de confiance entre les agents pour qu'ils choisissent les agents avec lesquels ils veulent travailler. L'utilisation de la relation de dominance nous encourage à confirmer les meilleurs choix grâce aux paramètres QoS.

En outre, ce travail peut gérer les changements de paramètres lors de la composition de services.

Dans le prochain chapitre, nous allons présenter la deuxième partie de notre contribution dans le but de résoudre le problème de composition. De plus, nous allons présenter des modèles de description afin de réduire le nombre de service dans le processus de composition.

Chapitre 6

Composition de services pour le déploiement

6.1 Introduction

L'émergence du *Cloud Computing* a apporté une révolution dans le monde informatique. Le cloud computing héberge des centaine de milliers de services (ou application) dans plusieurs centres de données distribués, et chaque service à besoin d'un espace de stockage, système de gestion de données (c'est à dire la gestion de machines virtuelles) et un système de gestion de couche réseau. En outre, cette évolution de Cloud représentée une solution proposée en faveur des utilisateurs que ce soit un client simple ou bien une entreprise. Parmi les solutions offertes par le Cloud c'est l'utilisation des services selon différents besoins du client à travers l'internet. Avant d'utiliser un service dans le Cloud, on doit passer par l'étape de déploiement. Le déploiement c'est une étape très importante car il va affecter le déroulement de service dans le future (lors d'utilisation de service). Afin de déployer un service, chacun utilise une combinaison d'autre modèle de services ; la nécessité d'une configuration de services. Généralement, les problèmes de combinaison ont été résolu par la composition de service. L'existence d'un système distribué comme Le Cloud Computing est justifié par la composition des milliers de services déployer dans les centres de données. Pour cette raison, nous allons présenter une solution considérable du problème de déploiement des services par la proposition d'une nouvelle architecture à base d'un système multi-agent.

Dans ce chapitre nous allons présenter aussi un exemple qui montre l'utilité et l'efficacité de

cette solution. Après, nous allons présenter l'architecture globale du système en détaillant son fonctionnement à l'aide d'un diagramme de séquence UML.

6.2 Etude de cas

Dans cette section nous allons présenter une étude de cas d'un problème de déploiement de services dans le Cloud. Dans notre cas : le déploiement des services bancaires. Le problème majeur lié aux services bancaires dans notre cas est : déployé ces services dans un environnement *Cloud Computing* en prend en considération le contrainte de la QoS. Afin de déployer ces service, il y a deux objectifs : nous allons proposer une méthode de combinaison de services qui ce base sur les trois modèles suivants : *Data as a Service (DaaS), Network as a Service (NaaS)*, et *Security as a Service (SECaaS)*. deuxième objective : assuré la QoS durant le processus de composition. Comme nous avons dit le Cloud héberge des milliers des services différents, nous avons besoin d'une méthode adaptable pour extraire un ensemble de services candidats parmi ces milliers. Notre objective premièrement consiste à collecter les services de types : NaaS, DaaS, et SECaaS. deuxièmement composer les services, finalement, déployer ces services bancaires.

6.3 Le modèle de description proposé

Afin d'assurer le succès de l'opération de déploiement, dans cette section nous allons présenter un modèle de description de service pour les différents modèles utiliser dans notre composition. Comme nous avons mentionné dans la section précédente qu'il existe plusieurs services déployés dans le *Cloud*, il est alors difficile de trouver un service approprié à notre besoin avec une bonne valeur de qualité de service. Les services DaaS ont des caractéristiques similaires de SaaS. Outre le service DaaS a était défini comme modèle de service qui est étendu des modèles de base de service Cloud (SaaS, PaaS, et IaaS). De plus, le modèle de DaaS a été proposé pour les entreprises qui utilisent les différents services sur les données. Le modèle présenté dans ce travail est basé sur des travaux déjà publiés dans la littérature, comme nous avons dit avant le modèle basé sur les services SaaS que nous allons utiliser est le modèle présenté dans [105]. Les caractéristiques de modèle proposé collecté depuis les travaux réalisés dans [133][151][98][124]. Le tableau suivant exprime le modèle utilisé :

TABLE 6.1 – Les caractéristiques de service de données (DaaS).

Catégorie	**Caractéristique**	**Description / Valeur**
Valeurs numériques	Valeur de la marque du vendeur	réel
	Version	Converti en entier
	Expérience des ingénieurs	Entier
	Niveaux de compétences des ingénieurs	Entier
Valeurs textuelles	Contributeur	Le nom de l'organisation qui contribue au développement de nouvelles versions de la cible DaaS
	Créateur	Le nom de l'organisation qui crée la cible DaaS
	Propriétaire	Le nom de l'organisation qui possède la cible DaaS
	Éditeur	Le nom de l'organisation qui publie la cible DaaS
	Identifiant	C'est une combinaison de chiffres et de lettres qui permettent d'identifier les DaaS cible uniquement.
	Le langage supporté	C'est la liste des langages supportés par la cible DaaS
	Certification	Le(s) nom(s) du (des) certificat(s) gagné(s) par la DaaS cible, attestant sa conformité à certaines normes ou codes.
Valeurs prédéterminées	Système de paiement	(i)Gratuit (ii)Dynamique (iii)Pay as you go
	Ouverture de Cloud	(i)Inconnu limité (ii)Basique (iii)Modéré (iv)Complet
	Type de Licence	(i)Open source (ii)Propriété
	Intégration	(i)Oui (ii)Non
	Accord formel	(i)Pas d'SLA (ii)SLA

	Standardisation	(i)Non (ii)API public (iii)Particulier (Standard, Organisation)
	Modèle de déploiement	(i)Privé (ii)Public (iii)Hybride (iv)Communauté
	Prise en charge des appareils mobiles	(i)Oui (ii)Non
	Prise en charge du mode hors ligne	(i)Oui (ii)Non
	Accessibilité aux données	(i)Lecture seulement (ii)CRUD
	Type de données	(i)Structuré (ii)Non structuré (iii)Semi-structuré
	Type de service	(i)DaaS générique (ii)DaaS spécialisées (iii)DaaS hybrides
	Modèle de paiement	(i)Prix à base de quantité (ii)Paiement par appel (iii) Modèle basé sur le type de données
	Prise en charge du modèle sémantique	(i)OWL-S (ii)WSMO (iii)SAWSDL (iv)RDF
	Modèle de représentation	(i)Relationnel (ii)Objet
	Type d'opération	(i)Basé sur la demande (ii)Basé sur la fonction
	Niveau de confidentialité	(i)Confidentialité des données (ii) Confidentialité de l'opération

Le tableau suivant (6.2) montre la description du modèle réseaux basé sur le modèle IaaS mentionné dans le travail de [105]. Les caractéristiques sont aussi collectées depuis les travaux de [33][156][24][31]. Le tableau suivant (tableau 6.2) exprime les différentes caractéristiques du service réseaux (NaaS) :

TABLE 6.2 – Les caractéristiques de service de réseaux (NaaS).

Catégorie	**Type de valeur**	**caractéristique**	**Description / Valeur**
Statique	Valeurs numériques	Version du système d'exploitation	Converti en entier
		Nombre d'instances	Nombre d'instances pouvant être crées dans l'infrastructure du Cloud pour le service Cloud (Entier)
		Bande passante du réseau (Gbps)	réel
		La latence du réseau	réel
	Valeurs textuelles	Editeur	Le nom de l'organisation qui publie le service NaaS
		Propriétaire	Le nom de l'organisation qui possède la cible NaaS
		Créateur	Le nom de l'organisation qui crée la cible NaaS
Pré-déterminé	Valeurs pré-déterminé	Pris en charge de la politique de conformité	(i) Oui(ii) Non
		Modèle de déploiement	(i) Privé (ii) Public (iii) Hybride (iv) Communauté
		Prise en charge de l'authentification des utilisateurs	(i) Oui (ii) Non
		Type de connectivité	(i)Direct (ii)Gateway
		Modèle de réseau de service	(i) Niveau 2 (ii) Niveau 3
		Type de service de réseau	(i)Multipath (ii)Multicast (iii)Multi-domaine
		Système de paiement	(i) paiement par utilisation (ii) abonnement mensuel
		Prise en charge l'aspect sémantique	(i)Oui (ii)Non (si le fournisseur utilise une ontologie pour décrire les services)

Lorsqu'on parle de service de sécurité la description de ce modèle est totalement différente d'un fournisseur à un autre. Dans notre travail nous avons assumé que chaque fournisseur a son propre modèle de description (car les modèles non déclarer pour des causes confidentielle). De plus ces services sont proposé comme étant une solution d'après l'équipe technique d'un fournisseur réseaux dans la même entreprise. Étant donné que la sécurité est proposée par l'entreprise par l'intermédiaire de son personnel technique, et afin d'accomplir notre processus de composition de service nous avons utilisé les valeurs de QoS calculées obtenues auprès de chaque fournisseur de services (calculé à partir de l'utilisation enregistrée précédemment) pour représenter les services de sécurité dans la solution proposée.

6.4 Architecture générale de système

Dans cette section, nous allons présenter notre approche et les outils utiliser afin de résoudre le problème de déploiement à travers le processus de composition de services. Notre architecture utilise le système multi-agents pour résoudre le problème concerné. De plus nous planifions d'intégrer l'approche proposée dans le Cloud, lorsqu'on intègre cette architecture dans le Cloud on peut utiliser ses caractéristiques. Ces caractéristiques sont présentées par une large bande passante pour assurer le transfert de données, le service à la demande (quand le broker nécessite des ressources pour compléter leurs opérations il peut l'obtenir). Lorsqu'on est intéressé par les systèmes multi-agents, l'aspect d'élasticité peut intervenir quand le système a besoin des ressources (machine virtuelle) pour créer ou éliminer un ou des agents. En outre cette section présente la description de notre architecture qui inclut le rôle de chaque agent. La figure suivante (6.1) illustre l'architecture proposée :

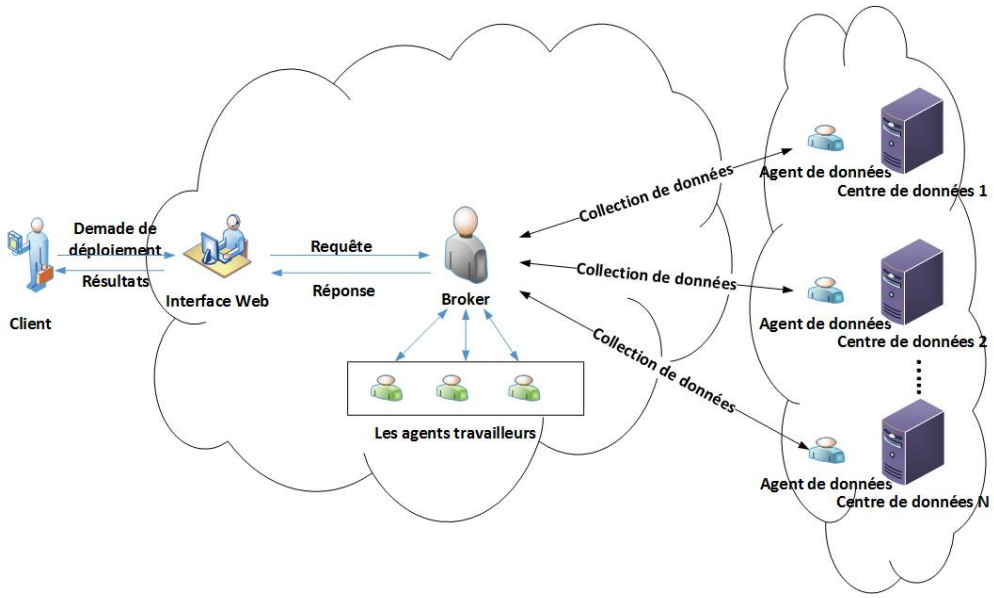

FIGURE 6.1 – L'architecture de déploiement à base d'agents.

La figure précédente montre l'architecture proposée pour résoudre le problème de déploiement dans le Cloud. En effet cette architecture est composée d'un ensemble d'agents certains situés dans une localisation et les autres mobiles pour se déplacer depuis une localisation à une autre. La figure 6.1. est composée par les composants suivants :

6.4.1 Interface Web

Dans notre architecture ce composant représente le lien entre le monde interne et externe (le client et le système) à partir d'une interface graphique. De plus le rôle principal de ce composant est de récupérer les requêtes de clients et affiche les résultats. En outre, dans cette interface nous assignons un agent situé pour interagir avec le client (fournisseur de service, dans notre cas) et le système. Après la récupération de la requête qui porte le type de service à déployer avec les QoS proposé par le fournisseur et une courte description de service, l'agent interface envoi ces données à l'agent broker.

6.4.2 Agent broker

Ce composant manipule le processus de déploiement de services et gère toutes les opérations du système. Cet agent est créé comme étant un administrateur dans notre système. De plus, l'agent broker remplace le coordinateur dans un système distribué, son rôle consiste est la

réception des requêtes de l'agent interface, recherche des services dans le Cloud par l'envoi des agents mobiles et l'affectation des tâches aux agents travailleurs.

6.4.3 Les Agents situés

L'architecture présentée dans la figure 6.1 est composée d'un ensemble d'agents. Dans cette architecture nous allons utiliser deux types d'agents. Ces derniers aident à compléter les différentes tâches dans l'approche proposée. Les agents utilisés sont présentés comme suit :

- **Agent travailleur** : cet un agent est responsable du calcul des valeurs de la qualité de services pour chaque service collecté afin de les utiliser dans l'algorithme. En effet, ces agents sont créés pour exécuter l'algorithme d'optimisation, dans notre travail nous allons utiliser le niched Pareto genetic algorithm (NPGA). Lorsque cette architecture est proposée pour être située dans le Cloud, cela signifie que plusieurs utilisateurs peuvent l'utiliser en même temps. Dans ce cas, l'agent broker crée d'autres agents travailleurs afin de gérer les nouvelles demandes.
- **Agent de données** : dans notre travail cet agent mobile se déplace d'une localisation à une autre localisation géographiques. L'objectif principal de cet agent est de découvrir les services selon la requête donnée. De plus ces agents utilisent la description mentionnée dans le tableau 6.1 et 6.2 afin de chercher les services dans les différents centre de données. Une fois l'étape de la découverte est terminée, ces agents retourne à la première localisation pour donner les résultats à l'agent broker. Pour des raisons de sécurité dans le Cloud, nous supposons que ces agents ont le droit de réaliser des tâches au niveau de centre de données.

