

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Blida -1-



Faculté des Sciences

Département d'Informatique

Mémoire de fin d'étude présenté en vue d'obtention du diplôme

MASTER EN INFORMATIQUE

Option : Systèmes Informatiques & Réseaux

Thème

**Composition de services Cloud de type SaaS
à base de sélection globale**

Présenté par :

M. AOUN SEGHIR Karim

M. MEHRAZ Abdelkader

Soutenus le : 29/11/2020, devant le jury :

Mme Boutoumi Bachira (Présidente)

M. Kameche Abdallah Hicham (Examineur)

Mme Mancer Yasmine (Promotrice)

Promotion : 2019/2020

Remerciement

*Avant tout nous formulons notre gratitude à Allah
le tout puissant de nous avoir donné la force
d'achever ce travail,*

*Nous tenons aussi à remercier : Mme Mancery,
notre promotrice, pour ses conseils,
ses orientations, et sa disponibilité,
qui nous ont permis de réaliser ce travail dans
les meilleures conditions,*

*Ainsi les membres du jury, pour avoir fait
l'insigne honneur d'accepter de lire et
juger ce travail,*

*Un très grand merci à tous les enseignants de notre
département, qui ont assuré notre enseignement pendant
tout notre parcours académique.*

*Sans oublier de remercier nos parents
pour leur contribution, leur soutien et leur patience,
nos proches,
nos amis, nos collègues
et toutes personnes qui nous ont aidées
par leur soutien permanent
de près ou de loin de nos études.*

Résumé

Les services Cloud sont d'une grande importance à l'heure actuelle, et l'importance de ces services est apparue lors de la crise qui a frappé le monde, représentée par la pandémie Covid 19, où l'utilisation des services Cloud a fortement augmenté en raison des mesures de quarantaine prises dans la plupart des pays du monde. Compte tenu de cela, les demandes des clients sont devenues plus complexes, invoquant plusieurs services à la fois. Ces derniers nécessitent une composition de services pour répondre aux exigences du client. Avec la disponibilité d'un grand nombre de services Cloud de type Software as a Service (SaaS), beaucoup d'entre eux offrent le même service, mais avec des valeurs de qualité de service différentes.

Dans ce mémoire, nous avons tenté de trouver une solution au problème de composition des services Cloud de type SaaS, qui dépend de la méthode de sélection globale, et prend également en compte les valeurs de qualité du service composé exigées par le client.

Afin de mettre en œuvre la solution proposée, cette solution a été modélisée conformément à ce qui a été mentionné dans l'étude conceptuelle, et en utilisant le langage Java EE, une application pour ce modèle a été développée, et des expériences et des simulations ont prouvé l'efficacité de cette solution, par rapport à d'autres solutions.

Mots clés : Cloud Computing, composition, sélection, Software as a Service (SaaS), qualité de service (QoS).

Abstract

Cloud services are of great importance today, and the importance of these services emerged during the crisis that hit the world, represented by the Covid 19 pandemic, where the use of Cloud services increased sharply due to the quarantine measures taken in most countries of the world. As a result, customer demands have become more complex, so that it invokes several services at once, requiring a mix of services to meet customer requirements. With the availability of a large number of cloud services, many of them offer the same service, but with different qualities, so we see that this customer is interested in the quality of service when requesting cloud services.

In this thesis, we tried to find a solution to the problem of cloud services composition, which depends on the global selection method, and also takes into account the quality of service of the composite service required by the customer, in order to select the set of services needed to create the required composite service with the quality of service that satisfies this customer.

In order to implement the proposed solution, this solution was modeled according to what was mentioned in the conceptual study, and using the Java EE language, an application for this model was developed. Experiments and simulations have proven the effectiveness of this solution, compared to other solutions, and the extent of its ability to meet customer demands.

Key words: Cloud Computing, composition, selection, Software as a Service (SaaS), Quality of Service (QoS).

ملخص

تعتبر الخدمات السحابية ذات أهمية كبيرة في وقتنا الحالي، وظهرت أهمية هذه الخدمات خلال الأزمة التي أصابت العالم والمتمثلة في جائحة كوفيد 19، أين تزايد بشكل كبير استعمال الخدمات السحابية نظرا لإجراءات الحجر الصحي المتخذة في جل دول العالم. ونظرا لهذا فإن طلبات الزبائن أصبحت أكثر تعقيدا، بحيث أصبحت تتضمن خدمات مركبة تتطلب تدخل العديد من الخدمات في كل مرة لتلبية رغبة الزبون. ومع توفر عدد هائل من الخدمات السحابية، فإن العديد منها تقدم نفس الخدمة لكن بنوعيات مختلفة، وبالتالي نجد أن هذا الزبون أصبح يهتم بنوعية الخدمة عند طلب الخدمات السحابية.

في هذه المذكرة، حاولنا إيجاد حل لمشكلة تركيب الخدمات السحابية، التي تعتمد على طريقة الاختيار الشامل، كما تأخذ بعين الاعتبار نوعية الخدمة المركبة المطلوبة من طرف الزبون، من أجل اختيار مجموعة الخدمات اللازمة لتشغيل الخدمة المركبة المطلوبة مع نوعية الخدمة التي ترضي هذا الزبون.

ولتطبيق الحل المقترح، تمت نمذجة هذا الحل فيما يتماشى مع ما جاء في جزء المفاهيم، وباستعمال لغة جافا بنسختها الموجهة للمؤسسات (java EE)، تم تطوير تطبيق لهذا النموذج، وقد اثبتت التجارب والمحاكاة مدى نجاعة وفعالية الحل المقترح بالمقارنة مع بعض الحلول الأخرى، ومدى قدرته على ارضاء طلبات الزبائن.

الكلمات الدلالية: الحوسبة السحابية، تركيب، اختيار، البرمجيات كخدمة، نوعية الخدمة.

Table des matières

Introduction générale.....	1
<i>Chapitre I : Généralités sur le Cloud Computing.....</i>	4
1. Introduction.....	5
2. Définition du Cloud Computing	5
3. Éléments constitutifs du Cloud Computing.....	6
4. La virtualisation	9
4.1 Rôle de la virtualisation dans le Cloud.....	10
5. Caractéristiques du Cloud Computing	11
6. Types de services dans les environnements de Cloud Computing	12
6.1 Software as a Service (SaaS).....	14
6.2 Plateform as a Service (PaaS).....	14
6.3 Infrastructure as a Service (IaaS).....	14
6.4 X as a Service (XaaS)	14
7. Modèles de déploiement du Cloud Computing	15
7.1 Cloud public.....	15
7.2 Cloud privé.....	15
7.3 Cloud hybride	16
7.4 Cloud communautaire.....	16
8. Architecture de référence du Cloud Computing.....	16
9. Acteurs du Cloud Computing	17
9.1 Consommateur Cloud (Cloud consumer)	18
9.2 Fournisseur Cloud (Cloud provider).....	18
9.3 Courtier en Cloud (Cloud broker).....	19

9.4	Relation entre les acteurs du Cloud computing.....	19
10.	Avantages et inconvénients du Cloud Computing	20
11.	Évolution du Cloud Computing.....	20
11.1	Inter-Cloud.....	21
11.1.1	Fédération de Cloud.....	21
11.1.2	Multi-Cloud	21
11.2	Topologies des différentes architectures de l'Inter-Cloud.....	22
11.3	Scénarios de l'Inter-Cloud.....	23
11.4	Défis de l'Inter-Cloud.....	24
12.	Conclusion.....	26
<i>Chapitre II : La composition des services dans le Cloud.....</i>		<i>27</i>
1.	Introduction	28
2.	Composition des services dans le Cloud Computing	28
2.1	Définition de la composition	28
2.2	Défis de composition des services.....	29
3.	Les modes d'exécution de la composition	30
4.	Étapes de composition de services	31
4.1	Conception et déploiement de la composition des services	31
4.2	Exécution et suivi de la composition des services.....	32
4.3	Retour d'information sur l'évaluation de la composition des services..	32
5.	Les services Cloud de type SaaS	33
5.1	Les technologies des services Web	33
5.1.1	Description d'un service Web	33
5.1.2	Langages de description	34
6.	Types de composition.....	35
7.	Implémentation de la composition des services.....	36
8.	Les exigences de la composition des services	37
8.1	Exigences fonctionnelles et non fonctionnelles	37
8.2	Critères de qualité de service	37

8.3 Fonctions d'agrégations des valeurs de QoS	38
9. Méthodes de sélection	39
10. La composition des services dans un environnement multi-Cloud	40
11. Travaux connexes.....	42
12. Étude comparative	48
12.1 Les critères de comparaison des solutions	48
12.1.1 Comparaison selon l'environnement Cloud	48
12.1.2 Comparaison selon le type de sélection	51
12.1.3 Utilisation des contraintes de QoS	51
12.1.4 Comparaison selon le type d'algorithme de solution	54
12.2 Comparaison des différents travaux	57
13. Conclusion.....	59
<i>Chapitre III : Modélisation de l'approche de composition</i>	60
1. Introduction	61
2. Spécification des besoins.....	63
3. Architecture proposée pour le système de composition.....	64
4. Scenarios de composition.....	66
5. Définition des critères de QoS	67
6. Correspondance entre la requête et les services disponibles.....	69
7. La sélection globale	69
7.1 Service Level Agreement	69
7.2 Mesure de performance du service composé	70
7.2.1 La fonction de fitness	71
8. L'algorithme de sélection	71
8.1 Encodage du problème	71
8.2 Design de l'algorithme	72
9. L'élaboration de la composition	76
10. Conclusion.....	77

<i>Chapitre IV : Expérimentation</i>	78
1. Introduction	79
2. Outils et environnement de développement	79
2.1 IDE Eclipse	79
2.2 Java Development Kit JDK	79
2.3 Apache Tomcat	80
2.4 Docker	80
3. Présentation de l'interface de l'application	81
4. Simulation et expérimentations	86
5. Résultats et discussions	93
6. Conclusion	97
Conclusion générale	98
Bibliographie	99

Liste des Figures

Figure 1: Le Cloud Computing.	6
Figure 2: Types de services dans le Cloud computing [6].	13
Figure 3: Architecture de référence du Cloud Computing [11].	18
Figure 4: Topologie des différentes architectures inter-Cloud[17].	23
Figure 5: L'architecture du courtier et les scénarios de communication [14].	24
Figure 6: Les modes de composition de services[19].	30
Figure 7: La composition des services dans un environnement multi-Cloud [15]. ...	41
Figure 8: Pseudo algorithme COM2 [31].	43
Figure 9: Architecture du système multi-agent [20].	45
Figure 10: L'algorithme (PSO+ABC) [10].	46
Figure 11: Traitement de la composition selon le type d'environnement.	50
Figure 12: Utilisation des contraintes de QoS dans la sélection locale.	53
Figure 13: Utilisation des contraintes de QoS par rapport à l'environnement Cloud.	54
Figure 14: Statistique sur le type d'algorithme utilisé dans la composition.	57
Figure 15: Le cycle de l'Extreme Programming[67].	62
Figure 16: Diagramme de cas d'utilisation.	64
Figure 17: L'architecture globale du système de composition.	66
Figure 18: Types de critères de QoS utilisés.	68
Figure 19: Encodage du problème.	72
Figure 20: Pseudo algorithme proposé.	73
Figure 21: Opération de croisement.	75
Figure 22: Organigramme de notre solution.	76
Figure 23: Interface inscription utilisateur.	81
Figure 24: Interface de connexion d'un utilisateur.	82
Figure 25: Interface fournisseur pour publication de service.	83
Figure 26: Interface de gestion des services publiés.	84
Figure 27: Page d'accueil utilisateur.	84
Figure 28: Interface de sélection d'un service.	85
Figure 29: Interface pour la spécification des valeurs de QoS.	85
Figure 30: Information sur le service sélectionné.	86

Figure 31: Configuration de la connexion eclipse docker tooling.	89
Figure 32: Configuration Dockerfile pour la création de l'image du service.	90
Figure 33: Commande 'docker image ls'	90
Figure 34: Commande 'docker container ls'	91
Figure 35: Code de la servlet qui exécute service compose.....	92
Figure 36: Interface pour la saisie des notes des modules.	92
Figure 37: Résultat après l'exécution du service composé	93
Figure 38: Valeurs de fitness par rapport au nombre des services candidats.....	94
Figure 39: Valeurs de fitness par rapport au nombre de tâches.	94
Figure 40: Temps d'exécution par rapport au nombre de services.....	95

Liste des tableaux

Tableau 1: Fonction d'agrégation pour chaque attribut de QoS [27].....	39
Tableau 2: Comparaison entre quelque approches de composition[38].	47
Tableau 3: Les approches de composition selon l'environnement Cloud.....	50
Tableau 4:Utilisation des contraintes de QoS selon l'environnement Cloud.	52
Tableau 5: Traitement de la composition selon le type d'algorithme.	56
Tableau 6: Comparaison entre les différentes approches.....	58
Tableau 7: Premier scénario de simulation.	87
Tableau 8: Deuxième scénario de simulation.	87
Tableau 9: Récapitulatif de l'approche proposée.	96

Liste des acronymes et abréviations

ABC	:	Artificial Bee Colony
API	:	Application Programming Interface
CA	:	Combinatorial Algorithms
FLA	:	Federation-Level Agreement
GA	:	Genetic Algorithm
IaaS	:	Infrastructure as a Service
NaaS	:	Network as a Service
NIST	:	National Institute of Standards and Technology
OWL-S	:	Web Ontology Language for Web Services
PaaS	:	Platform as a Service
PME	:	Petites et Moyennes Entreprises
PSO	:	Particle Swarm Optimization
QoS	:	Quality of Service
SaaS	:	Software as a Service
SLA	:	Service Level Agreement
SOA	:	Service Oriented Architecture
SOAP	:	Simple Object Access Protocol
UDDI	:	Universal Description, Discovery and Integration
WAR	:	Web Application Archive
WSDL	:	Web Service Description Language
XaaS	:	Anything as a Service
XML	:	eXtensible Markup Language

Introduction générale

Les solutions de Cloud Computing existantes n'ont pas été construites en tenant compte de l'interopérabilité. Elles verrouillent généralement les clients dans un service Cloud unique empêchant la portabilité des données ou des logiciels créés. En outre, la bataille pour la domination entre les grands fournisseurs, comme Amazon, Google et Salesforce, les rend réticents à se mettre d'accord sur des normes largement acceptées pour promouvoir leurs propres formats incompatibles.

La composition est l'élément manquant qui remédiera à cette situation et bénéficiera à la fois aux clients et aux fournisseurs Cloud. En particulier, dans un environnement Cloud interopérable, les clients pourront comparer et choisir des offres parmi les différentes offres Cloud de différentes caractéristiques, en négociant avec des fournisseurs Cloud à chaque fois que cela s'avérerait nécessaire, sans mettre en danger les données et les applications et pourront ainsi composer de nouveaux services Cloud à partir de services existants dans le but de satisfaire des demandes plus complexes.

Problématique

Dans l'environnement du Cloud Computing, la composition de systèmes Software as a Service (SaaS) joue un rôle très important dans la satisfaction des exigences de l'utilisateur. En général, la composition fait référence à la capacité de communication entre différents services Cloud à travers l'échange des résultats pour atteindre un objectif complexe.

La problématique de ce travail consiste à définir une solution pour la composition de services Cloud de type Software as a Service (SaaS), en se basant sur les critères de qualité de services (QoS) globaux à travers une sélection globale.

Objectifs

Les objectifs visés à travers ce travail sont les suivants :

1. Étude de la composition dans le Cloud Computing et dans les services SaaS.

2. Étude comparative des solutions existantes pour la composition au niveau SaaS.
3. Définition des exigences de composition dans les environnements Cloud Computing de type SaaS.
4. Définir une solution pour la composition de services Cloud de type SaaS. La solution proposée doit garantir les objectifs suivants :
 - a. Définition des scénarios de composition.
 - b. Définition des critères de qualité de services à utiliser.
 - c. Élaboration des correspondances entre la requête de l'utilisateur et les services disponibles.
 - d. Élaboration de la sélection globale.
 - e. Élaboration de la composition des services SaaS.
5. Validation de la solution proposée par une expérimentation à travers une application Java EE.

Organisation du mémoire

Ce document est composé d'une introduction générale, de quatre chapitres et d'une conclusion générale.

Ce travail va être débuté par une introduction générale, dans laquelle nous allons expliquer le concept général de ce mémoire, spécifier la problématique et fixer les différents objectifs.

Le premier chapitre est un chapitre de généralités, il représente une vue globale sur le domaine du Cloud Computing, il comporte des définitions pour les différents concepts, et il cite les éléments constitutifs du Cloud, les caractéristiques, les modèles, les avantages et inconvénients, ainsi que l'évolution du Cloud Computing.

Le deuxième chapitre est consacré à l'étude de la composition des services dans le Cloud de type SaaS, il se divise en deux parties. La première partie décrit les différents concepts, elle montre aussi les modes d'exécution, les défis, les étapes, les méthodes et les exigences de la composition. Elle donne aussi une vue sur la composition dans un environnement multi-Cloud. La deuxième partie est consacrée à

Introduction Générale

l'analyse des travaux connexes, dans le but d'extraire les différents critères d'évaluation des solutions, et se termine par une étude comparative de ces travaux.

Le troisième chapitre, ici nous allons définir l'architecture et la modélisation de notre système, avec la spécification des différents besoins du système. Nous allons spécifier et détailler les processus nécessaires de la composition, sous forme de diagramme et un pseudo algorithme, depuis l'arrivée de la requête utilisateur jusqu'à l'exécution de la composition.

Le quatrième chapitre c'est le dernier chapitre, dédié aux expérimentations, il commence par une présentation des outils et l'environnement de développement, puis une description des interfaces de l'application. Aussi, il cite les différentes étapes de simulation et les scénarios d'expérimentation, lors de la mesure de la performance de l'approche proposée, et se termine par une discussion des résultats obtenus.

La fin de ce mémoire, une conclusion générale est présentée comme une récapitulation de ce travail, et pour conclure le mémoire, un ensemble de perspectives sont proposées.

Chapitre I : Généralités sur le Cloud Computing

1. Introduction

Avec l'apparition et le développement de l'internet, l'homme a pu rendre les quatre coins du monde interconnectés, et avec son ambition constante de s'étendre et d'être présent partout, une nécessité est apparue, c'est l'accessibilité et la disponibilité de ses informations, depuis n'importe où, et à tout moment, cela pour les entreprises comme pour les particuliers.

D'un autre côté, le développement technologique a mené à l'amélioration des équipements informatiques, en termes de capacité de stockage, vitesse de calcul et débit de transfert. Et en essayons de bénéficier de ces facteurs, pour répondre aux exigences pour que l'information soit accessible et disponible, cela a donné naissance au Cloud Computing, qui depuis son apparition n'a pas cessé de se développer.

Dans ce chapitre, nous allons présenter la technologie du Cloud Computing, en spécifiant une définition, les types et les caractéristiques, des éléments constitutifs de l'environnement de Cloud computing, et voir à la fin l'évolution de cette technologie.

2. Définition du Cloud Computing

La recherche d'une définition pour le cloud computing, nous mène à trouver plusieurs. Et parmi ces définitions, celle de Microsoft [1] : «Le cloud computing est la fourniture de services informatiques (notamment des serveurs, du stockage, des bases de données, la gestion réseau, des logiciels, des outils d'analyse, l'intelligence artificielle) via Internet, dans le but d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle. En règle générale, vous payez uniquement les services cloud que vous utilisez (réduisant ainsi vos coûts d'exploitation), gérez votre infrastructure plus efficacement et adaptez l'échelle des services en fonction des besoins de votre entreprise ».

Nous pouvons aussi trouver une autre définition, celle de NIST (National Institute of Standards and Technology) [2] : « Le cloud computing est un modèle qui permet un accès omniprésent, pratique et à la demande à un réseau partagé et à un

ensemble de ressources informatiques configurables (comme par exemple : des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être provisionnées et libérées avec un minimum d'administration. Ce modèle est composé de cinq caractéristiques essentielles, de trois modèles de services et de quatre modèles de déploiement ». La figure suivante donne une vue générale sur le Cloud Computing.

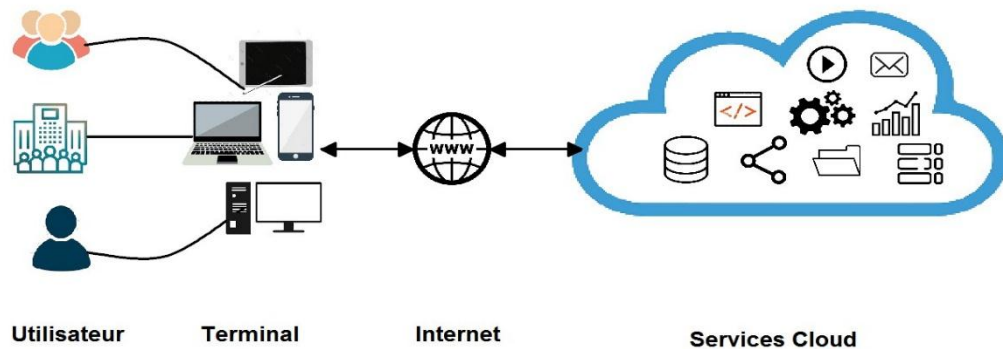


Figure 1: Le Cloud Computing.