6.5 Architecture des agents utilisés

Comme nous avons vue dans la section précédente notre architecture est composée d'un ensemble d'agents, dans cette section nous allons présenter les architectures de chaque catégorie. Comme nous le savons d'après la littérature dans les systèmes multi-agent, il existe plusieurs types d'agent, les agents utilisés dans notre approche sont de type cognitif et réactif ou mobile.

- **Agent Interface** : l'architecture de cet agent illustrée dans la figure 6.2 est composée de trois éléments qui sont :

- Base de connaissances : qui comporte les informations sur les demandes des clients pour construire un modèle qui assure que les données entrées sont compréhensible par le système.
- Le raisonneur : les tâches réalisées dans cet agent sont la vérification de la requête, la création d'un modèle pour la requête, et afficher les résultats envoyés de la part de l'agent broker.

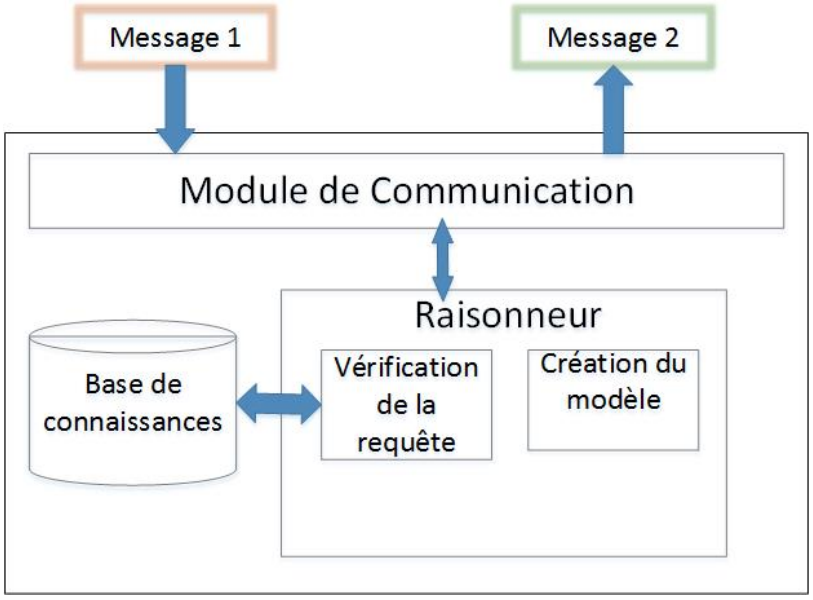

FIGURE 6.2 – Architecture de l'agent interface.

- **Agent broker** : l'architecture de cet agent est composée de :
  - Base de connaissances : pour stocker les connaissances de l'agent représentées par des règles à suivre afin d'exécuter ses tâches.
  - Module de communication : pour envoyer les requêtes aux agents travailleurs et les agents de données afin de récupérer les informations sur les services déployés.
  - Le moteur : comporte les opérations suivantes : décomposer la requête en sous-requêtes et les envoyer aux agents de données afin de récupérer les informations sur les services déployés. Affecter des tâches aux agents travailleurs, et recevoir les résultats pour prendre une décision, et exécuter l'opération de déploiement.

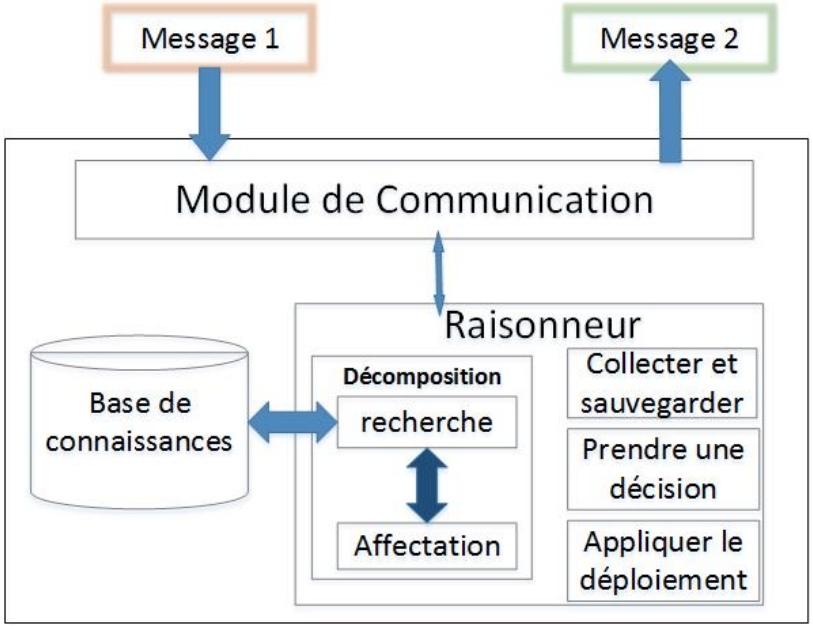

FIGURE 6.3 – Architecture de l'agent broker.

6.6 Le processus de déploiement de services

Après avoir présenté l'architecture de notre approche, dans cette section nous allons détailler les taches à réaliser pour déployer un service dans le Cloud. Comme nous avons mentionné précédemment, ce travail consiste à résoudre le problème par l'utilisation de la composition de services. La figure suivante (figure 6.4) présente les opérations dans notre processus de composition.

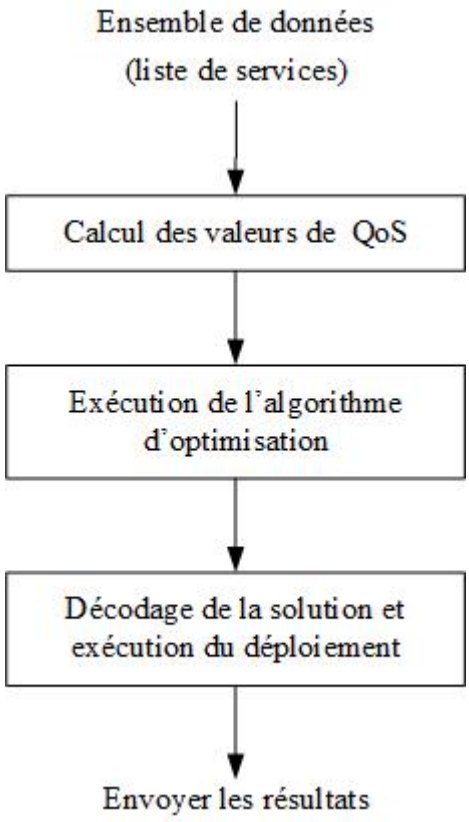

FIGURE 6.4 – Le processus de déploiement de services.

6.6.1 Calcul des valeurs de QoS

Après avoir collecter les données depuis les centres de données qui sont situés dans les différentes localisations géographiques, l'agent broker attribue aux agents travailleurs la tâche du calcul de valeurs des QoS pour chaque service en utilisant les équations présentées dans le chapitre 3 (équation 3.1 au 3.5). Quand cette opération est terminée, les agents travailleurs commencent l'exécution de l'algorithme afin d'extraire le meilleur résultat par rapport à la requête du client. De plus, les valeurs de qualités de services sont utilisées pour évaluer les services.

6.6.2 Exécution de l'algorithme d'optimisation

Dans cette étape, l'agent travailleur utilise les valeurs calculées précédemment afin de commencer l'algorithme NPGA [48]. L'algorithme utilisé est basé sur l'algorithme génétique traditionnel avec des modifications dans l'étape de sélection. En outre, cet algorithme utilise la relation de dominance dans une partie de la population. Cet algorithme essaye de générer des

solutions à partir des générations dans le but d'améliorer les résultats. Lorsque nous obtenons les meilleurs résultats conformément à la demande, les agents travailleurs envoient les derniers résultats avec leurs évaluations au broker (cette évaluation est calculée par l'utilisation de la distance euclidienne entre la requête et le service composite). La figure 6.5 Présente les différentes étapes de l'algorithme d'optimisation.

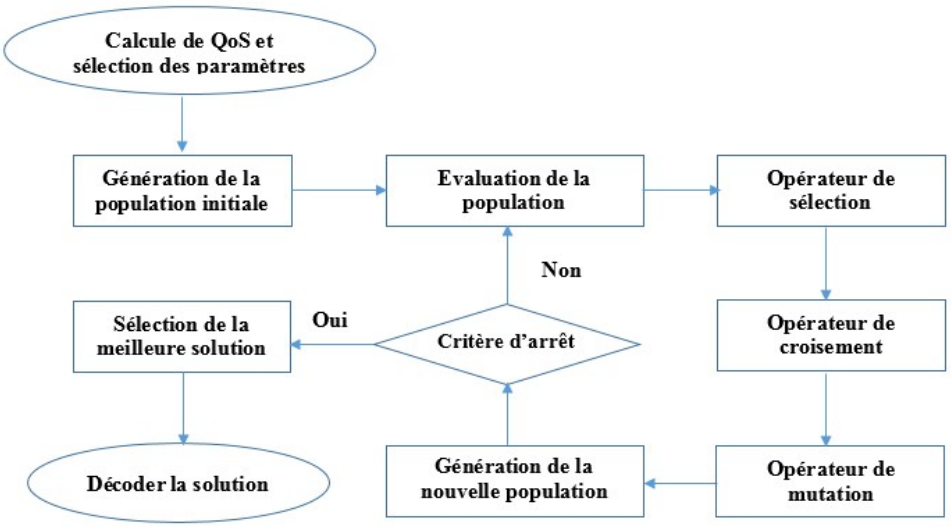

FIGURE 6.5 – L'organigramme d'algorithme NPGA.

La figure 6.5 montre les différentes étapes de l'algorithme NPGA, ces étapes sont présentées comme suit :

- **La sélection des paramètres** : dans cette étape, nous allons utiliser les valeurs de QoS qui sont calculées par les agents travailleurs. Dans la littérature, l'algorithme NPGA est défini comme un algorithme d'optimisation qui utilise la notion de dominance de Pareto [144] (cette notion donnée dans le chapitre 5).En outre, cet algorithme se base sur deux valeurs qui consistent en rayon de niche (σ_{share}) pour calculer le fitness partagé et un nombre de comparaison dans l'étape de sélection (t_{dom}).
- **Génération de la population initiale** : dans cette étape, nous allons construire la première génération à partir des valeurs de QoS qui correspondent aux services de données, réseaux et sécurité. En outre, nous avons besoin de prendre en considération une condition de combinaison entre les services réseaux et sécurité qui sont de même fournisseur de service. Cette précaution est mentionnée pour la topologie de service de sécurité proposée

du service réseau pour la même entreprise (problèmes de confidentialité). A la fin de cette étape, nous arrivons à N solutions pour commencer l'algorithme d'optimisation. Chaque solution dans la population représente les valeurs de QoS pour chaque type de service. La figure suivante présente une solution :

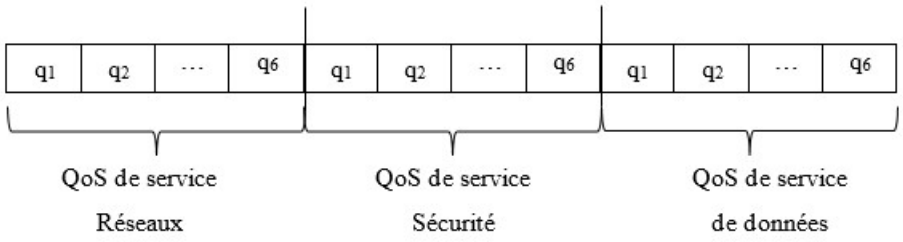

FIGURE 6.6 – Le codage d'une solution.

Où $q_i : 1 \geq i \leq 6$; représentent les valeurs de qualité de services présenté dans le chapitre précèdent.

– **L'évaluation de la population** : dans cette étape, nous allons évaluer chaque solution. Pour ce travail, lorsqu'on est dans un problème d'optimisation, nous avons utilisé des valeurs à maximiser et d'autres à minimiser. Pour cela, nous avons choisis le coût, et le temps de réponse comme objectif à minimiser et la fiabilité, la réputation, la pertinence, et la disponibilité à maximiser. En outre, dans ce travail, nous supposons que le temps de réponse est calculé séquentiellement. Afin d'évaluer chaque solution, nous avons utilisé les équations suivantes :

TABLE 6.3 – Les fonctions d'agrégation de QoS^A pour chaque solution générée.

Qualité de service	**Fonction**	**Mesure**
Coût	$\sum_{i=1}^{3} Coût\_QoS_i$	$
Temps de réponse	$\sum_{i=1}^{3} Temps\ de\ réponse\_QoS_i$	ms
Fiabilité	$Min(Fiabilité\_QoS_1, \ldots, Fiabilité\_QoS_3)$	%
Réputation	$Min(Réputation\_QoS_1, \ldots, Réputation\_QoS_3)$	%
Pertinence	$Min(Pertinence\_QoS_1, \ldots, Pertinence\_QoS_3)$	%
Disponibilité	$Min(Disponibilité\_QoS_1, \ldots, Disponibilité\_QoS_3)$	%

– **L'opérateur de sélection** : premièrement, nous avons choisis deux candidats à la fois de façon aléatoire et les comparons avec d'autre ensemble candidats (on a choisis t_{dom} candidats) en utilisant la relation de dominance définie dans le chapitre 5. Lorsque nous finissions cette opération à ces deux candidats sélectionnés, on prend le vainqueur (par exemple, quand on a S1 (0.1, **0.2, 0.3, 0.33**, et 0.2) et S2 (0.1, **0.3, 0.5, 0.4**, et 0.2) on peut voir que le service S2 a trois valeurs de QoS mieux que S1, alors on garde S2). Dans le cas où nous n'avons pas de gagnant dans cette opération, nous les comparons en utilisant la fonction de partage (voir l'équation 6.1). On continue ces étapes jusqu'à l'arrivée à la moitié de la taille pour la première génération (N/2) afin de les préparer pour l'étape suivante (opérateur de croisement). Nous pouvons résumer cette opération dans les étapes suivantes [48] :

  1. commence par i = 1 :
  2. choisir deux candidats x_1 et x_2 aléatoirement pour la sélection.
  3. choisir un ensemble d'individus depuis la population (t_{dom} individu).
  4. faire une comparaison entre les individus x_1 et x_2 avec les autres en utilisant la notion de dominance.
  5. si un candidat est dominé par la liste de comparaison tandis que l'autre est moins dominé, alors on le sélectionne pour la reproduction puis on passe à l'étape 7, sinon on doit vérifier l'étape 6.
  6. si l'un de ces candidats ou les deux sont dominés par la liste de comparaison, dans ce cas nous allons utiliser le niche pour sélectionner le gagnant.
  7. répéter ces étapes jusqu'à l'arrivée à la moitié de la taille pour la première population.