D'après ces définitions, nous pouvons constater que le cloud est une infrastructure dans laquelle des ressources matérielles et systèmes, gérées par des serveurs et accessibles via un réseau, sont mises à la disposition de ceux qui ont besoin d'eux, ou autrement, le cloud computing c'est de pouvoir utiliser des ressources informatiques sans les posséder.

3. Éléments constitutifs du Cloud Computing

D'après Vic (J.R.) Winkler¹ les éléments technologiques qui constituent les bases de l'infrastructure informatique du Cloud peuvent être rangés dans les grandes catégories suivantes [3] :

¹ Vic (J.R.) Winkler : expert en cyber sécurité. Il a plus de 30 ans d'expérience dans les domaines de la cyber sécurité et la sécurité de cloud. Il était responsable technique de la sécurité-Cloud chez Sun-Microsystems, actuellement consultant technique chez le gouvernement des États-Unis.

➤ **L'infrastructure**

L'infrastructure informatique du Cloud est un assemblage de serveurs, d'espaces de stockage et de composants réseau organisés de manière à permettre une croissance incrémentale supérieure à celle que l'on obtient avec les infrastructures classiques. Ces composants doivent être sélectionnés pour leur capacité à répondre aux exigences d'extensibilité, d'efficacité, de robustesse et de sécurité.

Les serveurs d'entreprise classiques ne disposent pas des capacités réseau, de la fiabilité ni des autres qualités nécessaires pour satisfaire efficacement et de manière sécurisée les accords de niveau de service (SLA, service level agreement). Par ailleurs, les serveurs d'un Cloud affichent des coûts de fonctionnement moins élevés et ils peuvent être plus fiables s'ils ne sont pas tous équipés de disques internes.

➤ **Le réseaux IP**

Dans une infrastructure de Cloud, le réseau non seulement connecte les utilisateurs au Cloud, mais sert également à l'interconnexion interne du Cloud. Le modèle mis en œuvre dans un réseau d'entreprise ne répond pas aux besoins d'efficacité et de sécurité associés à l'acquisition et au fonctionnement du Cloud. À l'échelle du Cloud, le réseau doit s'orienter vers un système Carrier-Grade, avec des stratégies réseau optimisées.

Les multiples commutateurs disséminés tout au long des chemins de données deviennent des points uniques de défaillance (SPOF, single points of failure) et ajoutent des coûts de différentes manières. Bien que l'optimisation puisse conduire à un seul réseau unifié, la sécurité nécessite un partitionnement ou une virtualisation du réseau pour une séparation réelle entre les différentes classes de trafic.

Le réseau sera probablement plus plat, mais vous devez vous attendre à plusieurs réseaux en parallèle pour une meilleure sécurité. Certains isolent la gestion de la plateforme des données publiques et du trafic de service, tandis que d'autres peuvent être nécessaires pour autoriser les évolutions. Ces réseaux

supplémentaires induisent de nouveaux coûts, mais vous obtenez alors une séparation physique et une meilleure sécurité.

➤ **La virtualisation**

Avec des racines profondément ancrées dans l'informatique, la virtualisation sert à partitionner un seul serveur physique en plusieurs machines virtuelles ou une seule ressource physique, comme un espace de stockage ou un réseau, en plusieurs ressources virtuelles. Elle permet une consolidation de serveurs avec une grande souplesse d'utilisation.

Dans le contexte du cloud computing, la virtualisation est importante pour la mise en service et le retrait rapide de serveurs. Un logiciel de virtualisation du Cloud présente également une perspective dynamique et une vue unifiée de l'utilisation et de l'efficacité des ressources, cela afin d'assurer le fonctionnement des services du Cloud. La virtualisation est la principale technologie permettant d'arriver à une utilisation rentable des serveurs tout en prenant en charge la séparation entre de multiples locataires d'un matériel physique. Il existe d'autres solutions pour arriver à ces objectifs, mais ses avantages en font l'approche de choix.

➤ **Le logiciel**

Il autorise la mise en œuvre de tous les aspects de la gestion, de la mise à disposition, du développement des services, de la comptabilité et de la sécurité de l'infrastructure du Cloud. Il est indispensable que l'infrastructure du Cloud soit capable d'appliquer dynamiquement des politiques de séparation, d'isolation, de surveillance et de constitution d'un service. Le choix d'une configuration standard pour l'infrastructure permet au logiciel d'automatiser les tâches sous-jacentes à l'élasticité et au changement de forme de manière à présenter des services constitués de serveurs, de machines virtuelles, d'espaces de stockage, de services et d'autres composants informatiques. Grâce au logiciel, nous pouvons automatiser la mise en service et le retrait.

➤ L'interfaces de service

L'interface de service placée entre le fournisseur et le client est un élément de différenciation du Cloud. Elle représente un contrat qui fait respecter la proposition de valeur décrite par des SLA et des conditions tarifaires. Si le Cloud semble nouveau, c'est principalement en raison de cette interface. Elle représente la valeur d'un fournisseur et sert de base à la concurrence. Par l'ajout d'interfaces de libre-service, nous obtenons d'autres optimisations. Les clients du Cloud sont en mesure d'engager des ressources de manière automatisée sans que le service informatique soit un obstacle.

L'espace de stockage et les ressources sont présentés au travers d'une interface graphique que l'utilisateur peut manipuler de manière à obtenir et à instancier une infrastructure informatique virtuelle. Un navigateur web et une carte bancaire, voilà tout ce qu'il vous faut pour construire votre propre Datacenter virtuel.

4. La virtualisation

La virtualisation, dans les mots les plus simples et les plus faciles, est l'abstraction des ressources informatiques. La virtualisation est une composante du Cloud qui permet de séparer le système d'exploitation du matériel sur lequel il fonctionne. Aussi, elle a la capacité de permettre à une ressource physique unique de servir de multiples ressources virtuelles et peut même faire fonctionner plusieurs ressources physiques comme une seule ressource virtuelle.

La virtualisation a apporté un grand changement dans le fonctionnement des organisations informatiques à tous égards. Ce qui suit explique clairement ce changement [4].

Avant la virtualisation :

- Une image du système d'exploitation sur une machine.
- Logiciel et matériel étroitement couplés.
- Tenter d'exécuter plus d'une application sur la même machine crée souvent des conflits.
- Une infrastructure moins flexible et même coûteuse.

Après la virtualisation :

- Indépendance du système d'exploitation et des applications du matériel sous-jacent.
- Les machines virtuelles peuvent être facilement fournies à tout système.
- Le système d'exploitation et l'application peuvent être facilement gérés comme une entité complète en les regroupant dans des machines virtuelles.

4.1 Rôle de la virtualisation dans le Cloud

La virtualisation et les systèmes d'exploitation multitâches ont des capacités de travail similaires. Il permet de centraliser un certain nombre de serveurs virtuels dans une seule machine physique. Si une organisation a besoin d'exécuter deux ou plusieurs serveurs pour une tâche particulière, au cas où l'un d'eux tomberait en panne, même si aucun d'eux n'est proche de l'utilisation complète des ressources.

La virtualisation peut être utile dans ce cas, car il est relativement facile de migrer d'un ordinateur physique à un autre. Un autre grand avantage est donc la migration. La virtualisation peut aider à économiser l'énergie, à réduire les coûts et l'encombrement du matériel serveur et donc aider les entreprises à maximiser leurs profits [4].

Dans la virtualisation de calcul, une couche de virtualisation réside entre le matériel et la machine virtuelle (sur laquelle tourne un système d'exploitation). La couche de virtualisation est également appelée hyperviseur. L'hyperviseur fournit des ressources matérielles standardisées (par exemple : CPU, mémoire, réseau, etc.) aux machines virtuelles. Les hyperviseurs sont classés en deux catégories : l'hyperviseur hébergé et l'hyperviseur bare-metal [4].

1. Hyperviseur Bare-metal : dans ce type, l'hyperviseur est directement installé sur le matériel. Après avoir installé l'hyperviseur, l'instance souhaitable du système d'exploitation est installée sur cet hyperviseur. Par conséquent, il est plus efficace qu'un hyperviseur hébergé. Les ordinateurs de bureau utilisent ce type d'hyperviseur.

- 2. Hyperviseur hébergé :** dans ce type, un système d'exploitation est installé sur le matériel, puis l'hyperviseur est installé sur ce système d'exploitation. Sur cet hyperviseur, une instance du système d'exploitation est installée. Puisqu'il fonctionne sur un système d'exploitation, il prend en charge la vaste gamme de configurations matérielles.

La virtualisation est l'une des techniques importantes d'économie d'énergie et de réduction du matériel utilisées par les fournisseurs de cloud computing. Avec la virtualisation du système d'exploitation, chaque machine virtuelle peut utiliser un système d'exploitation différent, et chaque système d'exploitation est isolé des autres [4].

5. Caractéristiques du Cloud Computing

Pour qu'un service soit considéré comme étant du cloud computing, il devrait satisfaire certains critères. Le NIST spécifie cinq caractéristiques [5] qui sont considérées comme essentielles pour le Cloud computing :

- 1. Libre-service à la demande :** Un consommateur peut bénéficier unilatéralement des capacités informatiques, telles que le temps du serveur et le stockage réseau, selon les besoins, automatiquement sans l'interaction avec fournisseur de services.
- 2. Large accès au réseau :** En utilisant des mécanismes standards, le cloud est accessible via le réseau en utilisant clients légers. Des exemples de clients sont les tablettes, les ordinateurs portables, les téléphones portables et les postes de travail.
- 3. Mise en commun des ressources :** Les ressources informatiques du fournisseur sont mises en commun pour servir plusieurs consommateurs en utilisant un modèle multi-locataires, avec différentes ressources physiques et virtuelles de façon dynamique affectés et réaffectés en fonction de la demande des consommateurs.

Il y a un sentiment de localisation l'indépendance en ce que le client n'a généralement aucun contrôle ni aucune connaissance sur l'emplacement des

ressources fournies, mais peut être en mesure de préciser l'emplacement à un niveau plus élevé (p. ex., pays, État ou centre de données). Les exemples de ressources comprennent le stockage, le traitement, la mémoire et la bande passante du réseau.

- 4. Élasticité rapide :** Les capacités peuvent être fournies et libérées de manière élastique, dans certains cas automatiquement, afin d'évoluer rapidement vers l'extérieur et vers l'intérieur en fonction de la demande. Aux consommateurs, les capacités disponibles pour l'approvisionnement semblent souvent illimitées et peuvent être appropriées en n'importe quelle quantité et à n'importe quel moment.
- 5. Service mesuré :** Les systèmes en cloud contrôlent et optimisent automatiquement l'utilisation des ressources en tirant parti d'une capacité de comptage à un certain niveau d'abstraction approprié au type de service (p. ex., stockage, traitement, largeur de bande et comptes d'utilisateurs actifs). L'utilisation des ressources peut être surveillée, contrôlée et signalée, ce qui assure la transparence tant pour le fournisseur que pour le consommateur du service utilisé.

6. Types de services dans les environnements de Cloud Computing :

Il existe généralement trois types de services dans le cloud computing. Ces services s'organisent en trois niveaux successifs : le niveau infrastructure (IaaS), le niveau plateforme (PaaS) et le niveau application (SaaS). La figure suivante (figure 2) montre ces types et niveaux [6].

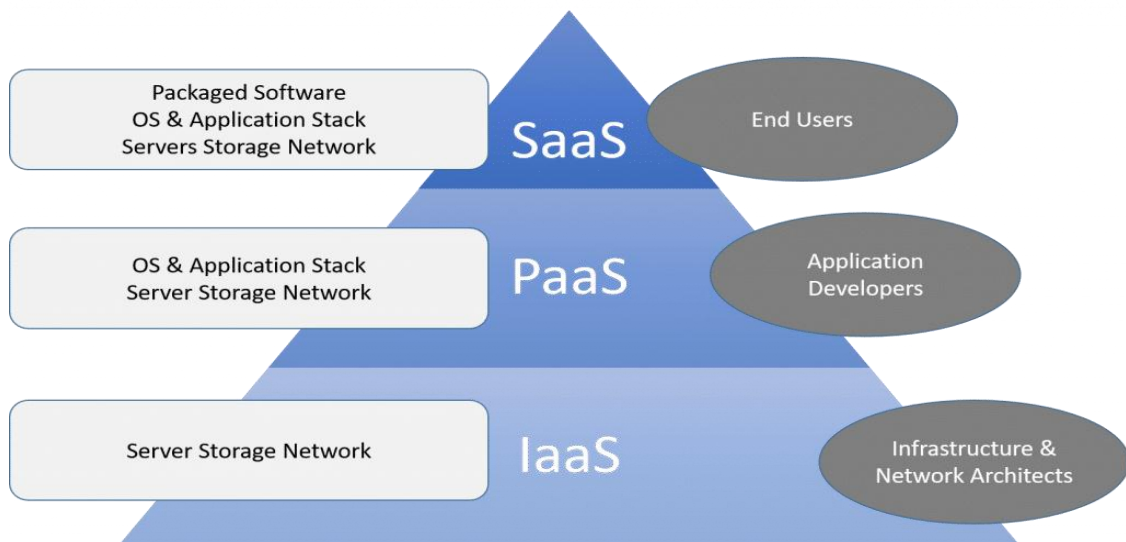


Figure 2: Types de services dans le Cloud computing [6].

Les principaux concepts : service, plateforme et infrastructure, sont définis par [7] comme suit :

- **Service** : Un service est un mécanisme qui est capable de fournir une ou plusieurs fonctionnalités, qu'il est possible d'utiliser dans le respect des restrictions et des règles définies par le fournisseur et à travers une interface.
- **Plateforme** : Une plate-forme est un système informatique fondamental qui comprend l'équipement matériel, les systèmes d'exploitation, et dans certains cas, des outils de développement d'applications et des interfaces utilisateurs sur lesquels les applications peuvent être déployées et exécutées.
- **Infrastructure** : L'infrastructure fait référence aux composants physiques qui sont nécessaires à un système pour exécuter ses fonctionnalités. Dans les systèmes d'information, ces composants peuvent contenir des processeurs, du stockage, d'équipement de réseau et dans certains les systèmes de gestion de base de données et les systèmes d'exploitation.

Les différents types de services dans l'environnement cloud sont définis par [7] comme suit :

6. 1 Software as a Service (SaaS)

Un logiciel ou une application qui est exécutée sur l'infrastructure d'un fournisseur est reconnue comme un service à condition que le consommateur dispose d'une autorisation d'accès limitée, la fourniture s'effectue via un client léger (un navigateur Web) ou une interface de programme pour envoyer des données et recevoir des résultats. Le consommateur n'a pas connaissance de l'infrastructure du fournisseur d'applications et dispose de droits limités pour configurer certains paramètres. Google Apps et Microsoft Office 365 sont deux exemples de ce service cloud.

6. 2 Platform as a Service (PaaS)

Dans ce modèle de service, le service fournit des exigences de base modérées, notamment le système d'exploitation, le réseau et les serveurs, ainsi que des outils de développement pour permettre au consommateur de développer les applications ou logiciels acquis et de gérer leurs paramètres de configuration. Un exemple de ce type de service cloud, Microsoft Azure.

6. 3 Infrastructure as a Service (IaaS)

Le consommateur a développé les applications requises et ne nécessite qu'une infrastructure de base. Dans de tels cas, les processeurs, les réseaux et le stockage peuvent être fournis par les fournisseurs en tant que services avec des dispositions pour les consommateurs. Amazon Web Services est le leader dans ce type de service.

6. 4 X as a Service (XaaS)

Anything-as-a-service, ou XaaS, fait référence à la diversité croissante des services disponibles sur Internet via le cloud computing par opposition à être fournis localement ou sur site. Également connu sous le nom de « tout en tant que service ». XaaS reflète le vaste potentiel des services cloud à la demande déjà fortement commercialisé et promu par des entreprises comme VMware et HP [8].

Anything-as-a-Service dérive le "X" dans son acronyme XaaS car il s'agit d'un terme fourre-tout pour tout. Du logiciel en tant que service (SaaS) au stockage en tant

que service, bureau en tant que service (DaaS²), récupération après sinistre en tant que service (DRaaS³), réseau en tant que service (NaaS⁴), infrastructure en tant que service (IaaS) et plateforme en tant que service (PaaS), et même des services émergents tels que le marketing en tant que service et les soins de santé en tant que service [8] .

7. Modèles de déploiement du Cloud Computing

Les services du Cloud Computing existants sont différenciés par rapport à leurs utilisations soit pour les entreprises ou autre type d'utilisateurs. Pour cela, le Cloud offre quatre modèles de déploiement, et ils sont décrit dans [9] comme suit :

7.1 Cloud public

L'infrastructure cloud appartient au fournisseur de services cloud, elle existe dans les locaux du fournisseur de cloud. Le grand public ou un grand groupe industriel peut accéder aux services pour une utilisation payante selon la méthode d'utilisation. Les utilisateurs se voient allouer les ressources du cloud à la demande, et les ressources sont fournies sur une base dynamique sur Internet.

Les petites et moyennes entreprises (PME) bénéficient dans une large mesure de l'utilisation de ce modèle de cloud. Les avantages des clouds publics sont l'indépendance de l'emplacement, la rentabilité, la fiabilité, la flexibilité, le coût du style utilitaire et une évolutivité élevée. Les inconvénients sont une faible sécurité et moins personnalisable.

7.2 Cloud privé

L'infrastructure cloud dans un cloud privé est exploitée uniquement pour une organisation. Il peut être géré par l'organisation elle-même ou un tiers. Le cloud privé peut exister sur site ou hors site. Les avantages des clouds privés sont une sécurité et une confidentialité accrues, un meilleur contrôle, des coûts et une efficacité énergétique. Les inconvénients sont l'évolutivité limitée en raison de ressources limitées, une tarification rigide et le cloud privé est limité à une zone particulière.

² DaaS : Desktop as a Service.

³ DRaaS : Disaster Recovery as a Service.

⁴ NaaS : Network as a Service.

7.3 Cloud hybride

L'infrastructure cloud hybride est une composition de deux ou plusieurs clouds (privés, communautaires ou publics). Chacune d'elles reste une entité unique mais est liée par une technologie standardisée ou propriétaire. Cette technologie permet la portabilité des données et des applications. Les avantages des clouds hybrides sont l'évolutivité, la flexibilité, la rentabilité et la sécurité. Les inconvénients sont les problèmes de mise en réseau et les conformités de sécurité.

7.4 Cloud communautaire

L'infrastructure dans un cloud communautaire est partagée par plusieurs organisations qui ont des préoccupations communes (par exemple, mission, exigences de sécurité, politique et considérations de conformité). Il est généralement géré par les organisations de la communauté ou par un tiers et peut être présent sur site ou hors site. Les avantages des clouds communautaires sont qu'ils sont sécurisés par rapport aux clouds publics et le partage des ressources entre plusieurs organisations.

Les inconvénients sont qu'il est moins sécurisé que le cloud privé et exige des politiques de gouvernance pour l'administration.

8. Architecture de référence du Cloud Computing

Alors que les détails de mise en œuvre varient largement d'un Cloud à l'autre, les architectures de Cloud Computing contiennent une série d'éléments communs [10] :

- **Couche de virtualisation** : La virtualisation des serveurs et des systèmes de stockage joue un rôle clé dans l'architecture de Cloud Computing, en offrant l'un des principaux avantages du Cloud : l'agilité. La couche de virtualisation permet aux fournisseurs d'autoriser ou de bloquer l'accès aux serveurs Cloud, en fonction des besoins des utilisateurs du service.
- **Stockage évolutif horizontal** : L'évolutivité est une autre marque de distinction de l'architecture de Cloud Computing. En matière de stockage, elle résulte généralement de technologies exploitant des ensembles importants de composants matériels standard, qui peuvent être facilement développés sur

demande, à moindres frais, au fur et à mesure de l'évolution des besoins en matière d'infrastructure et de stockage.