La fonction partagée et le niche utilisé dans cette opération sont définies comme suit :

La valeur de Niche : c'est la mesure du nombre de niche (m_i), elle fait référence à une estimation du degré auquel le voisinage (*niche*) de la solution i est encombré [28][48]. La fonction de partage et le nombre de niches utilisés dans cette opération sont calculés en utilisant les équations suivantes mentionnées dans [107][120] :

$$m_i = \sum_{i=1}^{N} Sh(d_{i,j}) \tag{6.1}$$

$$Sh(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_{share}}\right), & d_{i,j} \leq \sigma_{share} \\ 0, & Sinon \end{cases} \tag{6.2}$$

m_i représente la valeur de niche pour l'individu i, $d_{i,j}$ représente la distance euclidienne entre l'individu i et j, $Sh(d_{i,j})$ est la fonction de partage et σ_{share} c'est le rayon de niche.

Distance de Crowding : se réfère à une estimation de la densité des solutions entourant ce point courant (solution). Cette distance mesure une solution particulière à travers la distance moyenne de ses deux solutions voisines [28][83]. La distance de Crowding est présentée dans l'équation suivante :

$$D_{IR} = \sqrt{\sum_{i=1}^{M} \left(\frac{f_i(x) - R_i}{f_i^{max} - f_i^{min}}\right)^2} \tag{6.3}$$

D_{IR} représente la distance de Crowding entre la solution I et la requête R pour M QoS. f_i^{max} et f_i^{min} sont respectivement la valeur maximale et minimale de QoS^A i.

– **Opérateur de croisement** : dans cette opération, nous appliquons le croisement en utilisant l'opération traditionnelle inspirée depuis le phénomène biologique et utilisée dans l'algorithme génétique basique. Afin de reproduire des nouvelles solutions en utilisant les individus existants, dans notre travail, nous avons choisi deux individus de façon aléatoire et on a choisi un point pour faire le croisement. Dans le cas où les individus sélectionnés sont invalides en ce qui concerne la condition du réseau et des services de sécurité, on a besoin de choisir de différents individus par exemple, lorsqu'on a un service composite : SC1(**N1**,**S11**,D2) et SC2(**N12**,**S2**,D3) à la fin de cette opération, alors on a besoin de répéter l'opération et de générer des nouvelles solutions car le service réseau N1 combiné par le service sécurité S11 qui ne sont pas fait de même fournisseur ; c'est à dire dans notre étude de cas, nous avons assumé que chaque service réseau a 10 services de sécurité. La figure suivante montre l'opération de croisement :

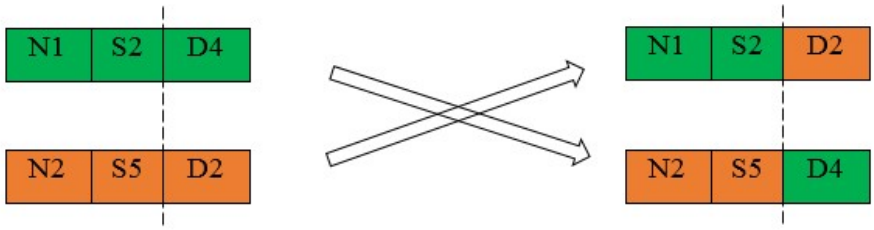

FIGURE 6.7 – Opération de croisement.

- **Opérateur de mutation** : dans cette opération, nous changeons un génome (type de service) de façon aléatoire avec d'autre individu depuis la population courante, en prenant en considération la même condition mentionnée dans l'opérateur de croisement.
- **La génération de la nouvelle population (reproduction)** : après l'exécution des étapes précédentes, cette étape détermine les individus à survivre et les autres à rejeter. Dans ce travail, nous avons utilisé deux mesures afin de sélectionner la meilleure population. La première mesure utilise la distance euclidienne entre la requête du client et la solution fournie, où la population qui a un individu avec la meilleure distance sera choisie. La deuxième mesure est d'utiliser deux méthodes ; la distance de crowding (voir l'équation 6.3) au cours de l'exécution de l'algorithme [29] et d'appliquer la relation de dominance dans la dernière génération. Dans le cas où nous obtenons des individus compétitifs, on prend la solution ayant la distance minimale.
- **Critère d'arrêt** : dans les algorithmes évolutionnaires, il existe deux conditions d'arrêter. La première est le nombre de génération et la deuxième utilise la fonction de fitness. Dans ce travail, nous avons utilisé le nombre de génération si on n'arrive à aucun changement pendant un nombre défini d'itération on arrête l'algorithme.

6.6.3 Décodage de la solution et exécution du déploiement

Dans cette étape, l'agent travailleur envoi les résultats obtenus à l'agent broker pour décoder la meilleur combinaison. Le but du décodage est de restaurer les paramètres originaux. Ensuite, cet agent envoie les résultats à l'agent interface, qui, à son tour, affiche les résultats auprès du demandeur du service. Après avoir accepté l'offre proposée, le fournisseur de service (client) applique le paiement requis ; puis l'agent broker démarre l'opération de déploiement du service.

Afin de résumer le processus global présenté dans cette section, nous la modélisons avec un diagramme de séquence UML utilisant des agents. Le scénario suivant présenté dans la figure 6.8 donne les interactions entre différents types d'agents dans notre système ; cela commence par la demande du fournisseur et se termine par la réponse.

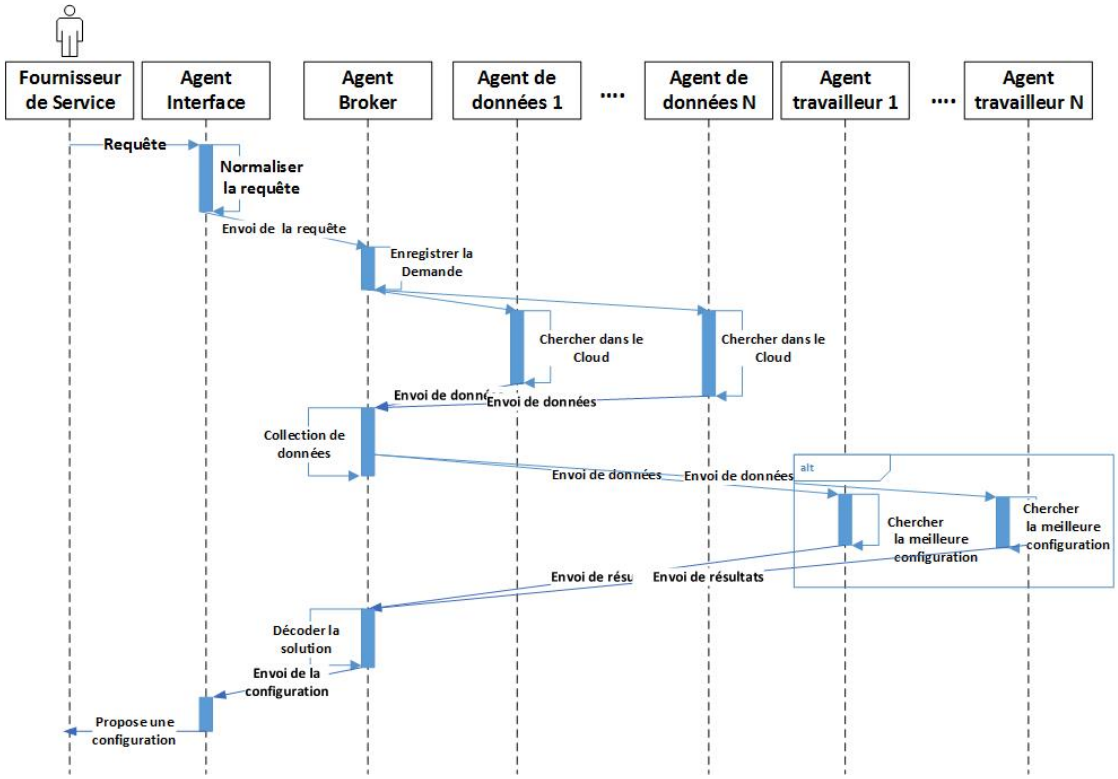

FIGURE 6.8 – Scénario de déploiement de service.

6.7 Conclusion

Le déploiement de services dans le *Cloud Computing* a besoin des opérations de sélection et de compositions de services. Pour cette raison, nous avons besoin de composer une liste de services qui peut répondre aux exigences de l'utilisateur. Puisqu'il existe un nombre très important de services déployés dans le Cloud avec des qualités de services différentes, le problème en main est devenu très complexes. En outre, chaque fournisseur de service utilise son propre modèle de description de service dans le Cloud, cela rend la tâche de découverte de services plus difficile. Afin d'éviter ce problème, dans ce travail, nous avons défini un modèle de description de service qui peut réduire le nombre de services candidats.

Comme nous avons mentionné précédemment, la solution proposée représentée par une architecture multi-agents afin de résoudre le problème de composition. Lorsqu'on a plusieurs QoS pour décrire un service, ce travail transforme le problème comme un problème multi-objectifs. Par l'utilisation d'un algorithme évolutionnaire, on cherche toujours à l'optimum local sans explorer tout l'espace de recherche, de plus, nous avons utilisé *NPGA* afin de réduire le calcul lors de l'exécution de l'algorithme. Comme nous avons vu dans la dernière section, nous avons utilisé deux mesures pour déterminer la solution qui peut donner la meilleur composition.

Chapitre 7

Résultats Expérimentaux et Discussions

7.1 Introduction

Dans le processus du développement, l'implémentation d'un logiciel vient après un enchainement de plusieurs étapes et son but principal est de réaliser un produit capable de résoudre les problèmes posés en utilisant des outils et des algorithmes.

Dans le but de réaliser et valider les idées proposées dans les chapitres 5 et 6, le présent chapitre montre les outils et les configurations utilisées afin de développer ce système.

Dans ce chapitre nous allons présenter les outils et les plateformes utilisés pour développer notre approche. Ensuite, nous présentons quelques interfaces qui montrent les résultats obtenus. Puis, nous présentons une discussion sur les résultats pour chaque évaluation.

7.2 Outils et Plateformes Utilisées

Dans cette section, nous allons présenter les plateformes utilisées afin de réaliser notre système. Les approches proposées dans notre travail sont implémentées sur un ordinateur de processeur Core I3 avec une RAM de 8 Go sous Windows 7, mais on peut implémenter ce système sur n'importe quelle machine grâce à la machine virtuelle du Java. Pour développer les agents, nous avons utilisés la plateforme JADE, grâce aux avantages du cette dernière qui nous offre un langage de communication entre les agents en utilisant le FIPA-ACL. Afin de simuler les idées proposées, nous avons utilisé la plateforme CloudSim pour simuler les ressources virtuelles ainsi que les tâches réalisées dans notre système.

7.2.1 Simulateur Cloud

Afin de développer et exécuter une application dans le Cloud qui représente un problème aux développeurs, puisque les frais des tests sur un environnement Cloud réel comme Amazon EC2 sont très élevés, et pour éviter cette limite, nous avons utilisé des simulations pour l'évaluation de notre approche.

Un simulateur Cloud est un outil qui permet aux développeurs d'utiliser des centres de données avec des serveurs de base et des matériaux informatiques qui sont des processeurs, Disque dur, RAM. Ces éléments sont mis dans le but d'assurer le bon fonctionnement du système afin de simuler la création et la distribution de tâches sur des machines virtuelles.

Pour valider les idées présentées dans les chapitres précédents, nous avons utilisé le simulateur CloudSim.

Plateforme Cloudsim

Cloudsim est un outil qui offre aux développeurs un moyen pour modéliser, simuler et réaliser des expérimentations dans le but de créer une infrastructure Cloud. Cette plateforme permet également aux développeurs de tester leurs services et leurs applications Cloud, sans s'inquiéter avec les détails de bas niveau liés aux infrastructures Cloud [81].

Les principales fonctionnalités de Cloudsim sont [99][104] :

- Supporter la modélisation et la simulation des centres de données Cloud à grandes échelles.
- Supporter la modélisation et la simulation des serveurs virtuels d'hébergement d'application, avec la possibilité de personnaliser la politique de fourniture des ressources d'hébergements aux machines virtuelles.
- Supporter la modélisation et la simulation de ressources virtuelles et logiques pour des tests et assurer l'optimisation de la consommation d'énergie sur le Cloud.
- Supporter la modélisation et la simulation des différentes topologies réseaux, centres de données et l'échange de messages entre applications Cloud.
- Supporter la modélisation et la simulation des systèmes Cloud fédérés.
- Supporter l'insertion dynamique de nouveaux éléments de simulation, avec des capacités d'arrêt et de reprise de simulation.
- Supporter tout type et politique d'allocation de ressources pour la création des machines

virtuelles, et tout type et politique d'allocation de ressources pour l'hébergement d'application et de services Cloud.

CloudAnalyst

Dans le cloud, le problème de localisation représente un problème dans le processus de recherche des services. Pour cette raison, l'intégration de la localisation géographique des centres de données est une tâche très importante afin d'évaluer l'efficacité des services Cloud. Malheureusement, le simulateur Cloudsim ne peut pas assurer l'opération de calculer la distance entre les centres de données et leurs clients qui vont affecter les résultats finaux. CloudAnalys est un outil de simulation qui permet aux développeurs d'exécuter et de tester les différentes simulations de façon reproductible, en prenant en considération les différents paramètres suscités.

CloudAnalys facilite l'interaction et l'utilisation avec les utilisateurs à travers une interface utilisateur graphique (GUI) (voir figure 7.1). De plus, cette interface permet aux développeurs de mettre en place des expériences de façon rapide et simple. En outre, CloudAnalyst permet de séparer entre l'aspect simulation et l'aspect programmation lors du développement des paramètres de chaque expérimentation. Grâce à ces avantages, ce simulateur donne aux développeurs la possibilité de se concentrer sur la complexité de la simulation au lieu de perdre son temps sur les aspects techniques de la programmation à l'aide d'un Toolkit de simulation [136].

FIGURE 7.1 – Le simulateur Cloud Analyst.

7.2.2 Plateforme JADE

Afin d'assurer le bon fonctionnement de notre projet et à cause de la nature de notre architecture qui est basée sur les agents, nous avons utilisé la plateforme de développement des systèmes multi -agent JADE. JADE comporte un ensemble d'outils et d'API qui permettent la construction et la mise en service d'agents sur un contexte bien spécifique.

Présentation générale

JADE est une plateforme implémentée avec le langage JAVA. Ce Framework est destiné aux développeurs qui s'intéressent à créer des systèmes multi-agents, et qui assure un langage de communication prédéfini et représenté dans les standards FIPA-ACL. Parmi les avantages offerts par la plateforme JADE ; l'interopérabilité sans limite pour les applications quelles prennent en charge, et l'indépendance de cette plateforme du système d'exploitation ou du matériel sur laquelle elle est implémentée [11]. JADE est composée de trois principaux modules qui dépendent directement des normes FIPA :

- *DF* : pour Directory Facilitator en anglais, qui sert à la fourniture d'un service de «page jaune» à toute la plateforme JADE.
- *ACC* : pour Agent Communication Channel en anglais, qui représente l'outil de gestion de communication entre agents, pour la plateforme JADE.
- *AMS* : pour Agent Management System en anglais, qui représente l'outil de gestion (authentification, supervision, accès ... etc.) des agents de la plateforme JADE.

Architecture du logiciel

JADE est une plateforme basée sur les standards FIPA, par conséquent l'architecture de cette plateforme est également basée sur l'architecture proposée par FIPA. En outre, le modèle de base de l'architecture de la plateforme JADE proposée par FIPA est intitulé AMRM (Agent Management Reference Model en anglais). Chaque module, qui compose l'architecture de cette plateforme, est présenté sous forme de service, ce qui permet aux agents de bénéficier d'une plateforme orientée service, afin de faciliter la communication et la collaboration entre eux [81].

Les principaux modules qui composent l'architecture JADE sont : DF, AMS et également le MTS (Massage Transport Service en anglais) qui sert de moyen de la communication entre plusieurs plateformes JADE.

Afin d'assurer un fonctionnement efficace des agents sur la plateforme JADE, cette dernière utilise :

- *AID* : pour Agent Identifier en anglais, afin de distinguer et d'identifier chaque agent.
- *DF* : joue le rôle d'un annuaire qui enregistre les compétences de chaque agent. Les pages jaunes fournis par ce service, sont destinées à mettre en relation les différents agents fonctionnels sur la plateforme JADE, et cela pour qu'un agent puisse consulter et interroger ce service, afin d'obtenir des informations sur les compétences des autres agents et par conséquent, d'assurer une bonne collaboration entre eux.
- *AMS* : joue le rôle d'un annuaire pour l'enregistrement des adresses de transport des différents agents de la plateforme. Le but est de fournir un service de «pages blanches» afin de mettre en correspondance les agents avec l'AID, pour faciliter leur contrôle et supervision.

7.3 Configuration et ajustement des paramètres

Dans cette section, nous allons présenter les listes des paramètres utilisées afin d'arriver aux résultats obtenus des deux approches qu'on a proposé dans les chapitres précédents. Ces configurations sont mises pour un but de fixer les valeurs utilisées dans les algorithmes afin d'arriver à des résultats fiables. Comme nous avons mentionné dans les chapitres précédents, les valeurs de qualité de services sont déterminées dans des intervalles définies comme suit : le coût entre [2-15], temps de réponse entre [20-1500], et la fiabilité, réputation, pertinence, disponibilité qui sont présentés respectivement dans les intervalles [0.3-0.9], [0.4-1], [0.2-0.9] et [0.95-1]. D'où, le nombre de services est égal à 100 pour chaque type. Pour le cas du déploiement, nous avons utilisé trois type de services et nous avons mis 10 services pour chacun. Pour les valeurs de l'algorithme NPGA, le nombre de comparaison t_{dom} égal à trois et la valeur du rayon de niche égale à 0.4886. Concernant la requête (les valeurs de qualité de services) du client, nous avons proposé les valeurs suivantes : 20, 1000, 0.8, 0.78, 0.6, et 0.9.

7.4 Présentation des interfaces du système

Dans cette section, nous allons présenter les interfaces utilisées afin de les exploiter dans notre système. En premier temps, nous allons présenter les interfaces qui concernent la composition de service. Par la suite, nous allons introduire les interfaces qui correspondent à l'opé-

ration de déploiement des services dans le Cloud.

La figure 7.2 représente l'interface du client qui permet aux utilisateurs de services de chercher les services dans le Cloud. Cette interface demande aux clients d'introduire sa demande sous une forme textuelle. De plus, nous avons mis des champs pour les qualités de services finaux, ces valeurs permettent, à notre système, de faciliter la tâche de recherche.

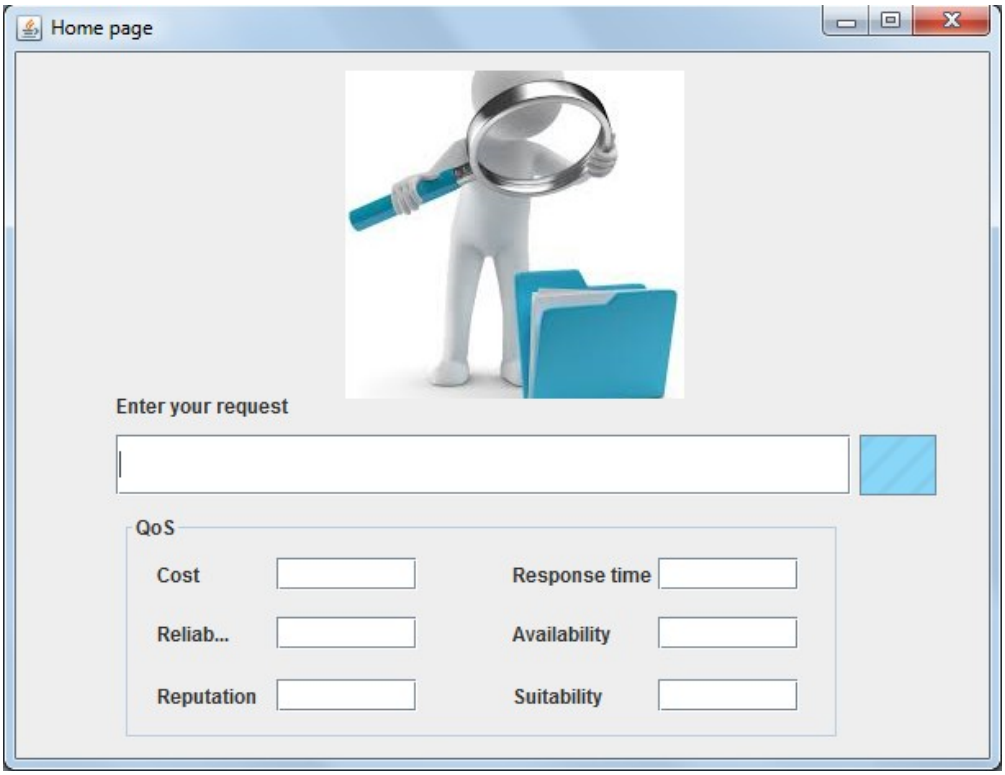

FIGURE 7.2 – Interface client pour la recherche de services.

La figure 7.3 présente l'interface du fournisseur qui lui permet d'introduire ses services en utilisant les champs textuels. En cas de changement de paramètres de service, il suffit d'introduire les informations nécessaires (les données à changer) sans remplir tous les champs.

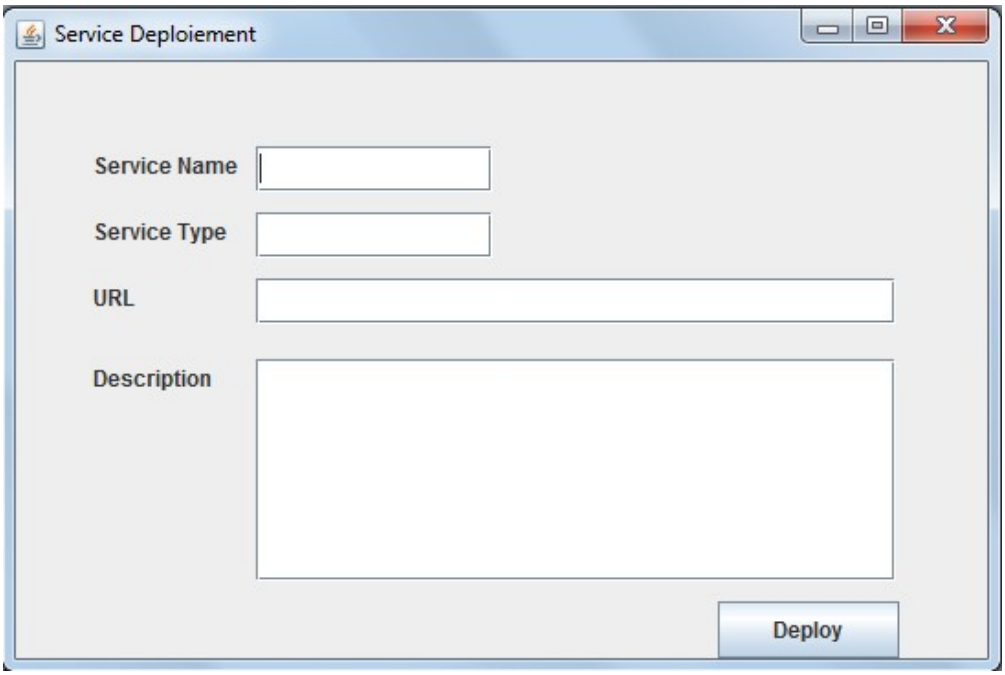

FIGURE 7.3 – Interface du fournisseur pour déployer un service.

Dans la prochaine figure (Figure 7.4), nous allons afficher les résultats obtenues à la fin de l'opération de composition. De plus, cette figure donne une liste de services composites avec les valeurs de qualité de services. Ensuite, le client doit choisir une offre parmi celles proposées selon leur besoins.

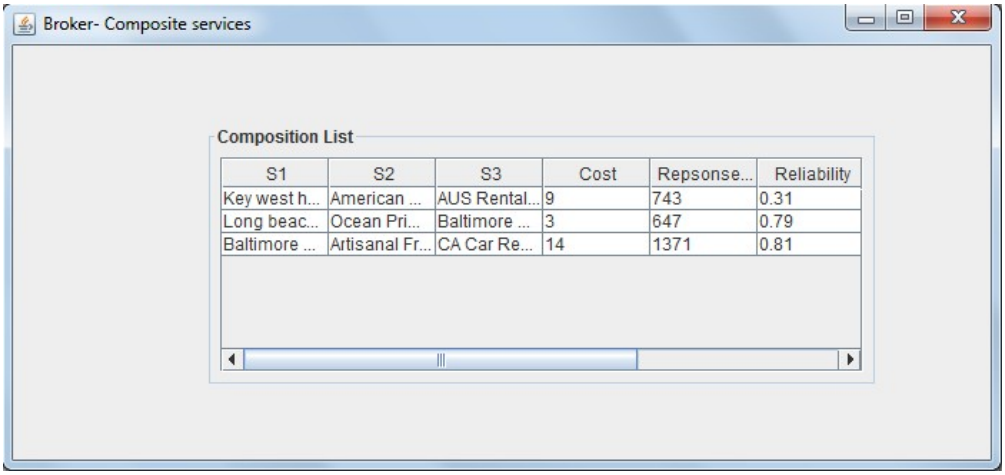

FIGURE 7.4 – Interface affichant les résultats collectés à la fin de l'opération.

Après la présentation des figures qui concernent la validation de l'approche présentée dans

le chapitre 5. Les figures qui suivent, montrent les interfaces utilisées pour développer l'approche illustrée dans le chapitre 6 qui consiste à l'opération de déploiement de services. L'interface présenté dans la figure 7.5 donne pour un client, qui est représenté dans notre cas, un fournisseur de service afin d'introduire les informations nécessaires de service à déployé. Le fournisseur de service doit introduire les informations sur le service qu'il veut déployer avec les valeurs de QoS qu'il préfère. Ensuite, le système va commencer à chercher les services qui peuvent répondre aux exigences.