- **Mécanismes de prise en charge de multi- utilisateurs :** Un service Cloud doit permettre la séparation physique ou virtuelle des données stockées pour chaque utilisateur, et être capable de retracer le suivi de l'utilisation du service par utilisateur. Il est intéressant de noter que la prise en charge de multi-utilisateurs est une caractéristique intégrale, même dans les Clouds privés : dans ce contexte, les divers services ou groupes de travail d'une entreprise représentent les différents utilisateurs.
- **API Web :** Un autre élément clé de l'architecture de Cloud Computing est l'ensemble d'API Web (utilisant des méthodes standard telles que les appels HTTP RESTful, XML et SOAP) au travers desquelles les services Cloud peuvent être invoqués. Ces API permettent la mise à disposition des services par le biais d'un navigateur Web standard ou d'une autre application cliente http.

9. Acteurs du Cloud Computing

Le NIST dans son architecture de référence du Cloud Computing [11] (voir figure suivante) définit les acteurs majeurs, et pour bien comprendre le fonctionnement du Cloud computing, il est nécessaire de connaître les différents acteurs du domaine, et le rôle de chacun d'eux.

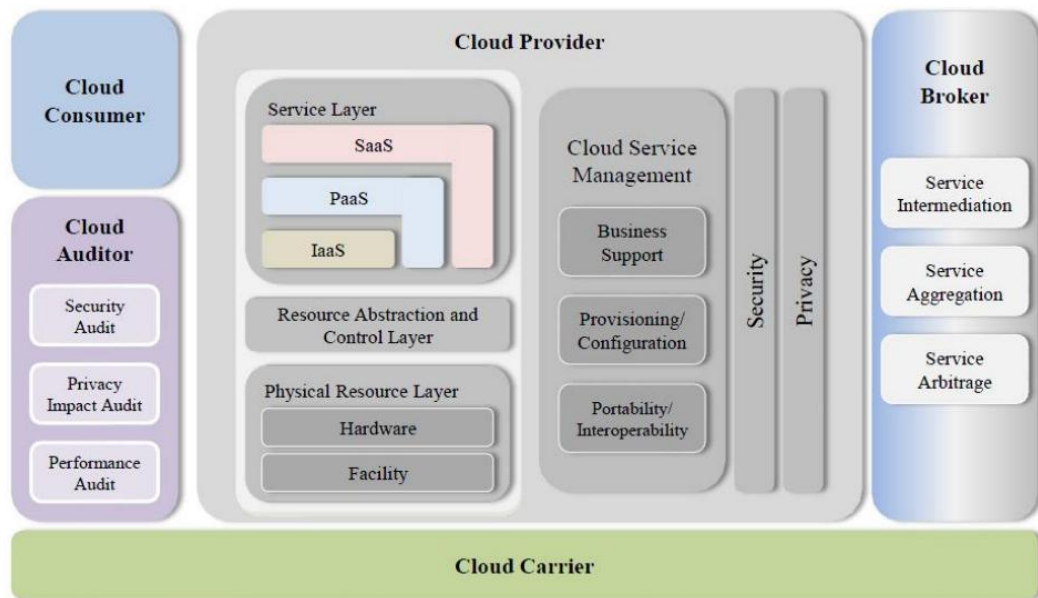


Figure 3: Architecture de référence du Cloud Computing [11].

9. 1 Consommateur Cloud (Cloud consumer)

Il s'agit de l'utilisateur qui consomme les services cloud. Le consommateur de cloud pourrait être l'un des suivants [12] :

- Les développeurs de l'entreprise qui créent les applications à l'aide de l'infrastructure cloud.
- Employés de bureau et consommateurs finaux qui accèdent aux applications de stockage et de productivité.
- Équipe de support informatique qui utilise les services cloud tels que les sauvegardes cloud pour compléter leurs ressources.

9. 2 Fournisseur Cloud (Cloud provider)

Il s'agit de l'entité qui fournit le service cloud. Le fournisseur de services cloud possède et contrôle la plateforme de cloud computing. Les services incluent SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service) [12], parmi les fournisseurs nous pouvons citer Amazon, Google, Microsoft et Sales force.

9.3 Courtier en Cloud (Cloud broker)

Un courtier en services Cloud, ou Cloud Broker, est une personne ou une entreprise tierce qui joue le rôle d'intermédiaire entre le consommateur d'un service de Cloud computing et les vendeurs de ce service. Son rôle peut se borner à faire gagner du temps à l'acheteur en étudiant les services proposés par différents fournisseurs, et en informant le client sur l'utilisation du Cloud computing dans le cadre de ses objectifs métier. Un Cloud Broker peut aussi être mandaté pour négocier des contrats. Dans ce cas de figure, il a le pouvoir de répartir les services entre divers fournisseurs dans un souci d'économie maximale.

Le courtier peut fournir une interface de programmation d'applications (API) et une interface utilisateur qui masquent alors ces complexités et permettent au client d'utiliser les services de Cloud comme s'ils provenaient d'un seul et même fournisseur [13].

Un autre type de courtier, parfois appelé « personneliste de Cloud » ou service Cloud « en marque blanche », sélectionne les services Cloud pour le compte d'un client, procède à leur intégration pour qu'ils fonctionnent ensemble et vend la nouvelle offre sous sa propre marque.

Un Cloud Broker peut aussi proposer des services complémentaires : assurer la duplication, le chiffrement et le transfert vers le Cloud des données du client, aide en matière de gestion du cycle de vie des données.

Un Cloud Broker désigne également une application qui permet de répartir les workloads entre différents fournisseurs de services Cloud. Ce type de courtier peut également être appelé Cloud Agent [13].

9.4 Relation entre les acteurs du Cloud computing

Le Consommateur de service cloud (Cloud Consumer) demande le service d'un fournisseur de service cloud (Cloud Provider) et le fournisseur offre le service aux clients avec un retour financier, donc il existe une relation commerciale entre le cloud client et le cloud fournisseur. Le courtier cloud (Cloud Broker) est un service médiateur entre plusieurs fournisseurs de cloud (Cloud Provider) et Consommateurs

de service cloud (Cloud Consumer), le cloud courtier offre un service relationnel avec le client et le fournisseur, il permet de négocier entre eux le meilleur service selon les besoins du client ainsi que d'autres opérations d'interopérabilité [14].

10. Avantages et inconvénients du Cloud Computing

Le cloud computing offre plusieurs avantages qui attirent les utilisateurs, des entreprises ou des particuliers. Parmi ces avantages, nous reprenons ceux cités dans [15] :

- **Gestion facile** : la maintenance de l'infrastructure est simplifiée, des applications faciles à être utilisées dans l'environnement cloud.
- **Réduction des coûts** : utilisation du cloud réduit le coût surtout pour les entreprises, il offre une puissance de calcul et stockage qui l'entreprise évite d'acheter un matériel coûteux et préfère utiliser les services cloud qui a un coût réduit et services riches.
- **Services ininterrompus** : pannes plus faibles sont causées par le cloud.
- **Gestion des catastrophes** : les fournisseurs des services effectuent des sauvegardes de manière circulaire pour sauvegarder les données dans des sites différents, dans le cas d'une catastrophe dans un site, les mêmes données existent dans ce site, ils existent dans un autre site.

D'un autre côté, l'augmentation de l'utilisation des services cloud fait apparaître des inconvénients dans l'environnement du cloud. Parmi les inconvénients on trouve le problème de sécurité et de confidentialité, connectivité et accès libre, manque de la fiabilité et de l'interopérabilité surtout dans la communication entre les clouds (inter-cloud et multi-cloud) [16].

11. Évolution du Cloud Computing

Avec l'augmentation de la demande des services fournis à travers le cloud, les requêtes des clients sont devenues plus complexes. La réponse à ces requêtes ne peut pas être effectuée par un seul cloud, ce qui nécessite une collaboration entre deux ou

plusieurs clouds pour offrir plus de services et permettre la réponse aux demandes des utilisateurs [14]. Cette évolution a conduit à l'apparition de l'inter-Cloud.

11.1 Inter-Cloud

Inter-cloud c'est de faire une interconnexion entre deux ou plusieurs fournisseurs des services du cloud [17]. C'est la communication entre plusieurs cloud pour découvrir et pour offrir plus de services qui ne peuvent pas être trouvés dans un seul cloud [14].

Le besoin de l'inter-cloud est nécessaire à cause de la limite des services et des ressources physiques (stockage, calcul,..) d'un cloud individuel. Par exemple, si les services physiques sont saturés, le fournisseur ne peut pas répondre à la requête du client, alors qu'avec l'inter-cloud il utilise le stockage, le calcul et d'autres ressources d'autres clouds.

L'environnement inter-cloud offre des avantages tels que la diversité des emplacements géographiques, une meilleure résilience des applications et évite le blocage des fournisseurs sur le client cloud. Les avantages pour le fournisseur de cloud sont une extension à la demande et un meilleur accord de niveau de service cloud [17].

Nous pouvons distinguer deux types d'inter-cloud, la fédération et le multi-cloud. Et chaque type est lui-même subdivisé en deux sous-familles celles qui sont citées par [17] :

11.1.1 Fédération de Cloud

La Fédération de Cloud c'est un inter-cloud où les fournisseurs collaborent leurs infrastructures du cloud pour partager les ressources entre eux, subdivisé en deux types : peer to peer et centralisé. La fédération de Cloud permet aux fournisseurs de développer leurs empreintes géographiques sans avoir à déployer leurs propres ressources informatiques dans le monde entier.

11.1.2 Multi-Cloud

Le Multi-cloud est un type d'inter-cloud computing, où un service ou un client utilise plusieurs clouds indépendants et la responsabilité de la gestion des ressources,

planification et l'approvisionnement est assurée par le client ou son représentant, il choisit les clouds et négocie les services qui lui conviennent. Il est aussi subdivisé en deux types : service et bibliothèque.

11.2 Topologies des différentes architectures de l'Inter-Cloud

Il existe quatre topologies pour les différents types de l'inter-cloud, deux topologies pour chaque type, cités par [17] dans ce que suit :

- **Fédération Inter-Cloud peer to peer** : Les clouds collaborent directement entre eux mais peuvent utiliser des entités distribuées pour les répertoires ou le courtage. Les clouds communiquent entre eux et négocient directement sans médiateur.
- **Fédération centralisée Inter-Cloud** : Les clouds utilisent une entité centrale pour former ou faciliter le partage des ressources. L'entité centrale fait office d'entrepôt où sont enregistrées les ressources disponibles des clouds.
- **Service Multi-Cloud** : Les clients accèdent à plusieurs clouds par le biais d'un service. Un service est hébergé par le client du cloud, soit en externe, soit en interne. Les services contiennent des composants de courtier.
- **Bibliothèques Multi-Cloud** : Les clients développent leurs propres courtiers en utilisant une API cloud unifiée comme bibliothèque. Les Inter-Clouds qui utilisent des bibliothèques facilitent l'utilisation des clouds de manière uniforme.

La figure suivante montre la topologie des différentes architectures inter-cloud.

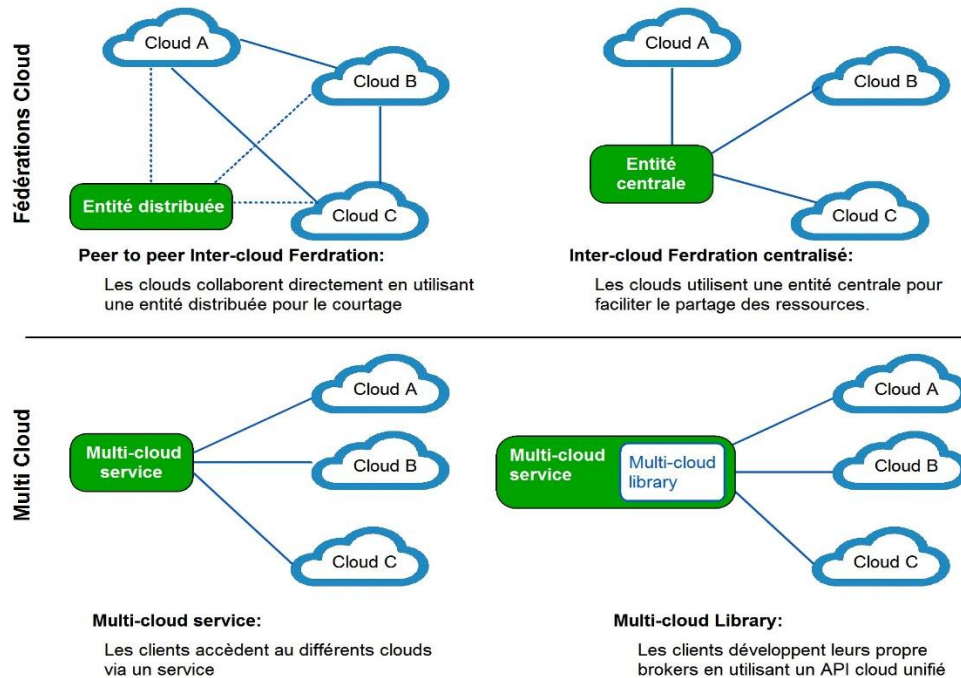


Figure 4: Topologie des différentes architectures inter-Cloud [17].

11.3 Scénarios de l'Inter-Cloud

La communication entre le client et le fournisseur du cloud peut être effectuée par deux différentes manières : avec un courtier ou sans courtier. Le courtier aide le client pour trouver le meilleur fournisseur ou service, le courtier offre des API qui gèrent les ressources cloud et d'autres API qui font la négociation des services aux clients. Avec l'accès direct, le client est obligé de gérer l'interopérabilité lui-même. La figure suivante montre l'architecture du courtier et les scénarios de communication [14].

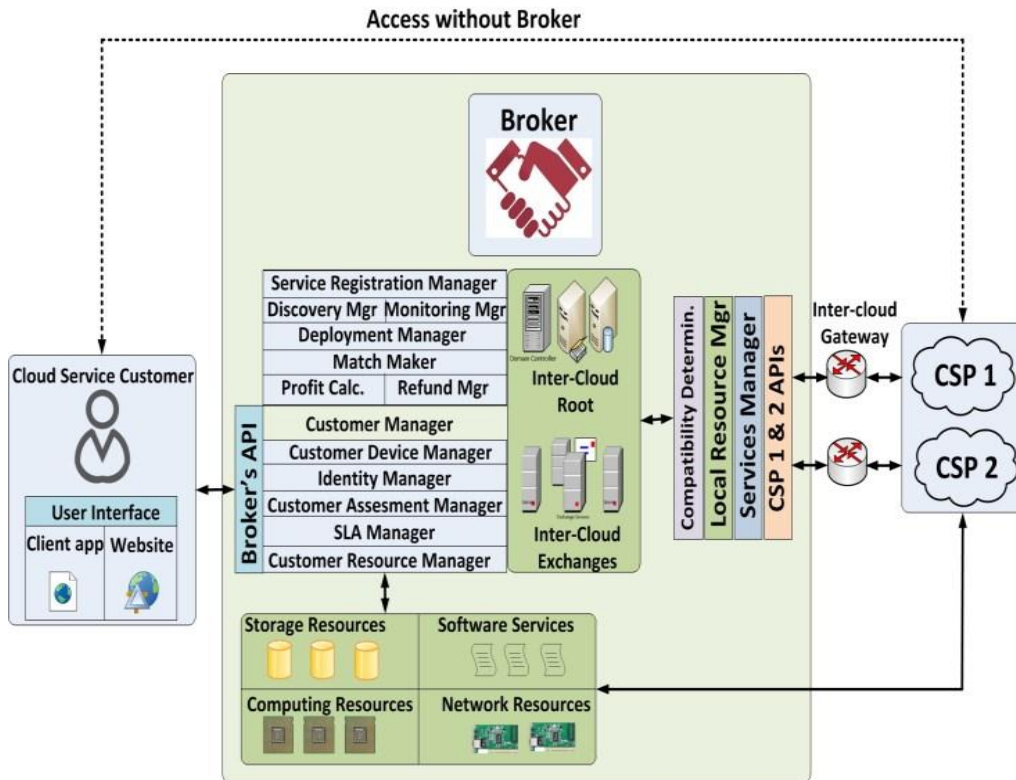


Figure 5: L'architecture du courtier et les scénarios de communication [14].

11.4 Défis de l'Inter-Cloud

Les principaux défis auxquels est confronté l'inter-cloud sont cités par [18] comme suit :

1. **Approvisionnement** : qui a son rôle englobe la découverte, la sélection et l'Allocation des services.
2. **Portabilité** : c'est la possibilité de migrer en live une machine virtuelle d'un hôte vers un autre, aussi la prise en charge des différents formats des données avec les différents droits de control et d'accès.
3. **Accord de niveau de service** : Chaque fournisseur de cloud a son propre contrat d'engagement(SLA) qui fixe surtout le niveau de service attendu, alors une nécessité est apparue c'est de mettre en place et de faire respecter un accord de niveau de service global. Aussi, il peut y avoir un contrat appelé Federation-Level Agreement (FLA) c'est l'accord de niveau fédération qui comprend l'ensemble des règles et conditions qui doit être signé par tous fournisseurs de

cloud en coopération. De plus l'utilisateur doit bénéficier d'un même niveau de service quel que soit le fournisseur, tout ça en prend en considération les questions d'ordre juridique.

- 4. La Sécurité :** c'est l'élément essentiel dans le cloud, les clients doivent faire confiance à un fournisseur de services de cloud pour la confidentialité et la sécurité de leurs biens, aussi la gestion de l'identité et d'autorisation doit être garanti dans l'environnement inter-cloud.
- 5. Monitoring :** c'est le système de surveillance qui rassemble les données de tous les composants de l'architecture cloud et fournit les données nécessaires à la gestion des infrastructures et des services. Les données de surveillance sont utilisées à différentes fins telles que l'application des accords de niveau de service (SLA), l'élasticité, la garantie de la qualité de service.
- 6. Économie :** Un fournisseur de services cloud est en mesure de répondre au pic des besoins en ressources en achetant des ressources à d'autres fournisseurs cloud computing. De même, lorsqu'un fournisseur de cloud dispose de ressources inutilisées, il peut vendre ces ressources, pour cela il y'a une nécessité d'un marché doté de facilités d'échange qui aide les fournisseurs à échanger des ressources entre eux, avec des prix soumis aux normes et règlements et sous un environnement de comptabilité et facturation clair et net.
- 7. Réseau :** garantir une connectivité de réseau sur des ressources distribuées, couvrant des ressources de différents fournisseurs, est une question difficile pour les utilisateurs et les fournisseurs, en impliquant la technologie de virtualisation des réseaux, l'adressage, un système de noms de domaines spécifique, et la multidiffusion surtout de contenu multimédia.
- 8. Autonomie :** Avec la complexité croissante des systèmes interconnectés tels que l'Inter-cloud, les tâches de gestion des systèmes deviennent trop complexes pour être effectuées uniquement avec une intervention humaine et une administration manuelle.

12. Conclusion

Dans ce chapitre, nous avons parlé des généralités du cloud computing pour bien comprendre ce domaine et pour avoir une idée globale sur l'environnement du cloud. Le but est de voir l'importance de cet environnement qui devient très large et très utile pour les utilisateurs et les entreprises. Aussi pour connaître les problèmes et les exigences qui existent dans un ou plusieurs environnements du cloud.

Les services offerts par le cloud donnent plusieurs avantages qui attirent les clients et les entreprises. L'utilisation des services cloud a beaucoup augmenté, ce qui pose plusieurs problèmes et exigences. Les fournisseurs de services cloud cherchent à améliorer leurs services pour satisfaire les besoins de leurs clients, l'évolution des services du cloud est obligatoire pour répondre à ces besoins.

Vu que plusieurs services nécessitent une collaboration entre plusieurs clouds pour les réaliser, l'inter-cloud est proposé pour résoudre les problèmes de la limite du cloud mais il pose aussi d'autres problèmes.

Le chapitre suivant sera dédié à l'étude de notre problématique dans le Cloud Computing à savoir la composition de services.

*Chapitre II : La composition des services dans le
Cloud*

1. Introduction

Dans le cloud computing, la connexion entre le fournisseur et le client est établie par Internet, où les clients utilisent un terminal pour exécuter les fonctionnalités offertes par le fournisseur de cloud en tant que service. Au niveau du Front End, les utilisateurs voulant satisfaire leurs demandes, où chaque requête est un ensemble de fonctionnalités, ils invoquent les services de cloud qui assurent l'exécution de ces fonctionnalités.

Au niveau de Back End, les services du cloud sont atomiques et chaque service assure l'exécution d'une seule fonctionnalité, alors que la requête de l'utilisateur est composée de nombreuses fonctionnalités. C'est pourquoi un service ne peut pas satisfaire la demande de l'utilisateur, mais il faut plusieurs services pour donner à l'utilisateur le résultat attendu, il doit communiquer et être combiné avec d'autres services. Dans ce cas, il s'agit de la composition des services ou « service composite ».