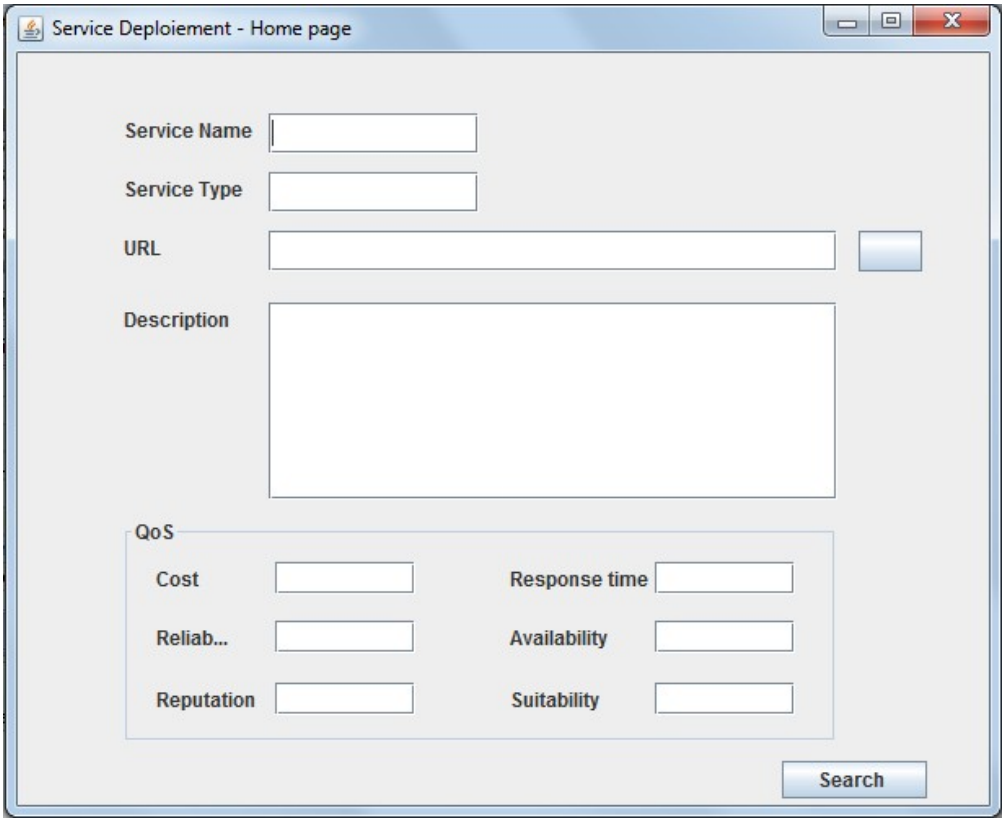

FIGURE 7.5 – Interface pour déployer un service.

Après l'étape précédente, le système va proposer aux fournisseurs un ensemble de services composites avec des distances pour faciliter le choix de service composite approprié, comme indique la figure 7.6.

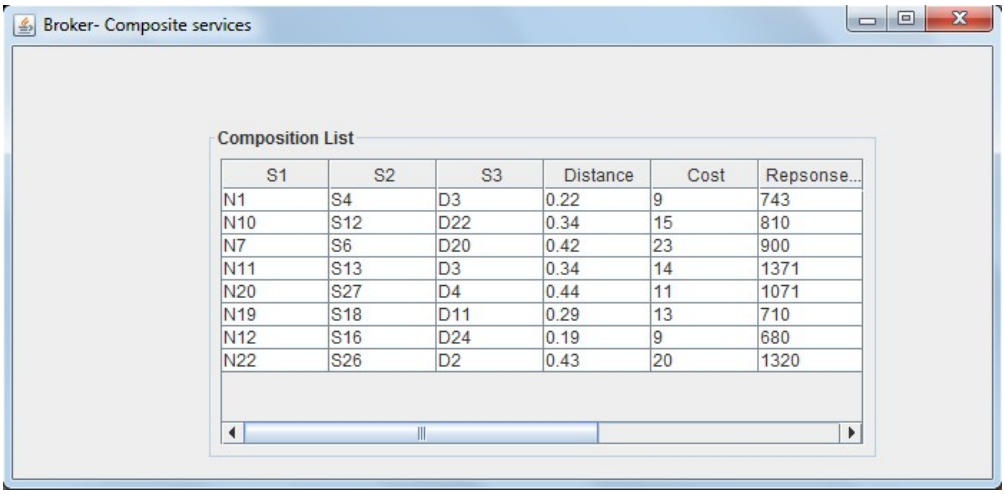

FIGURE 7.6 – La liste des services composites à la fin de l'algorithme.

7.5 Résultats obtenus et discussions

Dans cette section, nous allons présenter les résultats obtenus à la fin de chaque méthode proposée. On commence par la présentation des courbes obtenues depuis l'exécution de l'algorithme de composition avec le changement des paramètres au niveau de services candidats et le cas normal (sans changement des paramètres). Les autres résultats correspondent à l'approche qui manipule l'opération du déploiement des services dans le Cloud. La courbe présentée dans la figure 7.7, montre les résultats obtenus lors de l'exécution de l'algorithme 2 présenté dans le chapitre 5 pour dire l'efficacité de notre approche nous avons comparé avec d'autres algorithmes de base qui sont l'algorithme linéaire et l'utilisation de la méthode aléatoire définit comme suit :

- **La méthode linéaire** : l'idée de cette méthode est de trier les ressources (machines virtuelles et serveurs) dans une liste, puis rechercher séquentiellement pour la prochaine ressource disponible qui peut gérer la tâche en cours.
- **La méthode aléatoire** : dans cette méthode, le système de surveillance des ressources sélectionne de manière aléatoire une ressource à partir d'une liste de ressources disponibles qui peut satisfaire la tâche - la machine sélectionnée (machine virtuelle) pour notre étude peut ne pas être la meilleure.

Les résultats obtenus sont présenté comme suit :

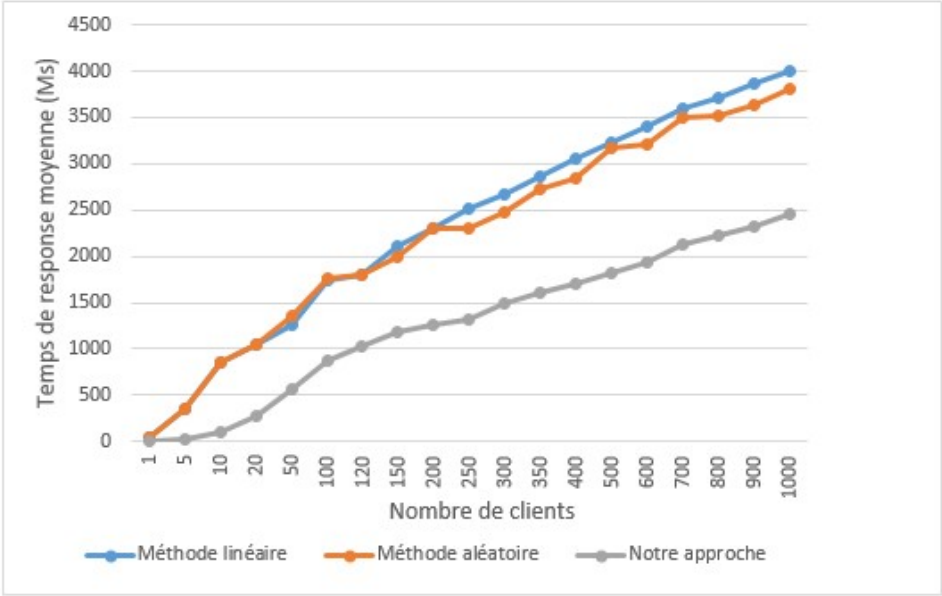

FIGURE 7.7 – Temps de réponse moyenne pour chaque méthode.

La figure précédente (7.7) décrit le temps de réponse moyen pour un nombre de requêtes (tel qu'il est indiqué dans l'axe des nombres de clients). Depuis la figure 7.7, on peut conclure que : (i) quand le nombre de requête varie entre 1 et 10, le temps de réponse moyen dans notre méthode s'incrémente de façon lente par rapport aux autres méthodes. (ii) Le temps de réponse tend à augmenter rapidement lorsque le nombre de clients varie de 10 à 100 ; et (iii) De 100 à 200, la tendance des temps de réponse à augmenter devient plus modérée au début (de 100 à 120) que l'intervalle précédent, mais augmente encore à partir de 120. La raison pour laquelle le graphique prend cette forme globale est que notre approche crée des machines virtuelles dynamiquement, comme elles sont requises, et ce n'est pas le cas pour les autres méthodes. En outre, la communication des agents réduit le nombre de messages échangés contenant les meilleures mesures de QoS. Par conséquent, nous pouvons conclure que notre méthode offre de tels services composites dans un délai de réponse approprié, par rapport aux autres méthodes (linéaire et aléatoire).

Dans notre travail, nous ajoutons une autre expérience qui consiste dans l'architecture adaptative. Les résultats obtenus montrent la différence entre le temps de réponse moyen des clients dans deux cas. Le premier est obtenu à partir de la composition du service dans un scénario normal (sans mise à jour du service en graphique) et le deuxième cas est obtenu pour le scéna-

rio impliquant des changements dans les paramètres du service pendant le processus de composition du service (comme mentionné précédemment, ces changements apparaissent lors du choix les voisins). La Figure suivante donne les résultats obtenus :

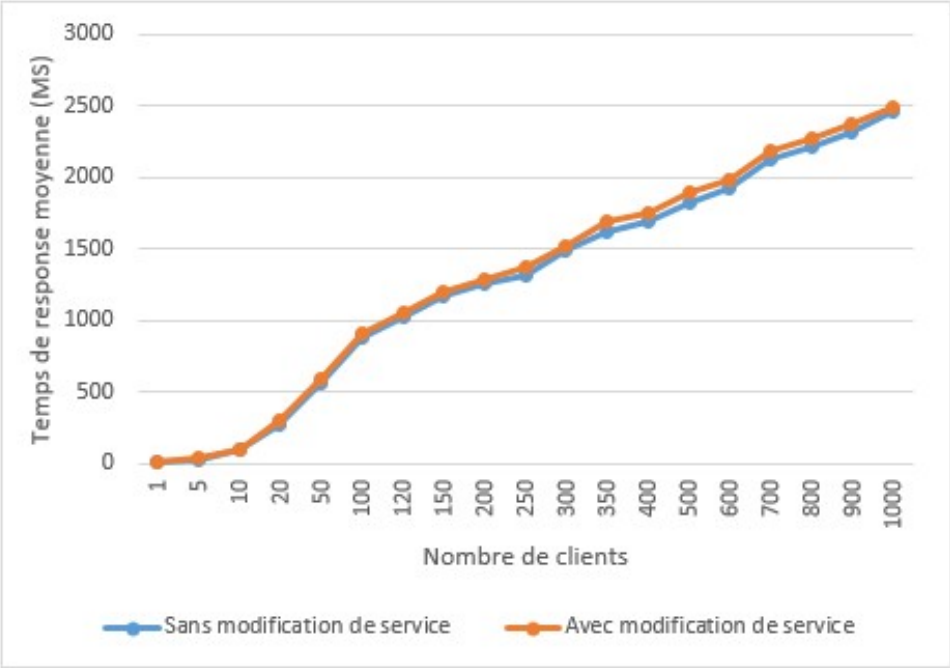

FIGURE 7.8 – Comparaison entre le changement des paramètres et le cas normal.

Le graphe précédent montre les résultats de mise en œuvre de notre algorithme et illustre la comparaison entre les compositions de service avec les services mis à jour et sans mises à jour de service. Nous voyons que la différence entre les deux formes est acceptable pour un nombre important de clients. Nous pouvons conclure que l'efficacité de notre algorithme peut satisfaire les besoins du demandeur de service.

Pour la deuxième approche qui assure le déploiement de services dans le Cloud, nous avons utilisé le modèle de description de services de type DaaS afin de réduire le nombre de services candidats. La figure (Figure 7.9) suivante montre l'utilisation du modèle qui utilise le degré de similarité entre la requête du client avec les services déployés.

Pour calculer le degré de similarité nous avons utilisé l'équation illustrée dans (équation 7.2) et avant d'utiliser cette équation nous avons utilisé la formule de tf-idf pour calculer le poids de chaque caractéristique donné dans l'équation 7.1 [103] :

$$W(terme) = tf - idf(terme) = tf(terme) \times \log_2(\frac{N}{df(terme)}) \tag{7.1}$$

W est le terme poids, *tf* est le terme fréquence définie le nombre d'occurrences du terme dans le document (WSDL), N est le nombre de documents dans la collection, et *df* est la fréquence du document qui donne le nombre de fois que le terme apparaît dans les autres documents de la collection. Après que le client présente sa demande, nous la transformons en un vecteur de poids en utilisant l'équation 7.1. Ensuite, nous utiliserons la fonction de similarité pour sélectionner le service approprié parmi les autres [126].

$$Sim(r,d) = \frac{\vec{r}.\vec{d}}{|\vec{r}| \times |\vec{d}|} = \frac{\sum_{i=1}^{N} r_i \times d_i}{\sqrt{\sum_{i=1}^{N} r_i^2} \times \sqrt{\sum_{i=1}^{N} d_i^2}} \tag{7.2}$$

Où $|\vec{r}|$ et $|\vec{d}|$ sont des normes de la demande et des vecteurs de document, N nombre de termes dans les vecteurs. Puisque $r_i \geq 0$ et $d_i \geq 0$, Sim (r,d) varie de 0 à +1 et montre le degré de similitude entre une requête r et un document d.

FIGURE 7.9 – Temps de réponse en utilisant le modèle proposé.

Comme on le voit dans la figure précédente qui montre le temps de réponse à un nombre donné de clients en utilisant notre modèle. De la figure 7.9, nous pouvons observer que les don-

nées obtenues sont transmis aux clients dans un délai raisonnable. De plus, à partir de figure 7.9 on peut voir qu'il y a une variation des valeurs de temps de réponse à 100, 200, 350 (nombre de requêtes). Les modifications apportées au graphique dépendent d'un certain nombre de services sélectionnés pour chaque demande (par exemple, la demande numéro 1 correspond à 10 services de données, la deuxième demande correspond à 5 services de données).

Après l'utilisation de l'algorithme pour résoudre le problème du déploiement de service dans le Cloud, nous présentons le tableau suivant qui présentent les résultats collectés après l'exécution de l'algorithme.

TABLE 7.1 – Résultats des valeurs de QoS^A moyennes obtenues de chaque approche

Méthode	**Itération**	**Coût**	**Temps de réponse**	**Fiabilité**	**Réputation**	**Pertinence**	**Disponibilité**
Aléatoire	1	18.42	1074.44	0.61	0.64	0.46	0.97
Dominance	1	21.26	1114.2	0.63	0.62	0.48	0.95
	50	22.6	1220.5	0.70	0.68	0.54	0.97
	100	23.38	1339.02	0.73	0.67	0.53	0.97
	150	24.16	1308.3	0.71	0.7	0.56	0.97
	200	22.62	1285	0.72	0.7	0.53	0.97
	250	23.4	1236.38	0.73	0.71	0.55	0.96
	300	23.32	1318.6	0.72	0.7	0.53	0.97
Distance	1	22.32	1094.38	0.69	0.58	0.5	0.97
	50	22.2	1327.68	0.73	0.68	0.6	0.97
	100	22.56	1353.8	0.73	0.69	0.59	0.97
	150	22.86	1342.8	0.72	0.69	0.61	0.97
	200	23.08	1242.4	0.74	0.7	0.6	0.97

Comme nous l'avons mentionné précédemment, nous évaluons notre algorithme en utilisant la distance euclidienne entre la solution obtenue et la demande du client. La Figure suivante (Figure 7.10) montre la distance Euclidienne moyenne pour chaque génération.

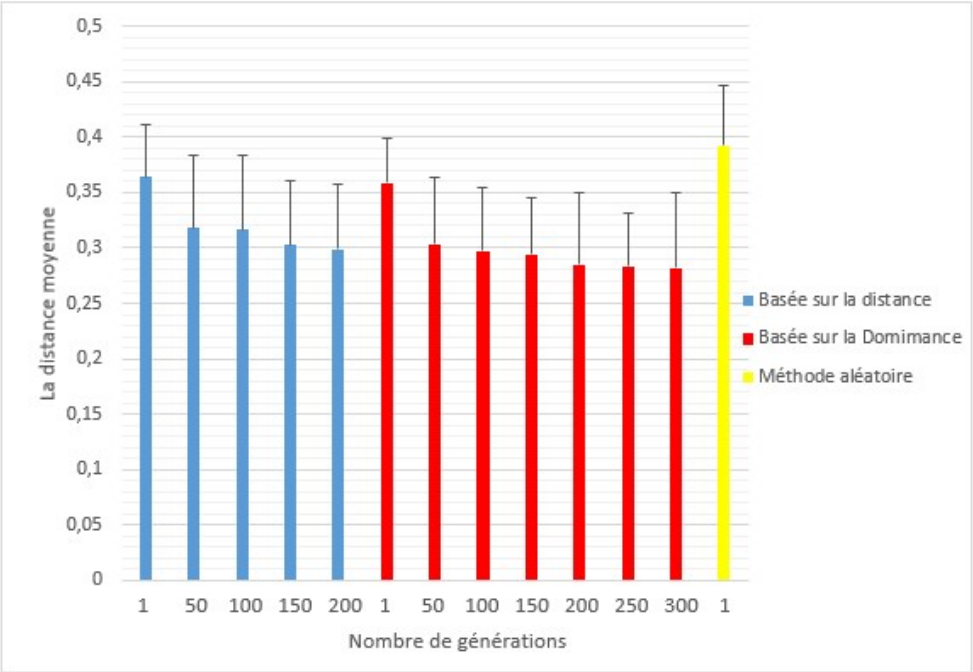

FIGURE 7.10 – La distance moyenne de chaque approche.

Comme nous avons vus dans la figure précédente, la méthode aléatoire représentée dans la dernière barre (couleur jaune) montre une distance égale à 0.39 par rapport à la demande de la requête. En outre, on peut affirmer que l'algorithme utilisé fournit de meilleures solutions par rapport à l'algorithme de ligne de base (méthode aléatoire). Comme le montre la figure, les valeurs bleues présentent la NPGA évaluée à distance Euclidienne. La première génération montre une distance moyenne de 0.37 par rapport à la demande. Les prochaines expériences réalisent 50 et 100 générations respectivement, et elles obtiennent presque les mêmes résultats (distance moyenne de 0.32 pour 100 générations). Lorsque nous augmentons le nombre de générations, la méthode s'améliore (le résultat est réduit à une distance de 0.3) et la solution qu'ils obtiennent est presque stabilisée entre 150 et 200 générations. La méthode suivante, qui combine la distance d'agglomération avec les relations de dominance est montrée dans les barres rouges. Les valeurs obtenues sont meilleures que celles dérivées de la précédente. Il est évident que dans la première génération la distance moyenne est égale à 0.36 en première génération. Pour les prochaines générations, de 50 à 200, il diminue linéairement. À la fin, le graphe montre qu'avec cette méthode, la valeur atteint une stabilisation entre 250 et 300 générations avec une distance moyenne égale à 0.28.

7.6 Conclusion

La composition de services dans le Cloud est le problème épineux à cause de la diversité du nombre exponentiel des services déployés dans le Cloud. Afin de déterminer l'efficacité d'un système, nous devons l'utiliser plusieurs fois pour dire que le système est fiable. Au cours de ce chapitre, nous avons présenté les outils et les plateformes utilisées afin d'implémenter le système de composition de services dans le Cloud. Ensuite, nous avons présenté les configurations utilisées dans notre approche, cela peut aider l'application de fournir des résultats fiables. Par conséquent, ceci peut rendre les clients satisfait de ces exigences. Puis, nous avons montré quelques interfaces sur notre application suivie par des explications pour chacune. Après, nous avons exposé quelques formes graphiques pour montrer l'efficacité de notre approche que ce soit pour la composition ou dans le cas de déploiement de service suivi par des discussions sur les résultats collectés.

Chapitre 8

Conclusion Générale et Perspectives

8.1 Conclusion

La composition de service dans le Cloud est un problème car les services déployés sont nombreux et de plus ils augmentent exponentiellement. Suite à ce nombre important des services dans le Cloud, ce problème est classé comme étant un problème *NP-complet*. Avant de répondre aux exigences des utilisateurs qui sont de différents types, nous avons besoin de proposer une approche qui résout ce problème par l'utilisation de la composition de services. Lorsqu'on commence l'étape de composition de services, parfois, les fournisseurs de services peuvent introduire de nouvelles modifications au niveau des services déployés. Dans la vision traditionnelle qui utilise le service web, le système peut donner un échec à la fin de l'opération. Par contre, lorsqu'on remplace la notion de service web par un agent, on peut résoudre ce problème.

Notre solution présentée dans cette thèse est basée sur l'utilisation un système multi agents qui peut résoudre le problème lorsqu'on est dans une situation de changement de paramètres (autonomie de l'agent). Dans cette thèse, nous avons proposé une architecture qui résout le problème de déploiement de service dans le Cloud. Cette approche peut offrir aux propriétaires de Cloud une stratégie pour faciliter l'opération de déploiement par l'utilisation d'un modèle mentionné précédemment dans les chapitres.

Dans cette thèse, nous avons fait, en premier temps, un survol sur l'état de l'art qui consiste à présenter les définitions sur la technologie de Cloud, et nous avons aussi discuté le Cloud qui tend à résoudre les problèmes posés par rapport aux autres technologies. De plus, le Cloud représente une révolution car il offre des services de plusieurs types. Grace aux bénéfices offerts

de la part du fournisseur de Cloud, les utilisateurs finaux et les entreprises trouvent que le Cloud est le bon choix pour l'utilisation des services.

Le troisième est une présentation des technologies qui permettent aux utilisateurs de connaître les technologies existantes, ainsi qu'une comparaison entre ces technologies. De plus, ce chapitre présente les modèles de description de services qui représentent le noyau de n'importe quel type de technologie. En outre, afin de choisir un service, nous avons cité quelques qualités de service qui servent à évaluer le service final. Du point de vue d'un concepteur, on ne peut pas dire qu'une technologie est meilleure qu'une autre, cela dépend des problèmes rencontrés et des besoins d'utilisateurs.

Afin de montrer l'intérêt de notre travail, dans le quatrième chapitre, nous avons donné une présentation avec une synthèse sur les travaux réalisés pour la résolution du problème de composition de services. Nous avons illustré les solutions du problème dans la littérature à travers deux cas (dans le cas où la composition se fait dans un environnement Cloud ou bien hors le Cloud Computing). Parmi les travaux réalisés, il existe des approches qui ne tenaient pas en compte de quelques points lors de la résolution de problème de composition tel que : l'aspect non-fonctionnel manquant ou bien ces solutions qui ne traitent pas le cas où le changement de paramètres lors du processus de composition de services.

Afin de résoudre le problème posé, le cinquième chapitre comporte une architecture que nous avons proposée qui est basée sur un ensemble d'agents. L'idée vise à proposer un protocole de coopération entre les agents afin de résoudre le problème. Comme nous avons vu, lorsque nous voulons fournir un service composite avec des mesures de QoS appropriées, des problèmes surviennent, ils concernent le conflit entre les services. L'idée derrière ce travail était de présenter l'interval de confiance entre les agents pour qu'ils choisissent les agents avec lesquels ils veulent coopérer. Dans le but de résoudre le conflit entre les offres, l'utilisation de la relation de dominance nous encourage à confirmer les meilleurs choix parmi le reste grâce aux paramètres QoS. En outre, ce travail peut gérer les changements des paramètres lors de la composition du service.

Comme nous avons mentionné, précédemment, la solution proposée dans le sixième chapitre est représentée par une architecture multi-agents afin de résoudre le problème de la composition. Lorsqu'on a plusieurs QoS pour décrire un service, ce travail transforme le problème

comme un problème multi-objectifs. En utilisant un algorithme évolutionnaire, on cherche l'optimum local sans explorer tout l'espace de recherche. De plus, nous avons utilisé NPGA afin de réduire le calcul lors de l'exécution de l'algorithme. Comme nous l'avons vu dans la dernière section, nous avons utilisé deux mesures pour déterminer la solution qui peut donner la meilleure composition.

Afin de valider les approches utilisées, dans le septième chapitre, nous avons montré les outils utilisés pour arriver à des résultats. En outre, nous avons défini les paramètres utilisés pour ajuster les algorithmes utilisés dans les différentes approches. Nous avons montré aussi quelques interfaces qui affichent les opérations du système proposé. Pour finaliser la mise en œuvre, nous avons introduit une discussion sur les résultats obtenus afin d'analyser les résultats collectés.

8.2 Perspectives

Afin de continuer ce travail, nous pensons à introduire des points dans notre travaux future ou pour les chercheurs dans ce domaine. Nous avons collecté ces points selon l'ordre suivant :

- Dans les approches utilisées, il existe de nombreuses ressources utilisées pour exécuter notre proposition dans le Cloud. Dans le travail futur, nous pensons proposer un algorithme qui assure la bonne gestion de ces ressources afin de réduire la consommation des ressources dans le Cloud.
- Introduire d'autres valeurs de qualité de service dans le processus de composition de services dans le but d'améliorer les résultats.
- Nous pensons aussi à proposer une solution basée sur les enchères combinatoires et de les résoudre par des algorithmes évolutionnaires pour faire une comparaison entre le travail que nous avons réalisé.
- Nous pensons à intégrer l'aspect sémantique dans les modèles de description proposées pour réduire le nombre de services candidats.
- Nous pensons à proposer un langage de description de service Cloud afin de faciliter la tache de la découverte.

Bibliographie

[1] Adrian Mocan, Emilia Cimpian, and Mick Kerrigan, "Formal Model for Ontology Mapping Creation". International Semantic Web Conference. 2006. Springer.

[2] Alonso, G., Casati, F., Kuno, H., & Machiraju, V., *Web Services - Concepts, Architectures and Applications*, Springer Verlag 2004.

[3] Amaia Lazcano, Gustavo Alonso, Heiko Schuldt, C. Schuler, "The WISE approach to electronic commerce", J. Comput. Syst. Sci. Eng. 15 (5) (2000) 345–364.

[4] Androcec, D., Vrcek, N., and Seva, J., "Cloud computing ontologies : a systematic review", In Proc. the 3rd International Conference on Models and Ontology based Design of Protocols, Architectures and Services', 2012, pp. 9-14.

[5] Anton Michlmayr, Florian Rosenberg, Philipp Leitner, Schahram Dustdar, "End-to-end support for QoS-aware service selection, binding, and mediation in VRESCo", IEEE Trans. Serv. Comput. 3 (3) (2010) 193–205

[6] Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M, "Above the clouds : a Berkeley view of cloud computing". Tech, Rep UCB/EECS-2009-28, EECS Dept, Uni of California, Berkeley ('09).

[7] Brahmi, Z., & Gammoudi, M. M. (2013, June). "QoS-aware automatic web service composition based on cooperative agents". In IEEE 22nd International Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE), 2013 (pp. 27-32). IEEE.

[8] Barry, D. K., Dick, D., Web Services, Service-Oriented Architectures, and Cloud Computing, 2nd edition, Elsevier, 2013.

[9] Bart Orriëns, Jian Yang, Mike P. Papazoglou, "A rule driven approach for developing adaptive service oriented business collaboration", in : Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC'05), Amsterdam, The Netherlands, December 2005, pp. 61–72

[10] Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). Developing multi-agent systems with JADE. Vol. 7. John Wiley & Sons.

[11] Bellifemine, F., Poggi, A. and Rimassa, G. (1999) "JADE—a FIPA compliant agent framework", Proceedings of the 4th international conference and exhibition on the practical application of intelligent agents and multi-agents, UK, pp. 97–108.

[12] Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R., "Heuristics for QoS-aware Web Service Composition",IEEE, International Conference on Web Services ICWS'06 ,pp. 72 – 82, December 2006.

[13] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. "A taxonomy and survey of cloud computing systems". Fifth International Joint Conference on INC, IMS and IDC, 2009.

[14] Bhattacharjee, R. (2009). "An analysis of the cloud computing platforms", MIT Masters Thesis.

[15] Buyya, R., Nroberg, J., Godcinski, A, *CLOUD COMPUTING Principles and Paradigms*, John Wiley & Sons, 2011.

[16] Canfora, G., Di Penta, M., Esposito, R., Luisa Villani, M., "An Approach for QoS-aware Service Composition based on Genetic Algorithms", Proceeding GECCO '05 Proceedings of the conference on Genetic and evolutionary computation ACM, New York, 2005.

[17] Caryer, G., Rings, T., Gallop, J., Schulz, S., Grabowski, J., Stokes-Rees, I., & Kovacikova, T. (2009, October). Grid/cloud computing interoperability, standardization and the Next Generation Network (NGN). In Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on (pp. 1-6). IEEE.

[18] Chana, I., & Singh, S. (2014). "Quality of service and service level agreements for cloud environments : Issues and challenges". In Cloud Computing (pp. 51-72). Springer International Publishing.

[19] Chatterjee, S., Webber, J., *Developing Enterprise Web Services*. Edition Prentice Hall PTR, le 14 November 2003.

[20] Chuanrong Zhang, Tian Zhao, Weidong Li, "Geospatial Semantic Web", Springer, 11 juin 2015.