2. Composition des services dans le Cloud Computing

Dans le cloud computing, les fournisseurs des services cloud mets plusieurs services à la disposition des utilisateurs pour les consommer. Mais la plupart de ces services sont atomiques et ne peuvent pas satisfaire les besoins des utilisateurs. Pour répondre aux requêtes des utilisateurs, la composition des services est nécessaire lorsqu'un seul service ne peut pas satisfaire une requête complexe d'un utilisateur.

2.1 Définition de la composition

Dans le cloud computing, la composition des services est nécessaire lorsqu'un seul service ne peut pas satisfaire une requête d'un utilisateur. La composition des services cloud est un problème qui combine plusieurs services atomiques ou complexes pour produire un service plus complexe afin de répondre à une requête d'utilisateur.

Le problème de composition des services cloud est un problème d'optimisation, et il est considéré comme un problème NP-Difficile (Non déterministe temps Polynomiale-Difficile) [7]. C'est est un problème dont la solution est en temps exponentiel.

En général, la composition des services est un processus qui permet de former et exécuter une séquence des services atomiques pour réaliser une tâche complexe qui ne peut pas être réalisé grâce à un seul service atomique.

2.2 Défis de composition des services

La composition des services est une opération complexe qui comprend plusieurs difficultés et défis qui rends cette opération plus complexe et plus difficile à mettre en œuvre, à savoir [7]:

- Le changement rapide des valeurs de qualité de service pour chaque service peut poser un problème dans la composition parce que les critères de qualité de service peuvent se changent chaque instant et ça rendre la mesure des attributs de qualité de service plus complexe et influencer la qualité de service du service composé.
- La sélection optimale des services par un courtier dépend de la disponibilité d'informations complètes et actualisées sur les services. Faire face à plusieurs changements dans les caractéristiques du service pourrait entraîner la perte de certaines données.
- Décrire et mesurer les attributs de qualité de service des services de réseau. L'absence de consensus sur la définition et la description des paramètres de qualité de service des services de réseau parmi les services de cloud computing mondiaux est toujours un défi important pour les développeurs de cloud computing. L'absence de moyens convenus pour mesurer la qualité de service des réseaux est un autre problème qui n'est pas complètement résolu et qui devrait être pris en considération.
- La sécurité est aussi un défi important, conception et l'application des règles, des politiques et des instructions de sécurité font partie des responsabilités fondamentales des fournisseurs de services cloud.

- L'interopérabilité entre les services ou bien entre les cloud est un défi dans la composition, parce que chaque service peut être décrit de manière différent par rapport à autre, et aussi chaque cloud utilise des protocoles et des langages différents, ça rendre la communication entre les services et les cloud difficile et le processus de composition deviendra plus complexe.
- L'augmentation de nombre des services candidats pour réaliser chaque tâche pose un problème de passage à l'échelle.

3. Les modes d'exécution de la composition

Pour définir la séquence des services nécessaires pour la composition, il faut d'abord définir les tâches (services abstraits) pour avoir une séquence des services concrets pour les services composés.

Les services peuvent être combinés sur la base de quatre modes de base, le mode séquentiel (sequential mode), le mode en boucle (loop mode), le mode conditionnel (conditional mode) et le mode parallèle (parallel mode)[19] , qui sont montré dans la figure suivante :

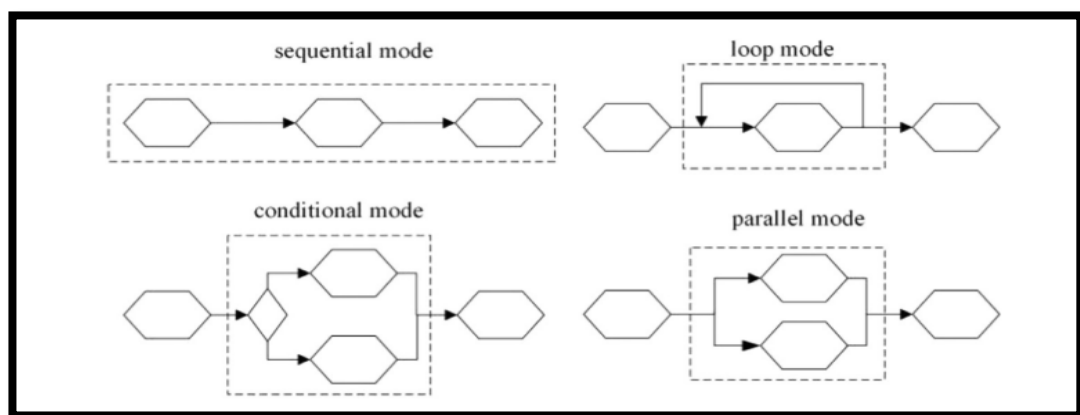


Figure 6: Les modes de composition de services[19].

La structure séquentielle fait référence à la dépendance séquentielle unidirectionnelle entre les sous-tâches. Elle est caractérisée par l'achèvement de la sous-tâche précédente avant le début de la sous-tâche suivante.

La structure parallèle signifie qu'il n'y a pas d'interaction d'information entre les sous-tâches, et que les sous-tâches sont dans un état complètement indépendant et peuvent être exécutées simultanément.

La structure conditionnelle fait référence à la substitution entre certaines sous-tâches, qui est caractérisée par l'apparition de sous-tâches avec une certaine probabilité, et la somme des probabilités des sous-tâches est de 1.

La structure en boucle signifie que certaines tâches nécessitent d'être exécutées plusieurs fois, et l'achèvement de la tâche nécessite de multiples itérations[19].

4. Étapes de composition de service

Comme il est connu, la composition des services cloud est un processus complexe qui nécessite plusieurs étapes. Selon[20], ce processus comprend trois parties qui sont : la conception et le déploiement de la composition, l'exécution et le suivi, et le retour d'information sur l'évaluation.

4.1 Conception et déploiement de la composition des services

Cette étape consiste principalement à construire des ensembles de candidats au service. Elle est divisée en deux étapes : la composition des tâches et la recherche de concordance des services. La décomposition des tâches vise principalement la description des tâches et des demandes. Une tâche est décomposée en plusieurs sous-tâches en fonction des exigences fonctionnelles fournies par les utilisateurs. Ce processus implique une analyse des exigences fonctionnelles, une analyse de la structure du processus et la construction d'une composition abstraite du service.

La recherche de service met en correspondance la composition abstraite du service avec la bibliothèque de services, qui forme un ensemble de services candidats par la recherche, la mise en correspondance et la sélection de services. Ce processus implique un raisonnement sémantique, une extension sémantique, une correspondance de similarité, un tri de l'ensemble des services candidats, une sélection de poids, une évaluation du service et d'autres opérations.

De nombreux ensembles de services candidats ayant des fonctions similaires et une qualité de service différente sont obtenus, et chaque service candidat a des sous-tâches correspondantes.

4.2 Exécution et suivi de la composition des services

Cette étape est aussi divisée en deux étapes : optimisation de la composition des services et surveillance des services. Les ensembles de services candidats générés lors de la conception et du déploiement sont évalués par la qualité de service. L'optimisation de la composition est réalisée grâce à un algorithme intelligent. Le chemin optimal de composition du service est sélectionné pour répondre à la demande de la tâche, et l'exécution de la tâche est surveillée.

La composition optimale du service est initialement formée par l'analyse des données et des contraintes dans l'ensemble des services candidats, et chaque demande de tâche est ensuite traitée. Le lien de surveillance se concentre principalement sur le schéma de configuration du groupement de la transformation dynamique, le lien de service et l'appel de processus, et le processus de coordination de l'exécution de la composition du service.

4.3 Retour d'information sur l'évaluation de la composition des services

Les résultats de l'exécution des tâches sont renvoyés aux utilisateurs pour évaluer les résultats de l'exécution. Les critères de qualité de service sont justifiés par la publicité et mises à jour en fonction de la satisfaction des utilisateurs, et les résultats de l'évaluation de la composition des services sont transmis à la bibliothèque de services en ligne.

En cas de résultats insatisfaisants, le système recomposera les services en fonction des besoins des utilisateurs et mettra à jour les statistiques sur la capacité de connaissance et d'évaluation en temps réel, qui serviront de référence pour le prochain appel de service.

5. Les services Cloud de type SaaS

Dans le cloud computing , le cloud de type Software en tant que service(SaaS) présente un programme ou une application complète en tant que service à la demande des utilisateurs [21] ,ou les services web sont composés pour produire ce programme ou cette application complète et les rendre accessibles via le web, avec un autre sens, plusieurs services sont regroupés pour satisfaire une requête SaaS [22]. Le grand défi dans le SaaS est la composition des services.

5.1 Les technologies des services Web

Le Cloud de type SaaS regroupe les technologies de service web et de l'architecture orienté service (SOA). Les services Web sont définis comme un ensemble d'opérations et de messages abstraits qui doivent souvent être combinés pour former un service complexe en utilisant des mécanismes de composition.

Architecture SOA est une architecture à faible couplage, Le principe du SOA est d'utiliser des interfaces de service ouvertes pour fournir des systèmes existants, tels que des bases de données ou des applications, des programmes en tant que services. L'architecture SOA est implémenté par le protocole SOAP (Simple Object Access Protocol).

5.1.1 Description d'un service Web

Les services Web sont décrits pour permettre aux utilisateurs de les découvrir et les consommer, la description des services aide pour choisir les bons services dans le processus de composition.

La description des services se base généralement sur le langage XML⁵(eXtensible Markup Langage), qui est un format de texte simple et très souple conçu à l'origine pour relever les défis de la publication électronique à grande échelle, XML joue également un rôle de plus en plus important dans l'échange d'une grande variété de données sur le Web et ailleurs.

⁵ <https://www.w3.org/XML/>

5.1.2 Langages de description

Il existe plusieurs langages pour la description des services web, et ils sont regroupées dans deux catégories : description syntaxique et description sémantique.

Une description syntaxique signifie que le service est interprétable par l'utilisateur et qu'il est invocable de manière statique. Description sémantique signifie que la machine peut interpréter et invoquer le service (recherche et invocation de service se fait dynamiquement).

Parmi les langages de description nous citons quelques langages les plus connues :

WSDL⁶: Le WSDL (Web Service Description Language) est un format XML pour décrire les services comme un ensemble de points terminaux fonctionnant sur des messages contenant des informations orientées soit vers des documents, soit vers des procédures. Les opérations et les messages sont décrits de manière abstraite, puis liés à un protocole de réseau et à un format de message concrets pour définir un point d'extrémité. Les points d'extrémité concrets connexes sont combinés en points d'extrémité abstraits (services).

Le WSDL est extensible pour permettre la description des points d'extrémité et de leurs messages, quels que soient les formats de message ou les protocoles de réseau utilisés pour communiquer.

OWL-S⁷: Web Ontology Language for Web Services est une ontologie de service web basée sur OWL, qui fournit un ensemble de constructions de langage de balisage pour décrire les propriétés et les capacités des services Web sous une forme non ambiguë et interprétable par l'ordinateur.

BPEL4WS⁸: WS-BPEL est un acronyme pour Web Services Business Process Execution Language. WS-BPEL 2.0 est une révision de l'acronyme original BPEL4WS (Business Process Execution Language for Web Services) 1.0 et 1.1. WS-BPEL est un langage basé sur XML

⁶ <https://www.w3.org/TR/wsdl>

⁷ <https://www.w3.org/Submission/OWL-S/>

⁸ <http://xml.coverpages.org/bpel4ws.html>

permettant aux utilisateurs de décrire les activités de processus d'affaires comme des services Web et de définir comment ils peuvent être connectés pour accomplir des tâches spécifiques. WS-BPEL est conçu pour spécifier des processus d'entreprise qui sont à la fois composés et exposés comme des services web.

WSDL-S : (Web Service Description Language-Semantic) est une version du langage WSDL qui permet l'utilisation de la sémantique.

Annuaire UDDI : (Universal Description, Discovery and Intégration) L'annuaire des service ou annuaire UDDI est une spécification pour un registre distribué de services web. Lorsque les services sont décrit par WSDL, il seront publiés dans UDDI pour que l'utilisateur peut les découvrir et les consommer[23].

La spécification UDDI définit un moyen de publier et de découvrir des informations sur les services web. Les registres UDDI utilisent la spécification UDDI pour publier des listes d'annuaires de services web.

La publication des descriptions est effectuée par le protocole SOAP (Simple Object Access Protocol), qui est un protocole d'échange des données structurés dans l'implémentation des services web basé sur XML.

6. Types de composition

La composition de service peut être fait dans un temps réel (en ligne) à la demande d'utilisateurs donc se fait dynamiquement, ou bien les services composés peut être générés et préparés avant la demande des utilisateurs. Donc la composition peut être statique ou bien dynamique[24].

Composition statique : Une composition statique utilise des services atomiques dans le contexte du client, ce type de composition crée des applications qui sont inflexibles et qui ne peuvent pas satisfaire les exigences de clients.

Composition dynamique : La composition dynamique cherche à répondre au requête du client en ligne, ce qu'exige que les transactions devraient être en temps réel. La composition des services ne peut être prédéfinie à l'avance et se fera également au moment de l'exécution.

7. Implémentation de la composition des services

Il existe différents systèmes pour la composition des services SaaS, chaque système a son mécanisme dans le processus de composition, ces mécanismes sont différents d'un système à un autre et peuvent être catégorisés selon différentes approches.

Plusieurs outils et logiciels peuvent être utilisés pour assurer la composition des services dans les cloud de type SaaS, puis la technologie SOA est utilisée pour l'implémentation de cette solution, dans [25], trois exemples sont cités comme approches qui assurent la mise en place de ces solutions :

Approche basée sur le Workflow : Le service abstrait est modélisé par un graphe créé manuellement mais il peut être changé dynamiquement, il définit l'ordre des nœuds dans le processus. Le graphe base sur trois types de nœuds : les nœuds de service, les nœuds de décision et les nœuds d'événements, les nœuds sont reliés entre eux par des arcs et chaque nœud il a un rôle particulier.

En effet, le nœud service décrit l'invocations de services atomiques ou composés, le nœud de décision spécifié les rôles et les alternatives pour contrôler le flux d'exécution et finalement, les nœuds d'événement autorisent la réception et l'envoi les différents types d'évènements.

Approche basé sur XML : c'est une approche de composition statique, dans cette approche on distingue deux type de composition qui sont : L'orchestration et La chorégraphie.

L'orchestration : les services web sont tous invoqués sous le contrôle d'un seul processus (orchestrateur) qui coordonne l'exécution des différentes opérations sur les services web.

La chorégraphie : au contraire l'orchestration, la chorégraphie n'a pas un coordinateur principal, alors ici chaque service sait exactement avec qui il doit interagir et quand doit effectuer les opérations. BPEL4WS c'est le langage le plus utilisé pour cette approche.

Approche basée sur l'ontologie : Le web sémantique est une extension de Web dans lequel les informations sont bien définies. L'élément essentiel pour réaliser un service web sémantique est le développement des langages convenables pour la description de contenu de service, parce que les données seront échangées de manière significative dans cette approche. Cette approche de composition est dynamique qui permet la composition en temps réel. OWL-S est un langage utilisé pour cette approche.

8. Les exigences de la composition des services

Le processus de composition des services doit respecter plusieurs contraintes surtout les contraintes exigées par l'utilisateur dans SLA et les critères de qualité de service.

8.1 Exigences fonctionnelles et non fonctionnelles

D'après [26], Il existe pour chaque service des contraintes fonctionnelles et les contraintes non fonctionnelles :

Les exigences fonctionnelles : sont les contraintes posées par l'utilisateur, ou l'utilisateur spécifié sa requête ou ça tâche souhaitée, les contraintes fonctionnelles spécifié comment exactement le service vont être exécuté.

Les exigences non fonctionnelles : sont généralement les exigences relatives au qualité de service(QoS) des services. Ces contraintes spécifié comment les services réalisent leurs fonctionnalités en terme de performances. Ces contraintes peuvent être divisés en deux catégories, les contraintes comportementales comme les attributs et les contraintes relatives à la sécurité et les paramètres de qualité de service mesurables comme le cout, temps de réponse, la disponibilité, etc.

8.2 Critères de qualité de service

La composition de service prend en considération les exigences non fonctionnelles des services pour optimiser le processus de composition à partir des critères de qualité de service.

Le vecteur de QoS pour chaque service est représenté par $Q(S_{ij}) = Q_1, Q_2, \dots, Q_k$ ou $Q(S_{ij})$ c'est k attributs pour j -ième service concrets défini pour i -ième tâche (service abstrait)[27]. Parmi les critères de QoS nous avons :

Cout d'accès : le cout d'accès est le cout nécessaire pour la consommation de services (certains services ne sont pas gratuits), défini par les fournisseurs de service cloud.

Temps de réponse : est défini comme l'intervalle de temps entre l'envoi et la réception de la réponse.

Disponibilité : représente la probabilité d'une invocation réussie [28].

Fiabilité : La fiabilité signifie la capacité du cloud à résister aux interférences extérieures et d'assurer le fonctionnement normal des services qui peuvent être obtenus par une base statistique[29].

Réputation : est une mesure essentielle pour la mesure de la confiance et fiabilité de service.[30].

Un critère de QoS peut être un critère à maximiser ou un critère à minimiser :

Un critère à maximiser : c'est les critères positifs qu'il faut maximiser comme la disponibilité ou la fiabilité.

Un critère à minimiser : c'est un critère négatif qu'il faut minimiser le maximum comme le cout et le temps de réponse.

8.3 Fonctions d'agrégations des valeurs de QoS

Pour calculer les valeurs de QoS pour le service composé, il faut agréger les valeurs de chaque attribut de QoS dans les différents modèles de composition (séquentiel, parallèle, conditionnel, boucle).

Le vecteur de QoS agrégé pour chaque composition peut être écrit comme $Q(SC) = Q_1, Q_2, Q_3, \dots, Q_K$ où Q_k représente le K -ième attribut agrégé pour toute composition arbitraire[27], et SC c'est le service composé. Le tableau suivant montre la fonction d'agrégation pour chaque attribut selon les différents patterns.

Critère de QoS	Séquentiel (n services séquentiels)	Parallèle (m services)	Conditionnel (appel S_{ij} avec probabilité pr)	Boucle (service S_{ij} boucle 1 fois)
Cout (Qco)	$\sum_{i=1}^n Qco(S_{ij})$	$\sum_{i=1}^m Qco(S_{ij})$	$\sum_{i=1}^m (Qco(S_{ij}) * pr)$	$l * Qco(S_{ij})$
Temps de réponse (Qtr)	$\sum_{i=1}^n Qtr(S_{ij})$	$\text{Max}(Qtr(S_{ij}))$ $1 \leq i \leq m$	$\sum_{i=1}^m (Qtr(S_{ij}) * pr)$	$l * Qtr(S_{ij})$
Disponibilité (Qds)	$\prod_{i=1}^n Qds(S_{ij})$	$\prod_{i=1}^m Qds(S_{ij})$	$\sum_{i=1}^m (Qds(S_{ij}) * pr)$	$Qds(S_{ij})^l$
Fiabilité (Qfb)	$\prod_{i=1}^n Qfb(S_{ij})$	$\prod_{i=1}^m Qfb(S_{ij})$	$\sum_{i=1}^m (Qfb(S_{ij}) * pr)$	$Qds(S_{ij})^l$
Réputation (Qre)	$\frac{1}{n} * \sum_{i=1}^n Qre(S_{ij})$	$\frac{1}{m} * \sum_{i=1}^m Qre(S_{ij})$	$\sum_{i=1}^m (Qre(S_{ij}) * pr)$	$Qre(S_{ij})$

Tableau 1: Fonction d'agrégation pour chaque attribut de QoS [27].

S_{ij} : représente le j-ième service concret à partir de l'ensemble de services candidats S_i .

9. Méthodes de sélection

La sélection des SaaS est très importante pour choisir les meilleurs services selon les contraintes fonctionnelles et non fonctionnelles. L'opération de composition des services cloud comprend trois méthodes selon le type de sélection : sélection locale, sélection globale et sélection intelligente [28] :

La sélection locale : Dans la méthode de sélection locale, pour la réalisation de chaque objectif ou tâche, on sélectionne le meilleur service parmi tous les services existants pour cette tâche. Bien que la méthode soit efficace, elle se limite à satisfaire uniquement les contraintes locales, comme par exemple trouver le service parmi des services qui satisfont aux critères de contraintes locales[27].

Cette méthode peut être résolu à partir d'une méthode d'aide multicritère a la décision ou bien par des méthodes d'optimisation.

La sélection globale : cette approche est une méthode d'optimisation globale. Elle vise principalement à satisfaire les contraintes globales. Les contraintes globales sont définies comme la satisfaction des valeurs de la qualité de service applicable à la composition des services individuels[27] .

La sélection intelligente : est aussi considéré comme une méthode de sélection globale. La différence est que le problème non linéaire peut être résolu par les algorithmes d'optimisation intelligents comme les algorithmes génétiques, les algorithmes d'optimisation des colonies de fourmis, les algorithmes PSO (Particle Swarm Optimization) et autres.