[21] Chunqing, C., Shixing, Y., Guopeng, Z., Bu Sung, L., & Singhal, S. (2012). "A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises". In IEEE 5th International Conference on Cloud Computing (CLOUD), 2012, (pp. 883–890)

[22] Collette, Y., & Siarry, P. (2011). *Optimisation multiobjectif : Algorithmes*. Editions Eyrolles.

[23] Coria, J. A. G., Castellanos-Garz?n, J. A., & Corchado, J. M. (2014). "Intelligent business processes composition based on multi-agent systems". Expert Systems with Applications, 41(4), 1189-1205.

[24] Costa, P., Migliavacca, M., Pietzuch, P., & Wolf, A. L. (2012, April). "NaaS : Network-as-a-Service in the Cloud". In Hot-ICE.

[25] D'Inverno, M., Luck, M., & Luck, M. M. (2004). *Understanding agent systems*. Springer Science & Business Media.

[26] Daniela Berardi, Fahima Cheikh, Giuseppe De Giacomo, Fabio Patrizi, "Automatic service composition via simulation", Int. J. Found. Comput. Sci. 19 (2) (2008) 429–451.

[27] Danilo Ardagna, Luciano Baresi, Sara Comai, Marco Comuzzi, Barbara Pernici, "A service-based framework for flexible business processes", IEEE Softw.28 (2) (2011) 61–67.

[28] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

[29] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multi-objective genetic algorithm : NSGA-II", IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[30] Debra VanderMeer, Anindya Datta, Kaushik Dutta, Helen Thomas, Krithi Ramamritham, Shamkant B. Navathe, "FUSION : a system allowing dynamic Web service composition and automatic execution", in : Proceedings of the IEEE International Conference on E-Commerce (CEC'03), IEEE Computer Society, Washington, DC, USA, 2003, pp. 399–404

[31] Dohare, U., & Lobiyal, D. K. (2017, February). "Coalition game theory based multi-metric routing in wireless adhoc networks". In Proceedings of the International Conference on Communication and Computing Systems (ICCCS 2016), Gurgaon, India, 9-11 September, 2016 (p. 389). CRC Press.

[32] Drew V. McDermott, "Estimated-regression planning for interactions with Web services", in : Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS 2002), AAAI Press, 2002

[33] Duan, Q. (2011, July). "Modeling and performance analysis on network virtualization for composite network-cloud service provisioning". In S, 2011 IEEE World Congress on ervices (SERVICES)(pp. 548-555). IEEE.

[34] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, Dana Nau, "HTN planning for Web service composition using SHOP2", J. Web Seman. 1(4), (2004), pp. 377–396

[35] Fabio Casati, Ming-Chien Shan, "Dynamic and adaptive composition of e-services", Inform. Syst. 26 (3) (2001) 143–162.

[36] Fensel, D., Lausen, H., Polleres, A., De Bruijn, J., Stollberg, M., Roman, D., & Domingue, J. (2006). "Enabling semantic web services : the web service modeling ontology". Springer Science & Business Media.

[37] Foster, I., Zhao, Y., Raicu, I. & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop. GCE '08, 10-56.DOI : 10.1109/GCE.2008.4738445

[38] Bao, H. H., & Dou, W. C. (2012). "A QoS-aware service selection method for cloud service composition". In 2012 IEEE 26th international parallel and distributed processing symposium workshops & Phd Forum(pp. 2254–2261). New York : IEEE

[39] Google App Engine.http ://appengine.google.com

[40] Gutierrez-Garcia, J. O., & Sim, K. M. (2010). "Agent-based service composition in cloud computing". Grid and distributed computing, control and automation, 1-10.

[41] Gutierrez-Garcia, J. O., & Sim, K. M. (2010, November). "Self-organizing agents for service composition in cloud computing". In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 59-66). IEEE.

[42] Hamza, S. Okba, K. "Découverte de services web via le Cloud computing à base d'agents mobiles", thèse de doctorat, Université de Biskra, 2015.

[43] Hao, Y., Zhang, Y., & Cao, J. (2010). "Web services discovery and rank : An information retrieval approach". Future Generation Computer Systems, 26(8), 1053-1062.

[44] Hashemi, S. M., & Bardsiri, A. K. (2012). "Cloud computing vs. grid computing". ARPN journal of systems and software, 2(5), 188-194.

[45] Hassina Nacer, Djamil Aissani, "Semantic web services : Standards, applications, challenges and solutions", journal of network and computer applications 44, 2015, pp.134-151.

[46] Hilley D (2009) Cloud computing : a taxonomy of platform and infrastructure-level offerings. Tech Rep GIT-CERCS-09-13,CERCS, Georgia Institute of Technology.

[47] Höfer, C. N., & Karagiannis, G. (2011). "Cloud computing services : taxonomy and comparison". Journal of Internet Services and Applications, 2(2), 81-94.

[48] Horn J., Nafpliotis N., and Goldberg D.E., " A niched Pareto genetic algorithm for multiobjective optimization", Proceedings of the First IEEE Conference on Evolutionary Computation, pp82-87, 1994.

[49] Hossain, M. S., Hassan, M. M., Al Qurishi, M., & Alghamdi, A. (2012, July). "Resource allocation for service composition in cloud-based video surveillance platform". In Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on (pp. 408-412). IEEE.

[50] http ://www-03.ibm.com/software/products/us/en/business-process-manager-family

[51] http ://www.oracle.com/technetwork/middleware/bpel/overview

[52] http ://www.microsoft.com/en-us/biztalk/default.aspx

[53] http ://help.sap.com/nwpi

[54] http ://www.activevos.com

[55] http ://ode.apache.org

[56] http ://www.jboss.org/jbpm

[57] https ://www.w3.org/2005/Incubator/usdl/ consulté le 20/09/2017

[58] Huhns, M. N. (2002). "Agents as Web services". IEEE Internet computing, 6(4), 93-95.

[59] Garg, S.K., Versteeg, S. and Buyya, R. "A Framework for ranking of cloud computing services", Future Generation Computer system, Vol . 29, No. 2, pp. 1012–1023, 2013.

[60] Gustavo, A., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web services : concepts, architectures and applications*. Springer.

[61] Garg, S. K., Versteeg, S., & Buyya, R. (2013). "A framework for ranking of cloud computing services". Future Generation Computer Systems, 29(4), 1012-1023.

[62] Jayashree, K., and Anand, S., "Web Service Diagnoser Model for managing faults in web services", Computer Standards & Interfaces 36 (2013) 154 – 164.

[63] Jeffrey J.P.T. et Lu. M (2006) *Security Modeling and Analysis of Mobile Agent Systems*, SERIES IN ELECTRICAL AND COMPUTER ENGINEERING, World Scientific Publishing Company, ISBN-13 : 978-1860946349.

[64] Jeffrey D. Ullman, Jennifer Widom, *A First Course in Database Systems*, Prentice-Hall, 1997

[65] Jian Yu, Jun Han, Quan Z. Sheng, Steven O. Gunarso, "PerCAS : an approach to enabling dynamic and personalized adaptation for context-aware services", in : Proceedings of the 10th International Conference on Service-Oriented Computing (ICSOC 2012), Springer-Verlag, Berlin, Heidelberg, 2012, pp.173–190

[66] John A. Miller, Devanand Palaniswami, Amit P. Sheth, Krys J. Kochut, Harvinder Singh, "WebWork : METEOR's Web-based workflow management system", J. Intell. Inform. Manage. Syst. 10 (2) (1998) 185–215

[67] Joyce El Haddad, Maude Manouvrier, Marta Rukoz, "TQoS : transactional and QoS-aware selection algorithm for automatic Web service composition", IEEE Trans. Serv. Comput. 3 (1) (2010) 73–85

[68] Jula, A., Othman, Z., & Sundararajan, E. (2013, April). "A hybrid imperialist competitive-gravitational attraction search algorithm to optimize cloud service composition". In Memetic Computing (MC), 2013 IEEE Workshop on (pp. 37-43). IEEE.

[69] Jula, A., Sundararajan, E., & Othman, Z. (2014). "Cloud computing service composition : A systematic literature review". Expert Systems with Applications, 41(8), 3809-3824.

[70] Kandukuri, B, R., Paturi. V. R., Rakshit, A, "Cloud security issues", IEEE International Conference on Services Computing, 2009.

[71] Karim, R., Ding, C., & Miri, A. (2013). "An end-to-end QoS mapping approach for cloud service selection". In 2013 IEEE Ninth World Congress on Services, June 2013 (pp. 341-348). IEEE.

[72] Karunamurthy, R., Khendek, F., and Roch H. Glitho., "A novel architecture for Web service composition", Journal of Network and Computer Applications 35 (2012) 787 – 802.

[73] Keita Fujii, Tatsuya Suda, "Semantics-based context-aware dynamic service composition", ACM Trans. Auton. Adapt. Syst. 4 (2) (2009) 12 :1–12 :31

[74] Kofler, K., ul Haq, I., & Schikuta, E. (2009, September). "A parallel branch and bound algorithm for workflow QoS optimization". In Parallel Processing, 2009. ICPP'09. International Conference on (pp. 478-485). IEEE.

[75] Lee-post, A, Pakath, R., *Cloud Computing : A Comprehensive Introduction, Advances in Business Information Systems and Analytics*, chapter. IGI Global, 2014.

[76] Lécué, F., Gorronogoitia, Y., Gonzalez, R., Radzimski, M., Villa, M., "SOA4All : an innovative integrated approach to services composition", in : Proceedings of the 8th IEEE International Conference on Web Services (ICWS'10), IEEE Computer Society, Washington, DC, USA, 2010, pp. 58–67

[77] Lee, J., Lee, S. J., Chen, H. M., & Wu, C. L. (2012). Composing web services enacted by autonomous agents through agent-centric contract net protocol. Information and Software Technology, 54(9), 951-967.

[78] Liu, M., Wang, M. R., Shen, W. M., Luo, N., & Yan, J. W. (2012). "A quality of service (QoS)-aware execution plan selection approach for a service composition process". Future Generation Computer Systems-the International Journal of Grid Computing and Escience, 28(7), 1080–1089.

[79] Ludwig, S. A. (2012, June). "Clonal selection based genetic algorithm for workflow service selection". In IEEE Congress on Evolutionary Computation (CEC), 2012 IEEE Congress on (pp. 1-7). IEEE.

[80] Lyon, A., Wintle, B. C., & Burgman, M. (2015). "Collective wisdom : Methods of confidence interval aggregation". Journal of Business Research, 68(8), 1759-1767.

[81] Maes, P. (1994) "Agents that reduce work and information overload", Communications of the ACM, Vol. 37, No. 7, pp. 30-40.

[82] Mahmood, Z., *Cloud computing challenges, Limitations and R& D Solutions*. Springer, 2014.

[83] Manne, J. R. (2015). *Multiobjective Optimization in Water and Environmental Systems Management-MODE Approach*. Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering, 120.

[84] Markus Keidl, Alfons Kemper, "Towards context-aware adaptable Web services", in : Proceedings of the 13th International World Wide Web Conference (WWW'04), ACM Press, New York, NY, USA, 2004, pp. 55–65

[85] Mell P, Grance T (2009) The NIST definition of cloud computing (v15). Tech Rep, National Institute of Standards and Technology.

[86] Mell P, Grance T (2011). "The NIST Definition of Cloud Computing". In N.I.o.S.a.Technology (Ed.) : U.S. Department of Commerce.

[87] Menken, I., *Cloud Computing - The Complete Cornerstone Guide to Cloud Computing Best Practices*. Emereo Pty Ltd, 2008. Pp. 6.

[88] Merizig, A., Kazar, O., Lopez Sanchez, M., "A Dynamic and Adaptable Service Composition Architecture in the Cloud-Based on Multi-agent System", International Journal of Information Technology and Web Engineering (IJITWE), Vol.13. No.1, 2018, pp. 50-68. DOI : 10.4018/IJITWE.2018010104

[89] Merizig, A., Kazar, O., Saouli, H., Parisa, G., "Cloud ressources management based on agent for service discovery and composition", International Conference on Pattern Analysis and Intelligent Systems (PAIS'15) 2015, Tebessa–Algérie.

[90] Ming, C., Zhen-wu, W., "An Approach for Web Services Composition Based on QoS and Discrete Particle Swarm Optimization", Eighth ACIS International Conference on Software Engineering, IEEE, 2007.

[91] Mlungisi, d. (2008) "Agents, Agent architectures and Multi-agent systems", Master of science, Ehlers, E.M., Oosthuizen, O.L., Johnnesburg, 143 p.

[92] Nadanam, P., & Rajmohan, R. (2012). "QoS evaluation for web services in cloud computing". In Third International Conference on Computing Communication & Networking Technologies (ICCCNT), IEEE, July 2012 (pp. 1-8).

[93] Nikolay Mehandjiev, Freddy Lécué, Martin Carpenter, Fethi A. Rabhi, "Cooperative service composition", in : Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE 2012), Springer-Verlag, Berlin, Heidelberg, 2012, pp. 111–126.

[94] Nouredine, G., & Hassina, S. (2009, April). Composition of web service based multi-agent. In Multimedia Computing and Systems, 2009. ICMCS'09. International Conference on (pp. 51-55). IEEE.

[95] Piprani, B., Sheppard, D., & Barbir, A. (2013, September). Comparative analysis of SOA and cloud computing architectures using fact based modeling. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 524-533). Springer, Berlin, Heidelberg.

[96] Rajeswari, M., Sambasivam, G., Balaji, N., Basha, M. S., Vengattaraman, T., & Dhavachelvan, P. (2014). "Appraisal and analysis on various web service composition approaches based on QoS factors". Journal of King Saud University-Computer and Information Sciences, 26(1), 143-152.

[97] Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V., & Villano, U. (2013, December). "Security as a service using an SLA-based approach via SPECS". In Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on (Vol. 2, pp. 1-6). IEEE.

[98] Rankins, R., Bertucci, P., Gallelli, C., Silverstein, A. T., and Cotter, H., *Microsoft SQL Server 2012 Unleashed*. Pearson Education. 2012.