10. La composition des services dans un environnement multi cloud

Les anciennes méthodes de composition des services cloud ont fait la composition des services en utilisant un seul cloud, et ça ne peut pas toujours satisfait la requête d'un utilisateur, parce que peut ne pas trouver un ou plus d'un service convenable pour exécuter certaines tâches pour les services composés dans un seul cloud, on est besoin de composer les services à partir de multiple clouds pour trouver plus de services convenables [31][32].

L'utilisation de plusieurs clouds pour la composition dans un environnement multiple cloud augmente le cout et temps de réponse, nous sommes besoin de trouver une solution de ce problème qui recherche une séquence des services à partir d'un nombre minimum des clouds pour optimiser le cout et le temps [33].

Dans un environnement multiple cloud, le système de composition des services peut contenir plusieurs éléments inclue le combinateur de cloud, le convertisseur de composition, le planificateur de composition, les ontologies des services et le multiple cloud environnement. La figure suivante montre la structure complète de ce système [33] :

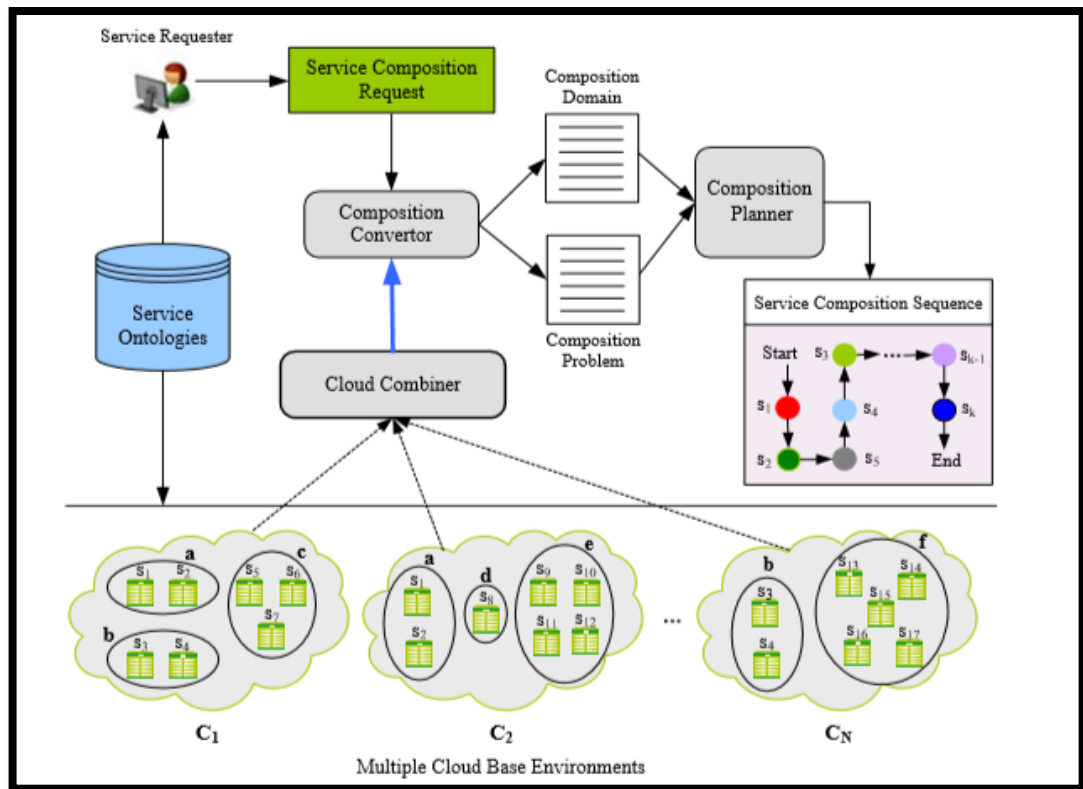


Figure 7: La composition des services dans un environnement multi Cloud [15].

Le demandeur de service (Service Requester) forme une requête de composition de services, qui contient une description de la demande de composition (Service Composition Request) en utilisant les ontologies.

Le combinateur de cloud (Cloud Combiner) sélectionne une combinaison des clouds qui sont appropriés à partir de l'environnement multi cloud en utilisant un algorithme d'optimisation. Après, le convertisseur de composition (Composition Converter) transforme la requête de composition à un problème de composition et un domaine composition ou la requête est divisée en tâches, pour que le planificateur de composition (Composition Planner) va s'exécuter et généré le plan de composition qui contient la séquence des services qui satisfait la requête d'utilisateur.

11. Travaux connexes

Pendant les dernières années, le problème de composition des services cloud a reçu beaucoup d'attention de la part de nombreux chercheurs. Le but est de chercher une solution optimale pour satisfaire les utilisateurs et pour améliorer la qualité de service.

Selon [19] ; Les différentes approches proposées dans la littérature peuvent être divisée en cinq catégories distinctes : les algorithmes classiques et les algorithmes à base de graphes (CGBA), les algorithmes combinatoires (CA), les approches à base de machine (MBA), les structures (ST) et les frameworks (FW). Parmi les approches proposé nous citons et discutons quelques approches ici.

- ❖ Kurdi et al. [31], ont proposé un algorithme d'optimisation combinatoire (COM2) pour la composition des services cloud dans un environnement multi cloud, cet algorithme assure que le cloud qui contient le grand nombre des services va être examine le premier parce que l'algorithme trie les clouds d'une manière descendante selon le nombre des services.

Ce mécanisme augmente la chance de trouver tous les services dans un minimum nombre des clouds dans le but de minimiser le cout et le temps. L'algorithme commence par accepte la requête de composition. Pour chaque itération ; l'algorithme choisi le cloud qui contient le nombre plus grand des services à partir de la liste des clouds triées par ordre décroissant.

Ensuite l'algorithme recherche les nouveaux services qui satisfait la requête, si l'algorithme trouve aucun nouveau service ; l'algorithme ignore le cloud et il passe à autre cloud pour chercher les nouveaux services qui satisfait la requête .si l'algorithme trouve un nouveau service qui satisfait la requête il lui ajoute à la liste de composition.

L'algorithme assure qu'un service le cloud sélectionne ne contient que des nouveaux services qui n'ont pas précédemment sélectionné dans la liste de composition par la soustraction des services la liste de composition des nouveaux services qui sont dans le cloud sélectionne. Pour chaque itération l'algorithme vérifier si la liste de composition contient tous les services nécessaires pour satisfait la requête d'utilisateur, si oui l'algorithme termine

son exécution et génère la liste de composition et l'envoyer à l'utilisateur. Sinon l'algorithme termine son exécution jusqu'à la satisfaction de la requête ou jusqu'à le parcours de tous les cloud de l'environnement multi cloud.

```
1: Input service request and MCE information
2: Output service composition sequence if available, otherwise, the algorithm terminates
3: Assumption clouds are sorted in decreasing order based on the number of services
4: //Initialize:
5:  $B \leftarrow \phi$  //B is the Combiner List of clouds sorted in decreasing order of number of services
6:  $n \leftarrow$  number of clouds in the MCE
7:  $P \leftarrow \phi$  //P is the Composer List of services
8: Get the users request R.
9: Select the cloud  $C_n$  that contains the largest number of services
10: //Check if  $C_n$  contains a new service that can fulfil the user request:
11: If  $((C_n \cap R) - P == \phi)$  // subtracting P is important to ensure cloud contains new services
12:   then go to 20
13: Else //the cloud contains new services that are can fulfill the users request
14:    $B = B + C_n$  //add the cloud to the Combiner List
15:    $P = P \cup (C_n \cap R)$  //add the services to the Composer List
16: //Check if P contains all required services that fulfill user request:
17: If  $(P == R)$ 
18:   then generate composition sequence // users request fulfilled
19: Else // users request has not been fulfilled yet
20:    $n = n - 1$ 
21:   If  $(n > 0)$  //if some clouds in the MCE has not been checked yet
22:     then go to 10 //check the next cloud
23:   Else //all clouds in the MCE have been checked
24:     Exit
```

Figure 8: Pseudo algorithm COM2 [31].

- ❖ Une approche méta-heuristique hybride pour une composition de services en cloud soucieuse de la qualité de service a été proposé dans [34].cet article propose une optimisation de l'essaim de particules hybrides(HSPO) technique qui combine l'optimisation des essaims de particules (PSO) et la mouche des fruits (FOA) pour effectuer le processus de recherche évolutive.

Cette solution pour améliorer le temps d'exécution, les valeurs du fitness et pour minimiser le taux d'erreurs, et pour réduire l'espace de recherche des service candidats, un algorithme Pareto pour optimiser la sélection des services candidats.

HSPO à plusieurs étapes, étape une pour l'initialisation, la deuxième étape c'est terminer l'algorithme HPSO et renvoyer la meilleure solution, si le critère d'arrêt est satisfait ; sinon, passer de l'étape 3 à l'étape 6, la troisième étape c'est

la phase PSO, cinquième étape c'est la phase FOA, et la dernière étape le retour vers l'étape 2 pour vérifier les conditions d'arrêt.

- ❖ Le travail proposé dans [24] présente une solution pour le problème de composition des services, en utilisant l'algorithme SFL qui est un algorithme méta heuristique d'évolution, Son principe est de générer une population d'une façon aléatoire comme les algorithmes génétiques classiques, cette population contient des frogs ou chaque frog représente une solution candidate, après l'algorithme calcul la valeur du fonction du fitness pour tous les frogs de la population, pour mesurer leurs performance. Ensuite l'algorithme trie la population en ordre décroissant, selon la valeur du fitness.

Après ça l'algorithme divise la population en groupes (memplexes), ou chaque memplexe contient un nombre fixe des frogs, ensuite il définit le meilleur et le mauvais frog pour chaque memplexe, et le meilleur frog dans tous la population, ensuite l'algorithme cherche à améliorer la position du mauvais frog dans chaque memplexe, en remplaçant les mauvais frog par autres suite à des opérations mimétiques, après l'algorithme vérifie les critères de convergence s'ils sont vérifiés ou pas. Si oui, il retourne le meilleur frog, sinon il refait les étapes du calcul du fitness jusqu'à l'évaluation.

Cette solution est une méthode de sélection globale qui utilise les contraintes de QoS dans la sélection.

- ❖ Dans l'article [35], les auteurs proposent Composition sécurisée des services avec contrôle des flux d'informations dans les services cloud, ou le contenu de service est vérifié et contrôlé d'abord avant la composition, cette approche assure la sécurité dans la composition des service dans un environnement multi cloud.
- ❖ La composition des services basé sur les agents a été proposé dans [36], les auteurs ont représenté le processus de composition par un système multi agent, chaque participant dans le cloud est représenté comme un agent.
La figure suivante représente l'architecture de système multi agents proposé. Dans ce travail les réseaux de Petri sont utilisés dans l'architecture des agents pour définir les services web.

service dans le cloud à base d'optimisation globale. La figure suivante montre l'algorithme (PSO+ABC).

Algorithm 1 A Hybrid (PSO+ABC) algorithm for cloud services composition

Input: M number of abstract services (tasks) i.e. Task1, Task2, ..., TaskM. Each abstract service is a set of N concrete services i.e. CS1, CS2, ..., CSN.

Output: Optimal service composition

Initialize Population size n_{pop} , QoS constraints, maximum iteration $maxitr$, cloud service CS_{ij} , Position Vector $p[j]$, Velocity Vector $V[j]$, Practical best position $p_best[j]$, Swarm best position $g_best[j]$ $P_Initial$.

Generate Initialize population randomly using Eq. 3

for $i = 1$ to $maxitr$ **do**

for $i = 1$ to $limit$ **do**

for $j = 1$ to n_{pop} **do**

 Execute selection and replacement of new particles.

 Calculate fitness function using Eq. 4

if $fitness_value(p[j]) > fitness_value(p_best[j])$ **then**

$p_best[j] = p[j]$

 Set g_best

if $fitness_value(p_best[j]) > fitness_value(g_best)$ **then**

 Update position and velocity vector using Eq. 1 and Eq. 2

end if

end if

end for

end for

 Scout Phase

for $j = 1$ to n_{pop} **do**

if $P_initial[j] = p[j]$ **then**

 Abandon old composition generate new one using Eq. 9

$P_initial = P$

end if

end for

end for

Return g_best

Figure 10: L'algorithme (PSO+ABC) [10].

Cette approche utilise une méthode intelligente pour la composition des services à base d'une sélection globale en utilisant les critères de QoS.

- ❖ Le travail de [32], propose un algorithme génétique pour la composition des services dans un environnement des cloud distribué, cette approche utilise un algorithme génétique pour la composition avec la prise en considération la latence de réseaux entre les clouds (Datacenter), pour optimiser le temps dans le processus de composition avec la sélection des services selon les contraintes de QoS posés par le client.
- ❖ Le travail dans [37] a étudié l'application de l'algorithme d'optimisation de l'apprentissage social (SLO) et son application dans la composition des services cloud tenant compte de la qualité de service.

Les différentes approches cherchent à optimiser la recherche, la sélection et le nombres des cloud pour améliorer la QoS de service composé. L'optimisation de certains paramètres peut dégrader la performance d'autres paramètres. Le tableau suivant montre des avantages et des faiblesses de quelques approches proposées[38] :

Nom	Approche	Avantages	Inconvénients
Singth et al. [39]	Basé sur les agents avec un Framework de double couches	Grande évolutivité. Cout minimal.	Temps élevé.
Kurdi el al. [31]	Algorithme d'optimisation dans un environnement multi cloud.	Temps faible, minimum nombre des cloud.	Sans prend en considération QoS.
Yu et al. [40]	Ant colony optimization algorithme.	Nombre petit des cloud.	Sans prend en considération QoS.
Wang et al. [32]	Genetic algorithm basé sur Skyline set.	Temps réduit. Cout réduit.	Sans tenir compte des exceptions et de l'incompatibilité des services.
Ivanovic et Carro.[41]	Se concentrer sur les orchestrations avec un flux de contrôle centralisé.	Grande évolutivité. Bonne optimisation.	Temps élevé.
Wang et al. [42]	Modèle d'apprentissage du renforcement multi-agents.	Haute efficacité Haute évolutivité.	Complexité élevé.

Tableau 2: Comparaison entre quelque approches de composition[38].

12. Étude comparative

Dans la suite, nous allons présenter une synthèse, sous forme de tableau, sur quelques études qui touchent la composition des services dans un environnement cloud de type SaaS en montrant les différents critères sur lesquels les auteurs ont basé.

12.1 Les critères de comparaison des solutions

Le problème de composition des services cloud a été étudié largement et spécialement dans cloud base SaaS, les chercheurs ont basé sur deux points essentiels dans ce problème. Le premier point c'est la sélection de service, ou les chercheurs pensent à obtenir les meilleurs services. Le deuxième point c'est la composition des services, pour réduire le nombre des cloud bases afin d'améliorer la qualité de service de la composition[43].

Aussi les approches proposé sont divisé on deux catégorie selon l'environnement, approches proposent des solutions à la composition dans un seul cloud et des approches dans un environnement multi cloud distribué [30].

Vu que le problème de composition est considéré comme un problème NP-Difficile, les solutions proposées ne sont pas des solutions exactes, ils peuvent être des solutions optimales. Les solutions peuvent être des algorithmes méta-heuristique comme les algorithmes génétique(GA),Ant colony optimization algorithme, Greedy Algorithme, particle swarm optimisation, ..etc[44] , ou des algorithmes heuristiques comme les algorithmes des arbres de recherche.

12.1.1 Comparaison selon environnement Cloud

L'environnement cloud utilisé pour la composition peut être un environnement mono cloud ou bien multi cloud.

- **La composition dans un environnement Mono-Cloud**

Les chercheurs ici utilisent les services d'un seul cloud pour effectuer le processus de composition. Plusieurs approches traitent le problème de composition dans un seul cloud. Le travail de recherche de C.Wu et al.[21] utilise les arbres de recherche pour la composition des services web dans le cloud de base SaaS qui utilise

les service à partir d'un seul cloud, aussi les travaux [45] et [46] utilisent des algorithmes génétique pour la composition dans un seul cloud.

- **La composition dans un environnement Multi-Cloud**

La composition des services se fait par la combinaison des services à partir de plusieurs fournisseurs de services. Plusieurs travaux ont traité ce problème dans un environnement multi cloud. Le travail de G.Zhou et al.[33] et H.Kurdi et al.[31] présentent les trois algorithmes d'optimisation combinatoire pour la composition des services dans un environnement multi cloud : All Cloud, Base Cloud, Smart Cloud et COM2. H.Kurdi et al.[47] ont utilisé cuckoo algorithme inspiré pour la composition des services dans multi cloud .

Les différentes approches proposées dans [32], [48], [49], [43] et [50] ont basé sur la composition dans un environnement multiple cloud.

Les différentes approches dans l'environnement multi cloud ont cherché à trouver les services a composé dans un nombre minimum des clouds pour minimiser le cout et le temps.

Le tableau suivant montre une classification de quelques approches de composition selon l'environnement du cloud utilisé.

Reference	Approche	Environnement
[31]	Algorithme d'optimisation Combinatoire COM2	Multi cloud
[49]	Algorithme cuckoo de recherche	Multi cloud
[24]	Shuffled Frog Leaping Algorithme	Mono cloud
[45]	Algorithme Génétique	Mono cloud
[47]	Cuckoo algorithme inspiré	Multi cloud
[28]	Artificial bee colony algorithme	Mono cloud
[43]	Programmation linéaire entière	Multi cloud
[51]	Aide multi critère a la descision(PROMETHEE)	Multi cloud
[52]	Multi agents distribuées	Multi cloud
[48]	Analyse de concept formelle	Multi cloud
[40]	Ant colony bee algorithme	Multi coud

[27]	PSO + ABC algorithme	Mono cloud
[53]	Composition dans les applications IoT	Multi cloud
[54]	Modelé mathématique basé sur la pénalité	Multi cloud
[32]	Algorithme génétique	Multi cloud
[36]	Approche basé sur les agents	Mono cloud
[33]	Smart cloud, All cloud, Base cloud combination	Multi cloud
[55]	Multi objective composition avec dépendances QoS	Mono cloud
[56]	Agents cooperatives	Multi cloud
[46]	Hybride (algorithme genetique + fruit fly)	Mono cloud

Tableau 3: Les approches de composition selon l'environnement Cloud.

Discussion

Comme il est mentionné dans le tableau, plusieurs travaux dans ce contexte utilisent un environnement multi cloud pour la résolution du problème de composition des services, mais la plupart des solutions sont dans un environnement mono cloud.

L'objectif de l'utilisation d'un environnement multi cloud, est d'augmenter la chance de trouver un plan de composition qui satisfait la demande utilisateur, où l'utilisation d'un seul cloud peut ne pas satisfaire les requêtes des clients. La figure suivante montre le pourcentage des travaux qui traitent le problème de composition selon le type de l'environnement.

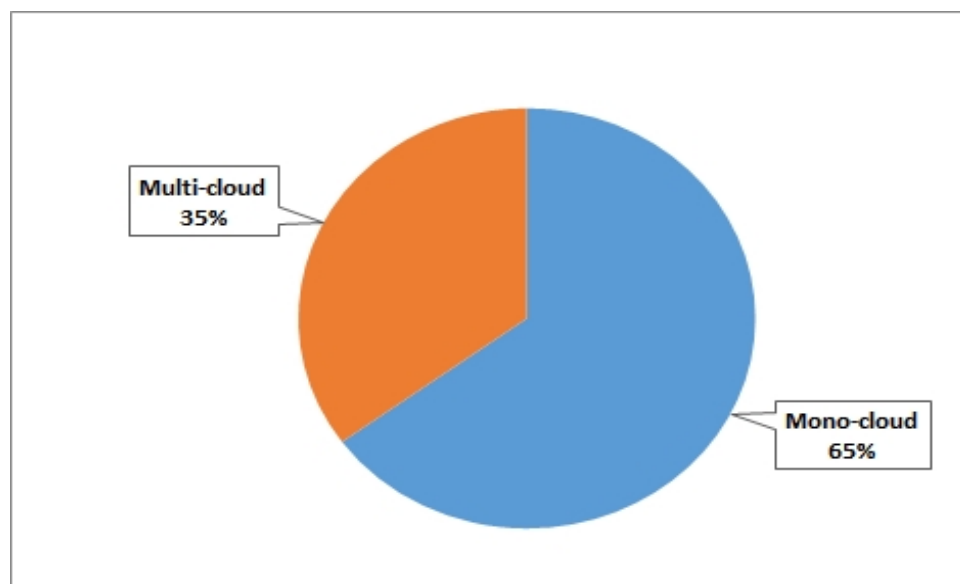


Figure 11: Traitement de la composition selon le type d'environnement.

12.1.2 Comparaison selon le type de sélection

La sélection des services pour la composition est très importante pour choisir les services qui conviennent selon les contraintes fonctionnelles et non fonctionnelles.

- **Sélection locale**

La composition est basée sur une méthode de sélection locale, différents travaux ont utilisé cette méthode de sélection. Le travail de S.Bharath Bhushan et al.[51], utilise une méthode d'optimisation locale, basé sur l'aide multicritère à la décision pour optimiser les critères de QoS locale pour chaque service dans la composition. H.Kurdi et al.[47][31] ont fait la sélection locale avec une méthode d'optimisation combinatoire mais sans prise en considération des critères de QoS.

- **Sélection globale**

La sélection globale est la méthode d'optimisation de sélection, qui prend en considération les valeurs de QoS globales pour le service composé. X.Wang et al.[57] proposent une approche de sélection globale optimale, basée sur les critères de QoS. R.Khanam et al.[27] proposent une nouvelle approche pour la composition à base d'une sélection globale, pour optimiser les contraintes de QoS. Plusieurs autres approches dans cette section, optimisation par les algorithmes génétiques dans [45],[32],[58] qui sont basés sur la sélection globale.

12.1.3 Utilisation des contraintes de QoS

La plupart des approches utilisent les contraintes de QoS pour optimiser l'opération de composition de services. L.Qi et al. [59] proposent une méthode de composition basée sur la qualité de service, qui prend en charge l'invocation de services multiplateformes dans un environnement cloud, toutes les approches de sélection globale utilisent les critères de QoS, et dans les méthodes de sélection locale, plusieurs approches n'utilisent pas les contraintes de qualité de services, comme le travail de Kurdi et al.[31].

Généralement, les méthodes de l'aide multicritères à la décision, utilisent les critères de QoS pour optimiser la sélection locale.

Chapitre II : La composition des services dans le Cloud

Reference	Approche	Type de sélection	Environnement	Utilisation de QoS
[47]	Multicuckoo inspiré	Local	Multi cloud	Non
[60]	Algorithme génétique (NSGA II)	Globale	Mono cloud	Oui
[33]	All cloud, smart cloud, base cloud	Locale	Multi cloud	Non
[57]	Basé sur QoS et la charge	Globale	Mono cloud	Oui
[61]	Un algorithme hybride impérialiste de recherche d'attraction compétitive et gravitationnelle	Globale	Mono cloud	Oui
[46]	Génétique + fruit fly algorithme	Globale	Mono cloud	Oui
[24]	Shuffled frog leaping algorithme	Globale	Mono cloud	Oui
[56]	Basé sur coopération des agents	Locale	Multi cloud	Non
[40]	Ant colony optimisation	Locale	Multi cloud	Non
[51]	Aide multicritère à la décision	Locale	Multi cloud	Oui
[32]	Algorithme génétique	Globale	Multi cloud	Oui
[45]	Algorithme génétique	Globale	Mono cloud	Oui
[38]	Agents basé et PSO Algorithme	Globale	Multi cloud	Oui
[34]	Approche hybride Meta heuristique	Globale	Mono cloud	Oui
[28]	Artificial bee colony algorithme	Globale	Mono cloud	Oui
[31]	Combinatorial optimization(COM2)	Locale	Multi cloud	Non
[49]	Algorithme cuckoo de recherche	Globale	Multi cloud	Oui
[54]	modèle mathématique basé sur pénalité.	Globale	Mono cloud	Oui

Tableau 4: Utilisation des contraintes de QoS selon l'environnement Cloud.

Discussion

Le tableau fait montrer plusieurs travaux traitent le problème de composition en utilisant les contraintes de QoS, où tous les travaux qui utilisent une méthode de

sélection globale, considèrent les critères de QoS dans la sélection, par contre, pour les travaux qui utilisent la sélection locale, la plupart d'eux ne considèrent pas les contraintes de QoS.

La figure suivante présente le taux d'utilisation des contraintes de QoS par rapport au sélection locale à partir du tableau précédent.

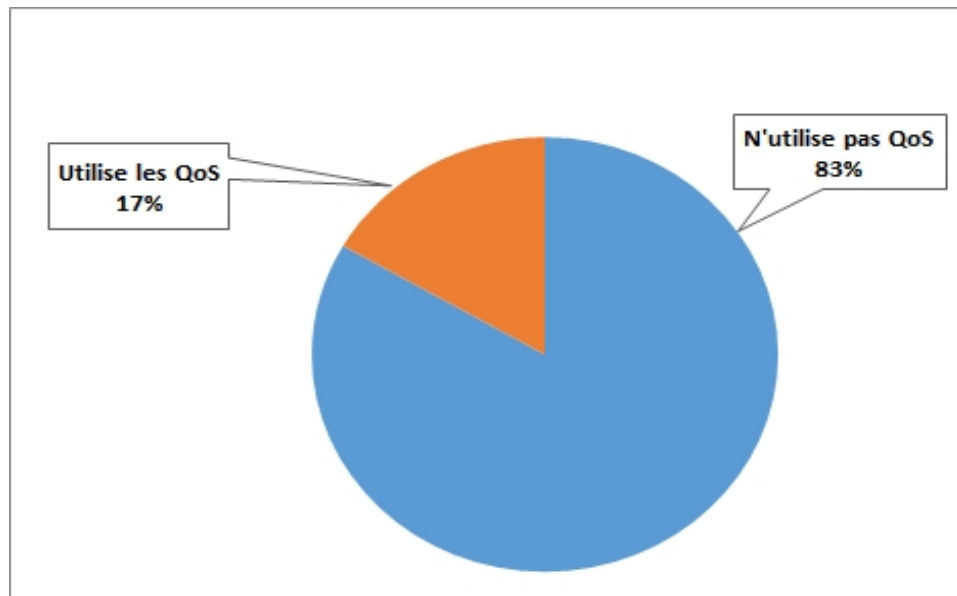


Figure 12: Utilisation des contraintes de QoS dans la sélection locale.

Cette figure montre que la plupart des approches qui utilisent la composition à base d'une sélection locale n'utilisent pas les contraintes de QoS.

La figure suivante, montre que l'utilisation des contraintes de QoS dans l'optimisation de sélection dans l'environnement mono cloud, est plus large par contre dans l'environnement multi cloud, ou la plupart des travaux ne traitent pas les contraintes de QoS.

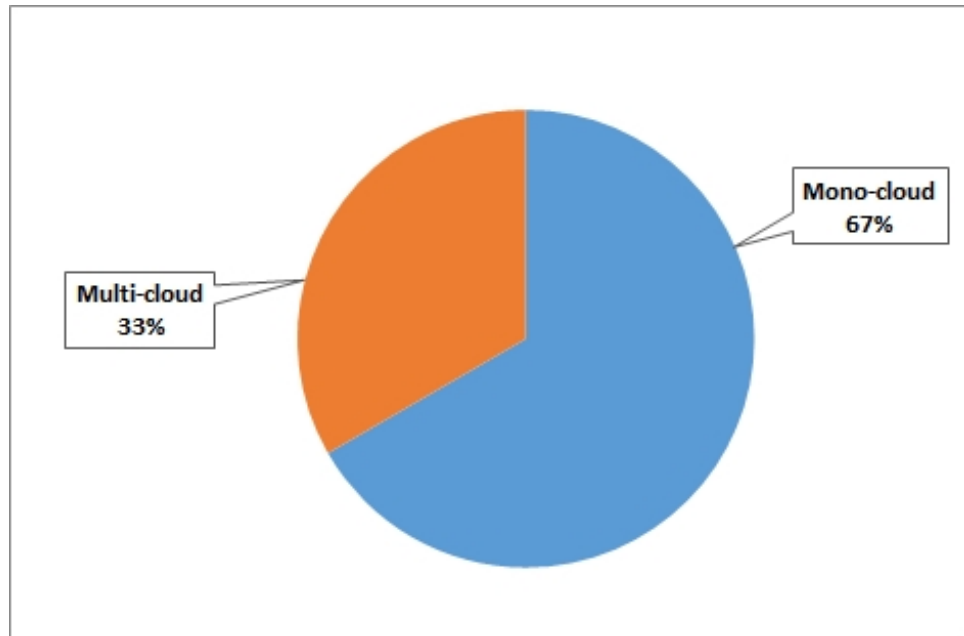


Figure 13: Utilisation des contraintes de QoS par rapport à l'environnement Cloud.

D'après ces résultats, nous pouvons conclure que la plupart des travaux qui traitent le problème de composition à base de sélection globale, se base sur un environnement mono cloud, et aussi la plupart des approches qui utilisent une méthode de sélection locale dans un environnement multi cloud ne prend pas en considération les contraintes de QoS.

Par conséquent, la plupart des travaux qui considères les contraintes de QoS sont à base d'une sélection globale, où la plupart de ces travaux utilisent un environnement d'un seul cloud.

12.1.4 Comparaison selon le type d'algorithme de solution

La solution du problème de composition nécessite un algorithme qui donnent une résultat optimale, vu que le problème de composition est considéré comme un problème NP-Difficile et qui ne peut pas avoir une solution exacte.

- **Algorithme non-heuristique**

Ce type d'algorithme, est généralement utilisé pour trouver un ensemble de solutions optimales aux problèmes d'optimisation. En règle générale, chaque problème d'optimisation peut obtenir une solution optimale unique, grâce à une recherche

exhaustive, mais il faut généralement un temps considérable pour trouver cette solution optimale [19].

Dans quelques approches, les chercheurs utilisent un algorithme non heuristique pour résoudre ce problème. S.Bharathan et al. [54] utilisent un modèle mathématique utilisant un algorithme de programmation linéaire entière pour la composition des services dans un environnement cloud distribué.

- **Algorithme heuristique**

Un algorithme heuristique est généralement désigné pour un problème spécifique et créé par "expérience", il vise à utiliser pleinement les caractéristiques spéciales du problème pour obtenir une solution réalisable de haute qualité, dans un temps de calcul raisonnable[19].

Plusieurs recherches utilisent des algorithmes heuristiques dans le problème de composition. Algorithme heuristique qui utilise les arbres de recherche pour la composition des services dans le SaaS proposé par C.Wu et al.[21] , aussi les travaux de kurdi et al.[31],[47] Pour la composition des services dans multi cloud environnement par des algorithmes heuristiques(COM2 et MultiCuckoo). Aussi, Algorithme heuristique d'une méthodes d'aide multicritères a la décision dans la composition des services[51], et le travail de [62] propose un algorithme heuristique pour la sélection des services rapide multi critères dans SaaS.

- **Algorithme méta heuristique**

Le méta-heuristique est une stratégie de haut niveau, pour explorer les espaces de recherche en utilisant différentes méthodes. Les algorithmes méta-heuristiques sont souvent conçus pour traiter des problèmes d'optimisation complexes, en utilisant des stratégies de haut niveau[19].

Les algorithmes intelligent sont des algorithmes méta-heuristiques, utilisées dans plusieurs travaux. Approche méta heuristique hybride par l'utilisation de l'algorithme Particle Swarm Optimization (PSO) et Fruit Fly (FOA) proposé par Bhushan el al.[34], et algorithme Hybride PSO+ABC proposé dans [27], utilisation des algorithmes méta heuristique génétiques pour la composition des services dans les travaux [32],[63],[45],[32],[64],[46] et [60], Algorithme méta heuristique ant colony

d'optimisation pour la composition des service dans le travail de Q.Yu et al.[40], Shuffeld Frog Leaping Algorithm proposé dans [24].

Généralement les algorithmes méta heuristiques sont des algorithmes intelligents, l'objectif d'utiliser ce type d'algorithme est l'optimisation à base des contraintes de QoS dans la sélection globale.

- **Basé sur les agents**

Le problème de composition des services a été résolu par un système multi agents dans certaines approches proposées. Système de composition dans un environnement multi cloud basé sur des agents coopératives, réalisé par M.Zaki Brahmi [56]. Les travaux dans [36],[65],[39] ont proposé des solution basé sur les agents et la coopération des agents pour résoudre ce problème.

Reference	Non heuristique	Heuristique	Meta heuristique	Base sur agents
[32]	✗	✗	✓	✗
[52]	✗	✗	✗	✓
[47]	✗	✓	✗	✗
[45]	✗	✗	✓	✗
[28]	✗	✗	✓	✗
[46]	✗	✗	✓	✗
[38]	✗	✗	✓	✓
[54]	✓	✗	✗	✗
[51]	✗	✓	✗	✗
[34]	✗	✗	✓	✗
[24]	✗	✗	✓	✗
[21]	✗	✓	✗	✗
[66]	✗	✗	✓	✗
[36]	✗	✗	✗	✓
[65]	✗	✗	✗	✓

Tableau 5: Traitement de la composition selon le type d'algorithme.

Discussion

Selon le tableau, plusieurs travaux utilisent des algorithmes méta heuristiques pour la résolution du problème de composition, vu que le problème est considéré comme un NP-Difficile, ce qui montre l'importance de l'utilisation des algorithmes intelligents méta heuristiques dans ce type des problèmes.

La figure suivante montre les résultats obtenus à partir du tableau.

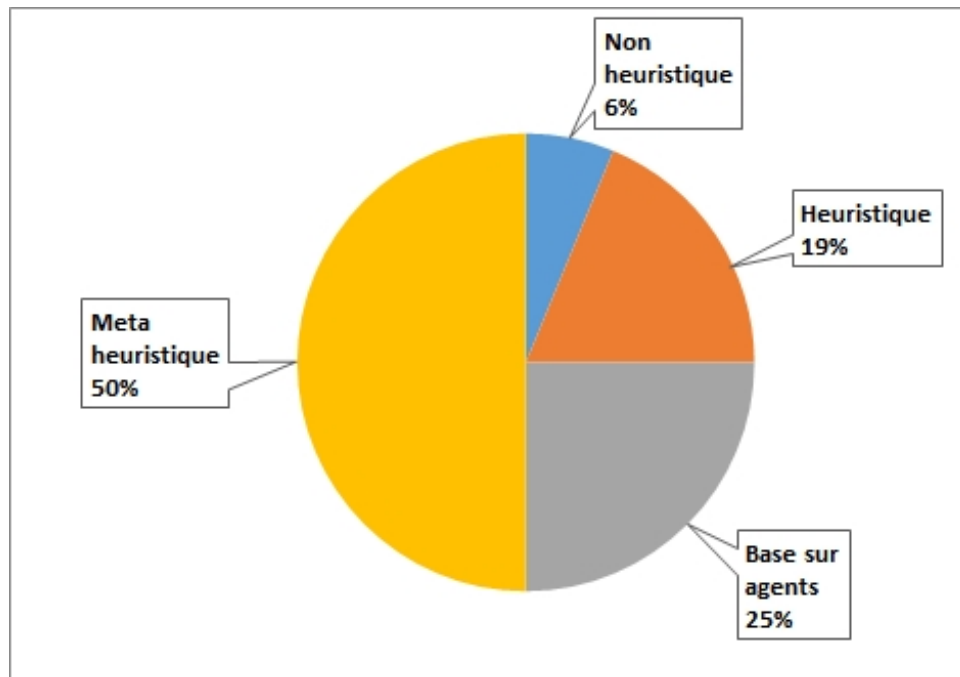


Figure 14: Statistique sur le type d'algorithme utilisé dans la composition.

12.2 Comparaison des différents travaux

Après l'extraction des différents critères, nous pouvons fixer les objectifs de recherche par la comparaison entre ces différents travaux.

Dans cette section, nous allons résumer les comparaisons entre les différentes approches, selon les objectifs des recherches, l'environnement cloud, et selon l'algorithme utilisé.

Dans le tableau suivant, différentes approches sont classifiées et comparées selon les critères suivants :

C1 : approche proposé dans un environnement mono-cloud.

C2 : approche proposé dans un environnement multiple cloud.

C3 : solution tenir en compte les contraintes de QoS.

C4 : solution à base d'une sélection locale.

C5 : solution à base d'une sélection globale.

C6 : solution avec un algorithme non heuristique

C7 : solution avec un algorithme heuristique.

C8 : solution méta-heuristique.

C9 : solution basée sur les agents.

Approches	C1	C2	C3	C4	C5	C6	C7	C8	C9
[50]	✗	✓	✓	✓	✓	✗	✗	✗	✗
[27]	✓	✗	✓	✗	✓	✗	✗	✓	✗
[62]	✓	✗	✓	✗	✓	✗	✓	✗	✗
[21]	✓	✗	✓	✗	✓	✗	✗	✗	✗
[28]	✗	✗	✓	✗	✗	✗	✗	✓	✗
[36]	✓	✗	✗	✓	✗	✗	✗	✗	✓
[39]	✓	✗	✓	✗	✗	✗	✗	✗	✓
[31]	✗	✓	✓	✗	✓	✗	✓	✗	✗
[46]	✓	✗	✓	✗	✓	✗	✗	✓	✗
[56]	✗	✓	✗	✓	✗	✗	✗	✗	✓
[51]	✗	✗	✓	✓	✗	✗	✓	✗	✗
[45]	✓	✗	✓	✗	✓	✗	✗	✓	✗
[54]	✗	✓	✓	✗	✓	✓	✗	✗	✗
[24]	✓	✗	✓	✗	✓	✗	✗	✓	✗
[32]	✓	✓	✓	✗	✓	✗	✗	✓	✗
Notre solution									

Tableau 6: Comparaison entre les différentes approches.

13. Conclusion

Dans ce chapitre nous avons étudié le problème de composition des services dans le cloud computing et spécialement dans le cloud de type SaaS. Nous avons défini les modes, les exigences et les différentes étapes de la composition. Aussi, nous avons présenté des travaux proposés dans ce contexte pour avoir une idée sur les différentes solutions proposées et pour définir les objectifs relatifs au processus de composition.

Nous avons terminé le chapitre par une conclusion, précédée par une étude comparative des différentes approches proposées, tenant en compte plusieurs critères, pour voir la différence entre ces approches et pour définir les objectifs de recherche pour chaque approche.

Cette étude nous a permis de fixer les objectifs et de définir les besoins de notre travail qui sera présenté en détail dans les chapitres suivants.

*Chapitre III : Modélisation de l'approche de
composition*

1. Introduction

Dans le chapitre précédant nous avons étudié la composition des services dans l'environnement du cloud computing, spécialement dans le cloud de base SaaS, et nous avons comparé les différentes solutions proposés dans la littérature. Cette étude nous facilite la réalisation de notre solution qui va être présentée dans ce chapitre.

Notre allons présenter une solution permettant de réaliser la composition des services cloud de type SaaS sur la base d'une sélection globale, cette solution va réaliser une sélection globale à partir des critères de QoS exigés par l'utilisateur pour avoir un service composé qui satisfait sa requête.

Aussi, nous allons définir l'architecture et la modélisation de notre système, avec la spécification des différents besoins du système, et nous allons effectuer les processus nécessaires de la composition à partir de la requête utilisateur jusqu'à l'exécution de la composition.

Démarche de travail

Tout au long des différentes phases de réalisation de notre projet, nous avons adopté la méthode eXtreme Programming (XP)⁹, qui est une méthode agile, utilisée en génie logiciel, plus particulièrement orientée sur l'aspect réalisation d'une application.

Cette méthode est mieux adaptée pour des équipes de tailles petites ou moyennes, des projets avec de nombreuses zones d'incertitude (risques) et lorsque les besoins sont vagues ou évoluent rapidement et nécessite une amélioration chaque fois.

L'Extreme programming a pour but principal la réduction des coûts du changement, et elle repose sur des cycles et des itérations rapides de développement dont les règles sont les suivantes[67] :

- **Planification** : consiste à diviser le projet en plusieurs itérations, et de crée un calendrier de livraison.

⁹ <http://www.extremeprogramming.org/>

- **Gestion** : consiste à dédier un espace de travail au membre, organiser des réunions quotidiennement, et surveiller l'état d'avancement du projet.
- **Conception** : utilisation des diagrammes et des schémas pour simplifier la représentation des différentes étapes du projet, avec possibilité de les reformuler chaque fois que possible.
- **Codage** : ici un ordinateur doit être dédié pour les tests unitaires des différentes parties du code, qui doit être rédigé conformément aux normes convenues.
- **Testes** : ici tous les codes doivent avoir des tests unitaires, et tout le code doit réussir tous les tests unitaires avant de pouvoir être remis.

Cycle de développement

L'eXtreme Programming repose sur des cycles rapides de développement, dont les étapes sont montrées dans la figure suivante, et qui sont comme suit [67]:

- Une phase d'exploration détermine les scénarios « client » qui seront fournis pendant cette itération.
- L'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels.
- Chaque développeur s'attribue des tâches et les réalise avec un binôme.
- Lorsque le produit satisfait tous les tests fonctionnels, il est livré.