[99] Rodrigo, N.C., Rajiv, R., Anton, B., César, D.R., Rajkumar, B., Calheiros, N.R., Ranjan, R., Beloglazov, A., De-Rose, A.C.F. and Buyya, R. (2011) "CloudSim : A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", Software : Practice and Experience (SPE), Vol. 41, No. 1. pp.23-50.

[100] Rosenberg, J. Mateos, A., *The Cloud at Your Service The when, how, and why of enterprise cloud computing*, Manning Publications Co, 2011. Pp. 1.

[101] Rountree, D., Castrillo, I., *The Basics of Cloud Computing Understanding the Fundamentals of Cloud Computing in Theory and Practice*, Elsivier 2014. Pp.45.

[102] Sarna, D.Y.E. *Implementing and Developing Cloud Computing Applications*, Taylor & Francis Group. 2011., pp. 2.

[103] Safeer, Y., Mustafa, A. and Ali, A. N., "Clustering Unstructured Data (Flat Files) An Implementation in Text Mining Tool", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 8, No. 2, 2010

[104] Saurabh, K.G. and Rajkumar, B. (2011) "NetworkCloudSim : Modelling Parallel Applications in Cloud Simulations", Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing, Victoria, Canada, pp. 105-113.

[105] Saouli, H., Kazar, O., Benharkat, A. N., "SaaS-DCS : software-as-a-service discovery and composition system-based existence degree", International Journal of Communication Networks and Distributed Systems, Vol. 14, No. 4, 2015.

[106] Saouli. H., Kazar, O., Benhrket A. N., Merizig, A., "Cloud service selection and ranking algorithms based software and infrastructural characteristics", Conférence sur l'Ingénierie Informatique C2I. Alger. 2015.

[107] Sareni, B., & Krahenbuhl, L. (1998). "Fitness sharing and niching methods revisited". IEEE transactions on Evolutionary Computation, 2(3), 97-106.

[108] Shankar R. Ponnekanti, Armando Fox, "SWORD : a developer toolkit for Web service composition", in : Proceedings of the 11th International World Wide Web Conference (WWW'02), May 2002, pp. 83–107

[109] Sheila Mcilraith, "Adapting golog for composition of semantic Web services", in : Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR'02), Morgan Kaufmann, 2002, pp. 482–493

[110] Sheng, Q, Z., Qiao, X., Vasilakos, V. A., Szabo, C., Bourne, S., Xu, X., "Web services composition : A decade's overview", Information Sciences 280, 2014, 218-238.

[111] Sikos, L. (2015). *Mastering structured data on the Semantic Web : From HTML5 microdata to linked open data*. Apress.

[112] Sim, J., & Wright, C. (2000). *Research in health care : concepts, designs and methods*. Nelson Thornes.

[113] Sim, K. M. (2012). "Agent-based cloud computing". IEEE Transactions on Services Computing, 5(4), 564-577.

[114] Srividya Kona, Ajay Bansal, Luke Simon, Ajay Mallya, Gopal Gupta, Thomas D. Hite, "USDL : a service-semantics description language for automatic service discovery and composition", Int. J. Web Serv. Res. 6 (1) (2009) 20–48

[115] Srinivasan. S., *Cloud computing basics*. Springer. 2014. Pp. 44.

[116] Srinivasan, S. , *Security, Trust, and Regulatory Aspects of Cloud Computing in Business Environments*, IGI Global, 31 mars 2014

[117] Srujana, S., Raju, V. S., & Kiran, M. K. (2015). "Semantic Web services discovery using logic based method". In Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing : Theory and Applications (FICTA) 2014 (pp. 623-629). Springer, Cham.

[118] Sycara K.P. (1998) "Multi-agent system", AI Magazine, Vol. 19, No. 2, pp. 79–92.

[119] Swaroop Kalasapur, Mohan Kumar, Behrooz A. Shirazi, "Dynamic service composition in pervasive computing", IEEE Trans. Parallel Distr. Syst. 18 (7) (2007) 907–918

[120] Tang, K. S., Chan, T. M., Yin, R. J., & Man, K. F. (2012). *Multiobjective Optimization Methodology : A Jumping Gene Approach*. CRC Press.

[121] Tang, Y., & Li, C. (2013), *The Study on Livestock Production Prediction in Heilongjiang Province Based on Support Vector Machine*, Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), March 2013, Published by Atlantis Press, Paris, France.

[122] Tim, M. Subra, K. & Shahed, L. *Cloud Security and Privacy An Enterprise Perspective on Risks and Compliance*, O'Reilly. 1st. Ed. 2009.

[123] Tong, H., Cao, J., Zhang, S., & Li, M. (2011). "A distributed algorithm for web service composition based on service agent model". IEEE Transactions on Parallel and Distributed Systems, 22(12), 2008-2021.

[124] Truong, H. L., Dustar, S., "On Analyzing and Specifying Concerns for Data as a Service", in the proceedings of the 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), 2009.

[125] Oberle K, Fisher M (2010) "Etsi cloud—initial standardization requirements for cloud services". In : Altmann J, Rana O (eds) Economics of grids, clouds, systems, and services. Lecture notes in computer science, vol 6296. Springer, Berlin, pp 105–115. doi :10.1007/978-3-642-15681-6_8.

[126] Özel, S. A., "A Web page classification system based on a genetic algorithm using tagged-terms as features", Expert Systems with Applications, Vol. 38, pp. 3407–3415, 2011

[127] Piff, S. (2012). The age of cloud will be Hybrid, IDC. Whitepaper.

[128] Pham, T. V., Jamjoom, H., Jordan, K., & Shae, Z.-Y. (2010). "A service compositionframework for market-oriented high performance computing cloud". In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (pp. 284–287). Chicago, Illinois : ACM

[129] Qi Yu, Xumin Liu, Athman Bouguettaya, Brahim Medjahed, "Deploying and managing Web services : issues, solutions, and directions", VLDB J. 17 (3) (2008) 537–572.

[130] Qian, Z., Lu, S., & Xie, L. (2005, November). "Mobile-agent-based web service composition". In International Conference on Grid and Cooperative Computing GCC (pp. 35-46).

[131] Quan Z. Sheng, Boualem Benatallah, Marlon Dumas, Eileen Mak, "SELF-SERV : a platform for rapid composition of Web services in a peer-to-peer environment", in : Proceedings of the 28th International Conference on Very Large Databases (VLDB'02), Morgan Kaufmann, 2002, pp. 1051–1054.

[132] Van Ommeren, E., Duivestein, S., deVadoss, J., Reijnen, C., Gunvaldson, E., *Collaboration in the Cloud How Cross-Boundary Collaboration Is Transforming Business*, Microsoft and Sogeti, 2009., pp.9.

[133] Vu, Q. H., Pham, T. V., Truong, H. L., Dustdar, S., and Asal, R., "Demods : A description model for data-as-a-service", In 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, pp. 605 – 612.

[134] Wang, L., Ranjan, R., Chen, J., Benalallah, B. *Cloud Computing methodology, systems, and Applications*. Taylor & Francis Group. 2012. p.3.

[135] Weinhardt C, Anandasivam A, Blau B, Borissov N, Meinl T, Michalk W, St??er J (2009), "Cloud computing—a classification business models, and research directions". Bus Inf Syst Eng 1(5) :391–399

[136] Wickremasinghe, B., Calheiros, N.R., and Buyya, R. (2010) "CloudAnalyst : A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), Perth, Australia, pp. 446-452.

[137] Wilkinson, B. (2009). *Grid computing : techniques and applications*. CRC Press.

[138] Wischik, D., Handley, M., & Braun, M. B. (2008). "The resource pooling principle". ACM SIGCOMM Computer Communication Review, 38, 47–52.

[139] Worm, D.,?ivkovi?, M., van den Berg, H., & van der Mei, R. (2012, December). "Revenue maximization with quality assurance for composite web services". In Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference on (pp. 1-9). IEEE.

[140] Wu L, Yang C. "A solution of manufacturing resources sharing in cloud computing environment". International Conference on Cooperative Design, Visualization and Engineering. In : Luo Y, editor. 6240 LNCS. Berlin Heidelberg : Springer-Verlag ; 2010. p. 247–52

[141] Xiaona, W., Bixin, L., Rui, S., Cuicui, L., & Shanshan, Q. (2012). "Trust-Based Service Composition and Optimization". In Software Engineering Conference (APSEC), 2012 19th Asia-PacificVol. 1 (pp. 67–72)

[142] Xu, J., Reiff-Marganiec, S., "Towards Heuristic Web Services Composition Using Immune Algorithm", IEEE, International Conference on Web Services ICWS'08, pp. 238 – 245, November 2008.

[143] Xu, X., "From cloud computing to cloud manufacturing", Robotics and computer-integrated manufacturing, vol. 28, No. 1, pp. 75 – 86. 2012.

[144] Xue, F., Sanderson, A. C., & Graves, R. J. (2003, December). "Pareto-based multi-objective differential evolution". In Evolutionary Computation, 2003. CEC'03. The 2003 Congress on (Vol. 2, pp. 862-869). IEEE.

[145] Yang, Z., Shang, C., Liu, Q., Zhao, C., "A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm", Journal of Computational Information Systems, pp. 2617-2622, 2010.

[146] Ye, Z., Zhou, X., & Bouguettaya, A. (2011). "Genetic algorithm based QoS-aware service compositions in cloud computing". In Database systems for advanced applications (pp. 321-334). Springer Berlin/Heidelberg.

[147] Yu, T., Lin, K.J., "Service Selection Algorithms for Web Services with End-to-end QoS Constraints", Journal of Information Systems and E-Business Management, Vol. 3, Number 2, Springer 2005.

[148] Yu, T., Lin, K.J., "Service Selection Algorithms for Web Services with End-to-end QoS Constraints", e-Commerce Technology CEC 2004, Proceedings. IEEE International, pp. 129 – 136, August 2004.

[149] Yu, H.Q., and Reiff-Marganiec, S., "A Backwards Composition Context Based Service Selection Approach for Service Composition", ServicesComputing SCC '09, IEEE International, pp. 419– 426, October 2009

[150] Yu, Q., & Bouguettaya, A. (2013). *Efficient service skyline computation for composite service selection*. IEEE Transactions on Knowledge and Data Engineering, 25(4), 776-789.

[151] Zehoo, E., and Hong, Y. W., *Pro ODP. NET for Oracle Database 11g*. Apress, 2010, p. 20.

[152] Zeng, C., Guo, X., Ou, W., & Han, D. (2009). "Cloud computing service composition and search based on semantic". IEEE International Conference on Cloud Computing, pp. 290-300.

[153] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q.Z., "Quality Driven Web Service Composition", Proceeding of 12th International World Wide Web Conference (WWW), 2003.

[154] Zhang, M., Ranjan, R., Nepal, S., Menzel, M., & Haller, A. (2012). "A declarative recommender system for cloud infrastructure services selection". In Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services (pp. 102–113). Berlin, Germany : Springer-Verlag

[155] Zhao, X., Wen, Z., & Li, X. (2014). "QoS-aware web service selection with negative selection algorithm". Knowledge and Information Systems, 40(2), 349-373.

[156] Zheng, K., Zhang, L., Xiang, W., Wang, W., (2016), "Heterogeneous Vehicular Networks", SpringerBriefs in Electronical and Computer Engineering, DOI 10.1007/978-3-319-25622-1_1.

[157] Zaigham, M., Hill, R., *Cloud computing for enterprise architectures*. Springer, 2011. pp. 85.

Annexe A

Liste des publications

A.1 Revues Internationales

- Abdelhak Merizig, Okba Kazar, Maite Lopez Sanchez, "A Dynamic and Adaptable Service Composition Architecture in the Cloud Based on Multi-agent System". International journal of information technology and web engineering, Vol.13. No.1, 2018. (**Disponible en-ligne**).
- Abdelhak Merizig, Okba Kazar, Maite Lopez Sanchez, "Multi-agents System Approach for Service Deployment in the Cloud", Int. J. Communication Networks and Distributed Systems, Vol. X, No. Y, pp.xxx–xxx. (**En cours d'évaluation**)

A.2 Conférences Internationales

- Abdelhak Merizig, Okba Kazar, "Information system for monitoring and forecasting the production of dates", 2ème conférence francophone sur les Systèmes Collaboratifs (SysCo'14), 27-29 Septembre 2014, Hammamet-Tunisie, pp. 173-184.
- Abdelhak Merizig, Okba Kazar, Hamza Saouli, Parisa Ghoudous, "Cloud ressources management based on agent for service discovery and composition", International Conference on Pattern Analysis and Intelligent Systems (PAIS'15), 25-26 Octobre 2015 Tebessa- Algérie, pp. 329-334.
- Hamza Saouli, Abderaouf Ghamri, Abdelhak Merizig, Okba Kazar, "A new cloud computing approach based SVM for relevant data extraction", The 2nd International Conference on Big Data, Cloud And Applications (BDCA'17), 29-30 Mars 2017, Tetouan-Maroc, pp.1-7.

– Abdelhak Merizig, Hamza Saouli, Maite Lopez Sanchez, Okba Kazar, Aicha-Nabila Benharkat, "An Extended Data as a Service Description Model for Ensuring Cloud Platform Portability", The 2nd International Conference on Advanced Systems and Electrical Technologies (IC_ASET'2018), Hammamet-Tunisie, pp.xx-xx. (Acceptée)

A.3 Conférences Nationales

– Hamza Saouli, Okba Kazar, Aicha-Nabila Benharkat, Abdelhak Merizig, "Cloud Service Selection and Ranking Algorithms based software and infrastructural characteristics", 1ère Conférence sur l'Ingénierie Informatique (C2I'14), 16-17 Décembre 2014 in Alger- Algérie, pp. xx-xx.

– Hamza Saouli, Okba Kazar, Aicha-Nabila Benharkat, Abdelhak Merizig, "Architecture Cloud computing basée agent mobile pour la découverte de services web", 8ème édition du Seminaire National en Informatique Biskra (SNIB'15), 20-22 Janvier 2015 Biskra-Algérie, pp. xx-xx.

– Abdelhak Merizig, Okba Kazar, Saber Benharzallah, "Web services composition in cloud computing based on agents", International Workshop on Artificial Intelligence and Information Communication Technologies (IWAIICT'15), 23-24 Novembre 2015, Biskra- Algérie