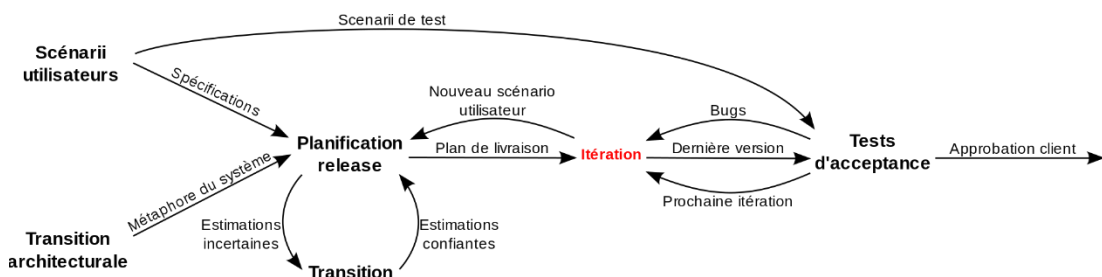


Figure 15: Le cycle de l'Extreme Programming[67].

Le cycle se répète tant que le client peut fournir des scénarios à livrer. Généralement le cycle qui précède la première livraison se caractérise par sa durée et le volume important de fonctionnalités. Après la première mise en production, les itérations deviennent plus courtes.

2. Spécification des besoins

Pour définir les besoins du système, il faut d'abord spécifier les acteurs qui interagissent avec le système :

Client : celui qui consomme les services.

Administrateur : qui fait la maintenance, le suivi des tâches et vérifie le fonctionnement du système.

Fournisseur des services : celui qui fournit les services.

Dans la suite, la figure montre le diagramme de cas d'utilisation, qui représente les interactions entre les acteurs et le système, ainsi qu'une explication des différentes actions.

- Le fournisseur des services cloud doit inscrire dans le système pour pouvoir s'identifier et publier les services, et faire la consultation et mise à jour pour la liste des services qu'il a publiés (supprimer, modifier,..).
- Le client aussi doit s'identifier pour pouvoir demander et consommer un service comme il peut évaluer le service composé.
- L'admin du système qui fait l'organisation des services, l'établissement et la modification des listes des services non dominés, et vérification des clients et fournisseurs après qu'il soit authentifié en tant qu'administrateur.

3.1 Récupération de la requête client : le client exprime sa requête à partir d'une interface client pour demander un service composé et le système récupère la requête pour traiter la demande.

3.2 Définir les tâches nécessaires pour réaliser la requête : après la récupération de la requête, le système va définir les tâches T_i ; $i = \{1, 2, \dots, n\}$, avec T_i représente la i -ième tâche nécessaire pour la réalisation de service composé et n représente le nombre des tâches qui sont nécessaires pour la réalisation de la demande d'utilisateur.

3.3 Définir les services pour réaliser chaque tâche : le système cherche la liste des services qui réalisent chaque tâche ou chaque tâche peut être réalisé par un seul service de la liste des services candidats.

À partir des services publiés par les fournisseurs des services, les services sont regroupés selon les tâches qui les relisent. Le système après la définition des tâches il définit les services candidats pour réaliser chaque tâche.

Donc chaque tâche T_i représente un ensemble des services qui ont la même fonctionnalité S_{ij} ; $\{S_{i1}, S_{i2}, \dots, S_{ij}, \dots, S_{im}\}$, avec S_{ij} représente le j -ième service qui réalise la i -ème tâche et m c'est le nombre des services pour chaque tâche. La tâche T_i nécessite un seul service S_{ij} pour être réalisé. Les services qui fait une même tâche peuvent avoir des différentes valeurs de QoS.

3.4 Élaboration de la sélection : après la spécification des services pour réaliser chaque tâche, un algorithme de sélection est exécuté pour sélectionner un plan de composition qui contient une séquence des services ou chaque service réalise une tâche selon les exigences posées le client, la sélection permet de trouver la séquence des services nécessaires pour exécuter le service composé SC, en respectant les exigences de QoS posés par l'utilisateur dans le SLA avec :

$SC = \{S_{1j}, S_{2j}, \dots, S_{nj}\}$ avec S_{ij} c'est le j -ième service qui réalise la i -ième tâche et n le nombre des tâches nécessaires pour exécuter le service composé SC.

3.5 Élaboration de la composition : le système exécute la séquence des services selon l'ordre et selon le mode d'exécution (séquentiel, parallèle, conditionnel, boucle) et il retourne le résultat d'exécution aux clients.

La figure montre l'architecture globale de notre système.

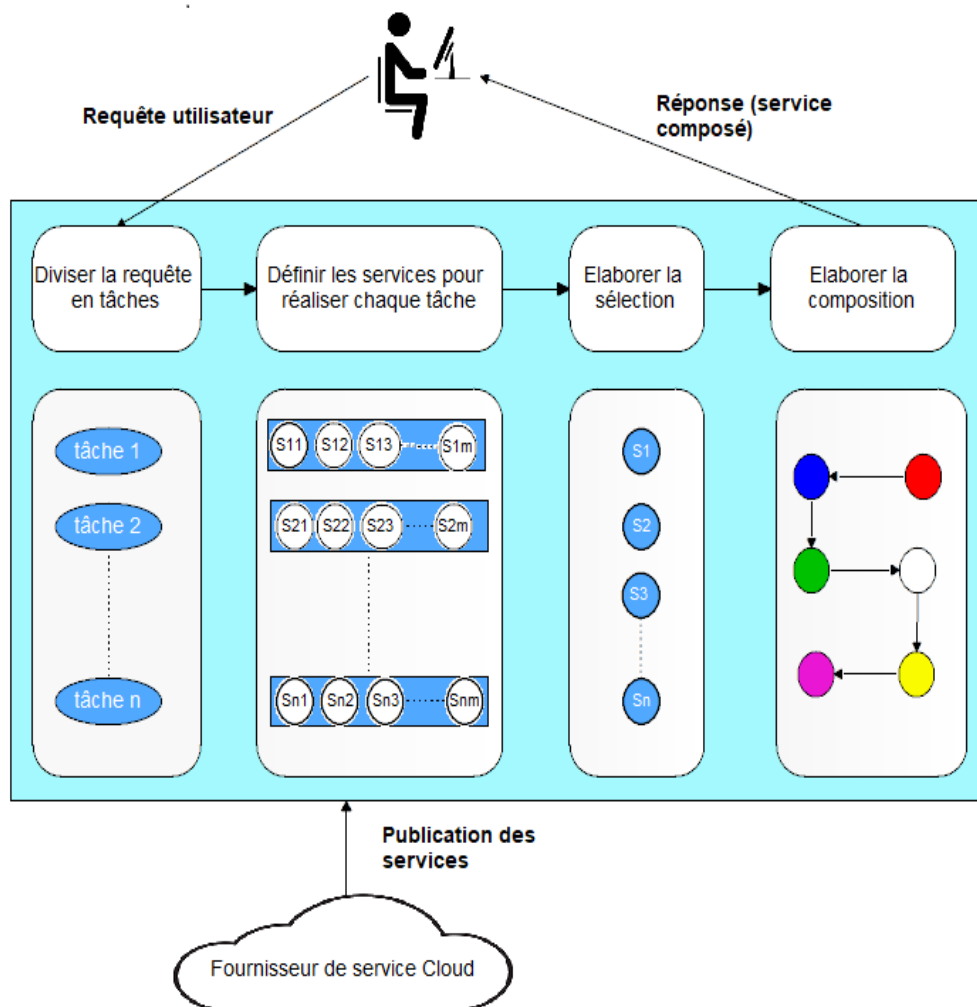


Figure 17: L'architecture globale du système de composition.

4. Scenarios de composition

Un scenario de composition est un cas d'utilisation, où nous sommes en besoin de la composition des services pour satisfaire une requête utilisateur. Dans ce scenario nous définissons les différentes étapes depuis la réception de la requête jusqu'à le retour du résultat au utilisateur.

✚ **Exemple d'un scenario de composition** : une demande d'un service en ligne pour un voyage touristique organisé.

- Cet exemple est modélisé comme un service composé parce que ce service ne peut pas être effectué grâce à un seul service.
- Supposons qu'il existe trois tâches pour la réalisation de ce service qui sont :

La réservation d'avion avec l'achat de tickets, la réservation hôtel avec tous les frais, et la réservation du transport pour visiter les différents sites touristiques.

- Plusieurs services disponibles pour la réalisation de chaque service, par exemple, pour la réservation hôtel plusieurs services qui offre ce service, mais ces services possèdent différentes valeurs de QoS, par exemple le prix se diffère d'un service à un autre.
- Le système va sélectionner les services selon les exigences posées par l'utilisateur, par exemple l'utilisateur demande un service de voyage organisé vers la Tunisie avec un prix de 45000 DA, alors les frais de tous les services nécessaires pour réaliser ce service ne doit pas dépasser 45000 DA.
- Après la sélection, le système retourne tous les informations pour le service demandé à l'utilisateur pour valider l'exécution de service.

✚ Pour tester l'exécution de la composition, nous allons utiliser deux services, un pour calculer la moyenne pour un étudiant, et un autre pour afficher l'évaluation de l'étudiant à partir de la moyenne, pour voir comment ce fait l'exécution d'un service composé.

5. Définition des critères de QoS

Nous avons défini les exigences dans la composition des services et nous avons parlé sur les critères de qualité de services, l'optimisation de la composition des SaaS se fait par l'optimisation des attributs de QoS.

Un services S_{ij} est défini avec ses propriétés non fonctionnels (critères de QoS) ou pour chaque service S_{ij} nous définissons un vecteur $Q_k(S_{ij})$; $k = \{1, 2, \dots, q\}$ avec Q_k

représente le k-ième critère de QoS et q représente le nombre des critères de qualité de services à utiliser.

Dans notre Système nous avons utilisé quatre attributs de QoS $Q_k(S_{ij})$, $k = \{1,2,3,4\}$

Avec $q=4$ et :

$Q_1(S_{ij})$: représente le cout(prix).

$Q_2(S_{ij})$: représente le temps d'exécution.

$Q_3(S_{ij})$: représente la disponibilité.

$Q_4(S_{ij})$: représente la réputation.

Les attributs de QoS ont catégorisé en critères positifs, qu'on besoin de les maximiser (disponibilité et réputation), ou bien des critères négatifs, qui doivent être minimiser (cout et temps d'exécution).

La figure suivante montre les catégories des critères de QoS utilisés dans notre système.

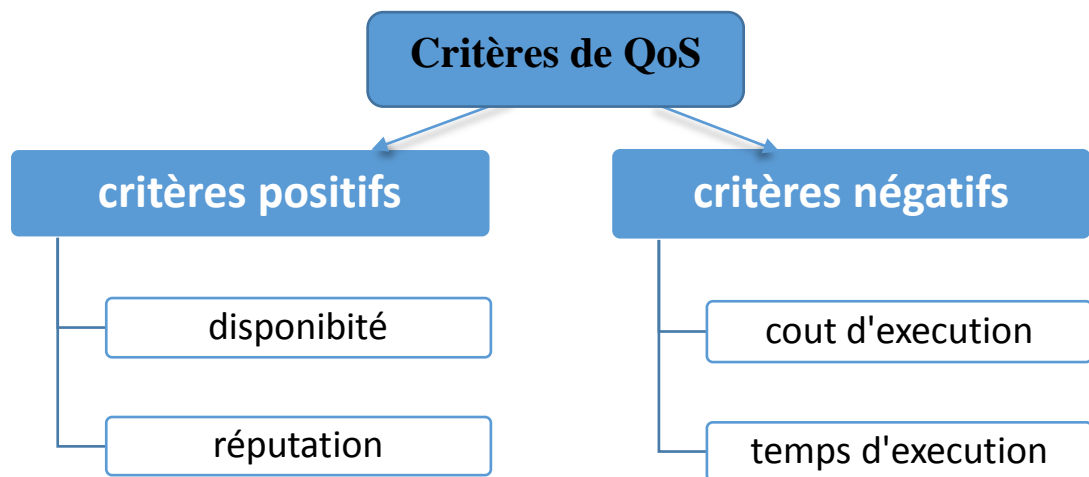


Figure 18: Types de critères de QoS utilisés.

6. Correspondance entre la requête et les services disponibles

Pour définir la correspondance entre la requête utilisateur et les services disponibles, le système propose une liste bien définie des services disponibles :

- Le système présente une interface utilisateur qui contient les services disponibles qui peuvent être réalisés par la composition.
- Une correspondance entre la requête et les services qui réalisent les tâches qui satisfont la requête d'un service composé. Par exemple un bouton qui fait référence vers le service nommé « voyage organisé » le système comprend que le service composé des tâches réservation avion, réservation hôtel et réservation de transport.
- Les fournisseurs des services sont obligés de publier les services, qu'ayant des propriétés fonctionnelles exigées par le système, où le système informe les fournisseurs des services de la liste des tâches (propriétés fonctionnelles) que le système supporte pour publier.

7. La sélection globale

La sélection c'est la plus importante opération dans la composition, pour choisir les meilleurs services qui satisfont les exigences posées par l'utilisateur à partir du SLA.

7.1 Service Level Agreement

Le client dans le SLA, signe un contrat avec le fournisseur de service SaaS, en précisant ses exigences de QoS pour le service composé. Dans la sélection globale, le demandeur de service pose ses exigences qui englobent les valeurs de QoS cumulé, pour tous services atomiques invoqués dans le service composé. Et donc le service composé doit respecter les exigences posées par l'utilisateur.

Par exemple un utilisateur demande le service voyage organisé, qui contient trois tâches, il exige un coût qui ne doit pas dépasser 35000 DA pour le service composé.

Si un plan de composition contient les trois services (Réserver Avion, Hôtel, Transport) qui réalisent les tâches du service demandé et les couts de chaque service sont (11000,21000,6000) respectivement.

Si nous calculons le cout du service global selon la formule définie dans le tableau (chapitre 2) nous trouvons que le cout = 38000 DA et ça dépasse le cout exigé par l'utilisateur qui est 35000 DA. Si nous remplaçons le service Hôtel avec un autre service pour la réservation d'hôtel qui a le cout 17000 DA, le cout cumulé devient 34000 DA qui ne dépasse pas 35000 DA, ça qui satisfait l'exigence de l'utilisateur.

7.2 Mesure de performance du service composé

La mesure de performance d'un service composé est très importante dans la sélection globale, pour identifier les services composés qui répond aux exigences d'utilisateurs. Et pour cela, il faut une normalisation des QoS, qui se fait comme suit :

- ✓ Pour mesurer la performance d'un service composé nous utilisons les valeurs de QoS des services atomiques dans le service composé.
- ✓ Nous commençons par le calcul des valeurs de QoS cumulé du service composé en utilisant le tableau du deuxième chapitre.
- ✓ Après nous sommes besoins de normaliser les valeurs de QoS selon les équations suivantes :
 - ❖ Pour les critères de QoS négatifs :

$$QNn(SC) = \frac{Qmax(n) - Qn(SC)}{Qmax(n) - Qmin(n)} \quad (1)$$

- ❖ Pour les critères de QoS positifs :

$$QNm(SC) = \frac{Qm(SC) - Qmin(m)}{Qmax(m) - Qmin(m)} \quad (2)$$

Avec $n = (\text{cout, temps})$ et $m = (\text{disponibilité, réputation})$. $Qmax(n)$, $Qmin(n)$, $Qmax(m)$, $Qmin(m)$ représente la valeur maximale et minimale du QoS cumulée des critères négatifs et positifs respectivement pour le service composé, et QN représente la valeur normalisée des critères de QoS pour le service composé SC.

7.2.1 La fonction de fitness

Après la normalisation des valeurs de QoS du service composé nous pouvons mesurer la performance de notre service composite en utilisant la fonction du fitness.

La fonction du fitness calculé avec la formule suivante décrite dans [32] :

$$f(SC) = \sum_{i=1}^2 QNni * wi + \sum_{j=1}^2 QNmi * wj \quad (3)$$

Avec w_i et w_j représentent les poids pour chaque critère de qualité de service posé par l'utilisateur avec $\sum_{i=1}^p w_i = 1$.

8. L'algorithme de sélection

Un algorithme d'optimisation permet de trouver le plan de composition qui répond aux demandes des utilisateurs. Le problème est encodé selon un problème d'optimisation pour permettre l'exécution de cet algorithme.

8.1 Encodage du problème

Avant l'exécution de l'algorithme, un encodage pour faire la correspondance entre le regroupement des services pour chaque tâche et le lien qui fait la référence pour les services dans la liste pour chaque tâche afin de choisir des combinaisons des services composites qui sont candidats d'être choisi dans le processus de sélection.

Dans notre algorithme, une solution candidate (individu) est encodé par un tableau d'entiers, ou chaque entier fait référence vers un service concret dans l'ensemble des services pour chaque tâche. La figure suivante montre l'encodage du problème.

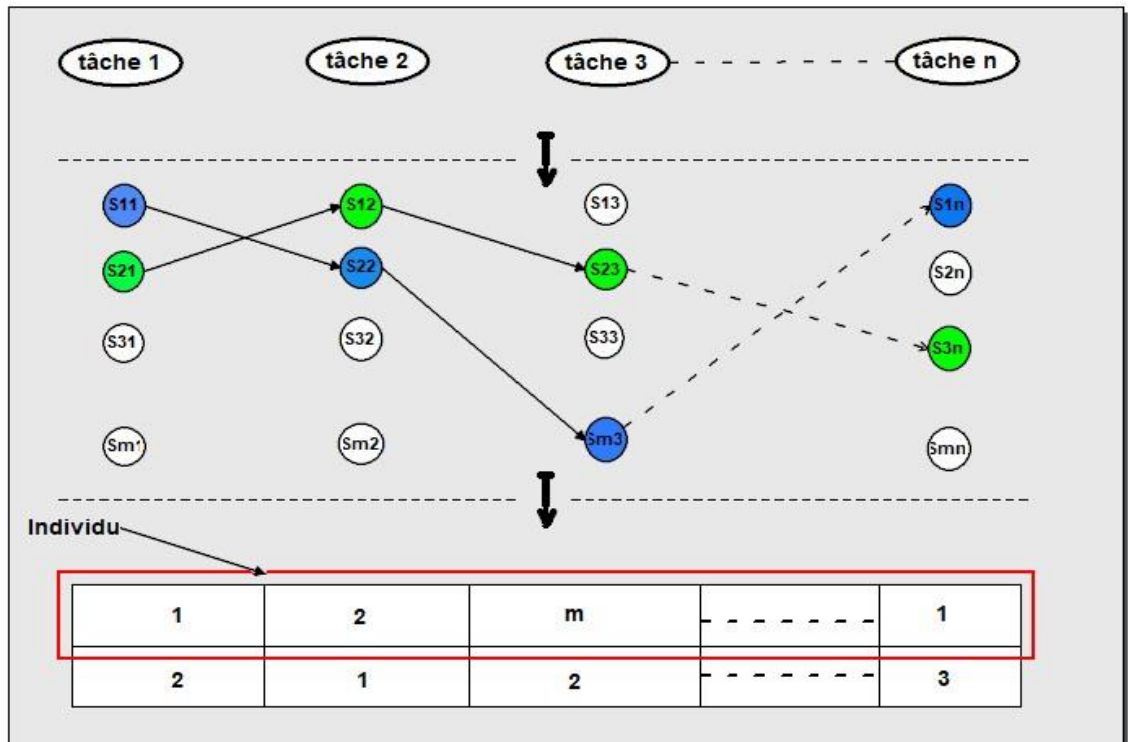


Figure 19: Encodage du problème.

8.2 Design de l'algorithme

L'idée de l'algorithme pris à partir de l'algorithme SFL (Shuffled Frog Leaping) proposé dans [24], qui est un algorithme méta-heuristique d'optimisation. Son principe est de choisir une combinaison des solutions candidates (services composites) à partir des listes des services, ou une solution candidate est considéré comme un Frog(grenouille).

Après, l'algorithme calcule la fonction du fitness pour chaque Frog, et il trie la population en ordre décroissant selon la valeur de son fitness, dans l'étape suivante l'algorithme divise la population en groupes, de taille fixe(memplexes) et l'algorithme effectue des opérations évolutives dans chaque memplexe, pour évoluer et améliorer les Frogs, et dans chaque itération l'algorithme vérifie si les critères de convergence sont vérifiés ou pas.

Notre algorithme proposé prend les mêmes étapes comme l'algorithme SFL, jusqu'à la division de la population en groupes (en SFL memplexes), la différence que notre

Chapitre III : Modélisation de l'approche de composition

l'algorithme remplace l'individu qui a la mauvaise valeur de fitness, dans chaque groupe, par un individu généré à partir de la liste des services non dominés.

La figure suivante présente le pseudo algorithme proposé :

```
Algorithm Sky-SFL

Inputs: listesServices, popSize, GrpSize, ContraintesClient ;

Output: composition_Plan SC;

Begin

1- Pop_init= generer_population(popSize, listesServices); // générer la population initiale
2- calculFitness(pop_init ); // calcul de la fonction de fitness pour chaque individu
3- triePopulation(Fitness, pop );// trie la population selon la valeur du fitness ordre décroissant
4- while( ! satisfait(ContraintesClient)) //vérifier si la condition d'arrêt est satisfaite ou pas
   {
5-   grps=diviser_population(GrpSize); // division de la population en groupes
6-   for (i=0 ; i<popsize/GrpSize; i++) { // parcours de tous les groupes d'individus.
7-     generer_non_dominer(pop_init , grps[GrpSize-1]);//remplacer le mauvais
8-     croisement(0,dernierIndice) ;
   }
9-   trieFitness(pop_init) ;
   }

End.
```

Figure 20: Pseudo algorithme proposé.

Les étapes de cet algorithme sont :

1. Initialisation des paramètres de système (la taille de population P, la taille des groupes N, ...) avec $P=N*M$.
2. Génération de la population initiale (un ensemble des individus) de taille P d'une façon aléatoire.
3. Calcul la fonction du fitness pour chaque individu de la population en utilisant l'équation Eq. (3).

4. Trie la population en ordre décroissant selon la valeur de la fonction du fitness pour chaque individu.
5. Vérifier si les conditions de convergence sont vérifiées (s'il existe une solution qui satisfait la demande d'utilisateur), si oui l'algorithme termine son exécution et il retourne l'individu qui a la meilleure valeur de fonction du fitness, sinon l'algorithme continue son exécution en allant vers l'étape suivante.
6. L'algorithme divise la population en groupes de taille N fixe ou chaque groupe contient M individus d'une façon ou le premier individu aller vers le premier groupe, le deuxième aller vers le deuxième groupe, le N-ième individu aller vers le N-ième groupe et le N+1 individu aller vers le premier groupe.
7. Pour chaque groupe l'algorithme définit le meilleur et le mauvais individu (le meilleur qui se trouve dans le premier ordre dans le groupe et le mauvais dans le dernier ordre dans le groupe).
8. Dans cette étape l'algorithme remplace le mauvais individu dans le groupe par un autre individu généré aléatoirement à partir de listes des services qui sont non dominés (skyline set). Nous avons pris l'idée de l'ensemble de l'horizon à partir de travail dans[32] qui utilise la notion du skyline set dans un algorithme génétique pour la composition des services .

La dominance : un service Sa domine un service Sb si et seulement si :

$$\forall Qn(Sa), Qm(Sb); (Qn(Sa) \leq Qn(Sb)) \wedge (Qm(Sa) \geq Qm(Sb)) \quad (4)$$

$$\exists Qn(Sa), Qm(Sb); (Qn(Sa) < Qn(Sb)) \vee (Qm(Sa) > Qm(Sb)) \quad (5)$$

Exemple : supposons un service 'Sa', caractérisé par un cout égal à 200 et d'une disponibilité de 0.95, et un service 'Sb' avec un cout de 150 et disponibilité 0.95. Si nous appliquons la loi de la dominance nous trouvons que 'Sb' domine 'Sa' car : selon l'équation Eq. (4) nous trouvons $0.95 \geq 0.95$ et $150 \leq 200$ (la disponibilité à maximiser et le cout à minimiser) et selon Eq. (5) : $150 < 200$, et ça montre que 'Sb' domine 'Sa'.

L'ensemble d'horizon (skyline set) : c'est un ensemble qui contient que les services qui sont non dominés. Le système applique un algorithme pour générer la liste des services non dominés. L'ensemble d'horizon contient les services

avec les meilleures valeurs de QoS par rapport les autres services qui n'appartient pas à cet ensemble.

Le calcul de l'ensemble d'horizon n'a aucune influence sur le temps d'exécution de l'algorithme, car le système définit cet ensemble une fois et stock l'ensemble dans la base de données. Chaque fois un fournisseur de service publie un nouveau service le système fait la mise à jour de la liste d'ensemble d'horizon pour la tâche spécifique.

9. Après le remplacement du mauvais individu, un opérateur de croisement (crossover) effectué entre l'individu meilleur dans le groupe et l'individu qui a été généré à partir de la liste des services non dominés.

Le croisement effectué par la génération d'un point de coupure l , aléatoirement et l'échange se fait entre les deux individus à partir du point $l+1$ jusqu'à n la taille des individus. La figure suivante montre l'opération de croisement.

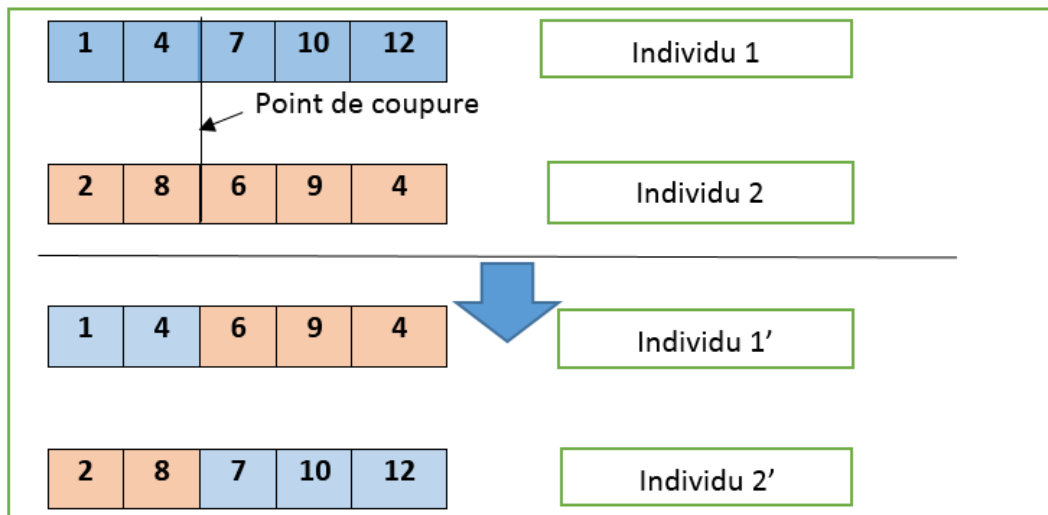


Figure 21: Opération de croisement.

Individu 1' et 2' représente les individus qui se produisent à partir de croisement entre individus 1 et 2.

10. Après le parcours de tous les groupes, l'algorithme refait le calcul du fitness et le trie de la population et il va retourner vers l'étape 5 pour vérifier si les

critères de convergences sont vérifiés ou pas. Si non l'algorithme poursuit les étapes de 6 jusqu'à 10 et refait ces étapes tant que les exigences de QoS posés par l'utilisateur n'ont pas satisfaits.

La figure suivante représente le schéma global (organigramme) de notre algorithme.

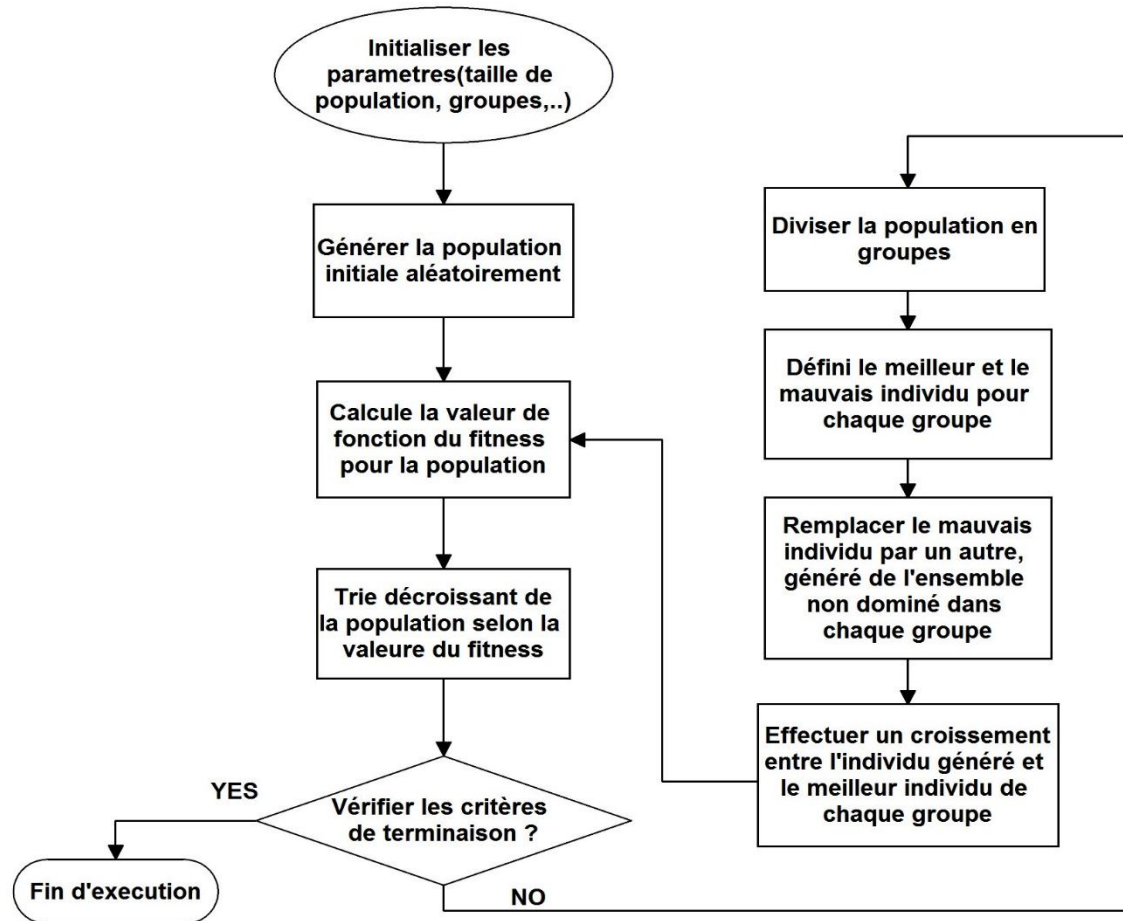


Figure 22: Organigramme de notre solution.

9. L'élaboration de la composition

L'opération de la sélection produit un service composé prêt à être utilisé, et exécuté par l'utilisateur. Le déploiement et l'exécution du service sont les dernières étapes dans le processus de composition.

- ✓ Les fichiers de services sont créés à base d'XML. Ces services et leurs différentes fonctionnalités et points d'extrémité sont décrits à travers la génération des descripteurs WSDL, puis publiés dans un annuaire UDDI pour permettre de les découvrir, et à la fin, la publication des descriptions est effectuée par le protocole SOAP.
- ✓ Le système exécute le service composé en utilisant un orchestrateur qui crée des instances pour les services sélectionnés et invoque les services et retourne le résultat pour l'utilisateur.
- ✓ Nous utilisons une servlet comme un orchestrateur pour l'exécution de la séquence des services nécessaires pour exécuter le service composé et elle retourne le résultat final au utilisateur.
- ✓ Le système garde la trace de service composé pour la réutilisation afin d'optimiser la sélection.

10. Conclusion

Dans ce chapitre, nous avons modélisé notre solution, aussi nous avons défini et présenté les différents modules de notre système, ainsi que les fonctionnalités de chaque module, pour permettre et faciliter la mise en œuvre de cette solution, et faire l'expérimentation et simulation dans le but de conclure les résultats qui confirment la validité de notre solution.

Dans le chapitre suivant, nous allons présenter la solution avec les différentes expérimentations et la simulation pour tester tous les modules de notre système, et nous allons présenter et discuter cette solution, et la comparer avec quelques travaux connexes.

Chapitre IV : Expérimentation

1. Introduction

Après l'étape de modélisation de notre approche, nous arrivons dans ce dernier chapitre à l'implémentation et la mise en œuvre de la solution proposée, le test selon des scénarios, la récupération des résultats pour la comparaison, l'analyse et la discussion.

Pour cela, nous avons utilisé un ordinateur qui tourne avec un système d'exploitation Windows 10, alimenté par un processeur Intel Core i5, et une RAM de 6 GO, aussi nous avons utilisé des outils de développement qui nous ont semblé être un bon choix vu les avantages qu'ils offrent.

2. Outils et environnement de développement

Notre choix est tombé sur le langage de programmation java, et sa plateforme Java EE, qui signifie Java Enterprise Edition, axées sur le développement, le déploiement, et de la gestion des applications d'entreprise centralisées sur un serveur.

Java est un langage de programmation orienté objet, libre, simple et portable. Ce langage est intégré dans l'environnement de développement Eclipse, et utilise JDK (Java Development Kit) qui regroupe l'ensemble des éléments et outils nécessaires pour le développement. D'autres outils sont utilisés, tel que le serveur d'application Tomcat, et Docker.

2.1 IDE Eclipse

Eclipse est un projet de la fondation Eclipse, et un IDE qui cherche à créer un environnement de production de logiciels englobant toutes les activités de programmation, modélisation, conception, test, gestion de configuration et reporting, aussi vise à supporter tout langage de programmation.

2.2 Java Development Kit JDK

Le Java Development Kit est un ensemble de bibliothèques logicielles de base et spécifique du langage de programmation, nécessaire pour le développement des

applications Java, comporte aussi les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java. Parmi les JDK disponibles nous pouvons citer le Java SE et Java EE. Ce dernier, nous l'avons choisi parce que il est mieux adapter au développement des applications web.

2.3 Apache Tomcat

Nous avons opté pour l'utilisation du serveur Apache Tomcat, qui est un serveur d'applications Java et combine aussi un serveur web, et une implémentation open source des technologies Java Servlet, Java Server Pages et Java Expression Language. Il permet de produire une représentation HTML à une requête après avoir effectué un certain nombre d'actions tel que la connexion a une base de données[68].

2.4 Docker

Le terme « Docker » désigne plusieurs choses, nom d'entreprise, nom d'un projet et aussi une technologie ou nom d'un logiciel, ce que nous intéresse est ces deux derniers, qui sont les solutions qui nous a permet de faire la simulation pour le test de notre approche proposée.

Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur. Avec la technologie Docker, vous pouvez traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs vous offrent une grande flexibilité. Vous pouvez les créer, déployer, copier et déplacer d'un environnement à un autre, ce qui vous permet d'optimiser vos applications pour le cloud.

La technologie Docker utilise le noyau Linux et des fonctions de ce noyau, elle vise à séparer les processus afin qu'ils puissent s'exécuter de façon indépendante. Cette indépendance reflète l'objectif des conteneurs : exécuter plusieurs processus et applications séparément les uns des autres afin d'optimiser l'utilisation de votre infrastructure tout en bénéficiant du même niveau de sécurité que celui des systèmes distincts[69].

3. Présentation de l'interface de l'application

Dans cette section, nous allons présenter les différentes interfaces de notre système, avec l'explication des rôles et fonctionnalités de chaque interface, permettant l'exécution des modules de système décrits dans le chapitre précédent, avec des exemples d'application.

- ❖ L'interface « inscrire client » permet à un nouveau client de s'inscrire, pour qu'il puisse se connecter au système et bénéficier des services proposés.

La figure suivante montre l'interface inscrire utilisateur.

The screenshot displays a registration form titled "Inscription" on a light purple background. The form contains the following fields and elements:

- Nom ***: Text input field containing "Aoun Seghir Karim".
- Téléphone ***: Text input field containing "0663582304".
- Adresse Email ***: Text input field containing "karim@gmail.com".
- Mot de passe ***: Password input field with 10 black dots.
- Pays ***: Dropdown menu showing "Algerie" with a close button (x).

Below the fields, there is a legend: *** : Champ obligatoire**.

At the bottom of the form, there are two dark blue buttons: "Inscription" and "Effacer".

At the very bottom, there is a link: "Vous etes inscrit? [Cliquer ici pour Connecter](#)".

Figure 23: Interface inscription client.

- ❖ Après qu'il a inscrit dans le système, le client doit s'authentifier pour connecter à son compte pour pouvoir consommer un service. La figure suivante montre l'interface de connexion d'un client.

Connexion

Adresse Email

Mot de passe

Vous n'etes pas encore inscrit? [Cliquer ici pour Incrire](#)

Figure 24: Interface de connexion d'un utilisateur.

- ❖ Les fournisseurs de services aussi doivent s'inscrire et s'identifier, pour qu'il peuvent publier un service comme il peuvent gérer leurs listes de services publiées.

L'interface suivante c'est l'interface de fournisseur, qui lui permet de publier un service avec tous les champs obligatoires, cette interface lui permet de publier un service avec son fichier de description, et en spécifiant sa fonctionnalité et ces propriétés non fonctionnels.

Nom de Service *	<input type="text" value="myHotel"/>
la fonction de service*	<input type="text" value="Reservation Hotel"/>
Definir les Criteres de QoS	
Coût d'exécution *	<input type="text" value="15000"/>
Temps d'exécution *	<input type="text" value="120"/>
Disponibilité *	<input type="text" value="0.98"/>
Reputation *	<input type="text" value="0.75"/>
Fichier de description	<input type="text" value="C:\Users\ Parcourir..."/>
* : Champ obligatoire	
<input type="button" value="Publier"/> <input type="button" value="Effacer"/>	

Figure 25: Interface fournisseur pour publication de service.

- ❖ La figure suivante montre l'interface qui affiche les services publiés par un fournisseur, où le fournisseur peut gérer la liste publiée par lui-même.



Numero de Service	Le Nom de service	La fonction de Service	Date publication	Action
1	ticket avion	Reservation Avion	07/11/2020 13:38:00	Supprimer <input type="checkbox"/>
2	myPlane	Reservation Avion	07/11/2020 14:04:59	Supprimer <input type="checkbox"/>
15	ReserveHotel	Reservation Hotel	07/11/2020 14:44:09	Supprimer <input type="checkbox"/>
16	Hotel4	Reservation Hotel	07/11/2020 14:44:57	Supprimer <input type="checkbox"/>
17	Mon Hotel	Reservation Hotel	07/11/2020 14:45:56	Supprimer <input type="checkbox"/>
24	myTransport	Reservation Transport	07/11/2020 15:05:27	Supprimer <input type="checkbox"/>
25	visiteTouristTaxi	Reservation Transport	07/11/2020 15:06:50	Supprimer <input type="checkbox"/>
26	visiteTouristTrain	Reservation Transport	07/11/2020 15:12:14	Supprimer <input type="checkbox"/>

Figure 26: Interface de gestion des services publiés.

- ❖ Après la connexion, le client arrive à la page d'accueil ou il y a le menu principal de l'application, la figure suivante montre la page d'accueil.

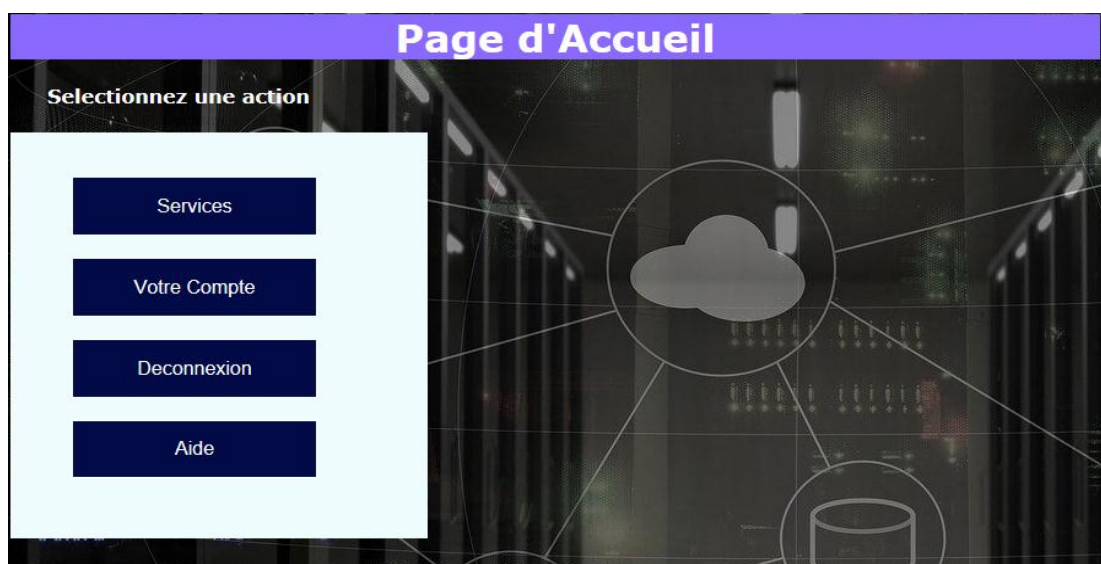
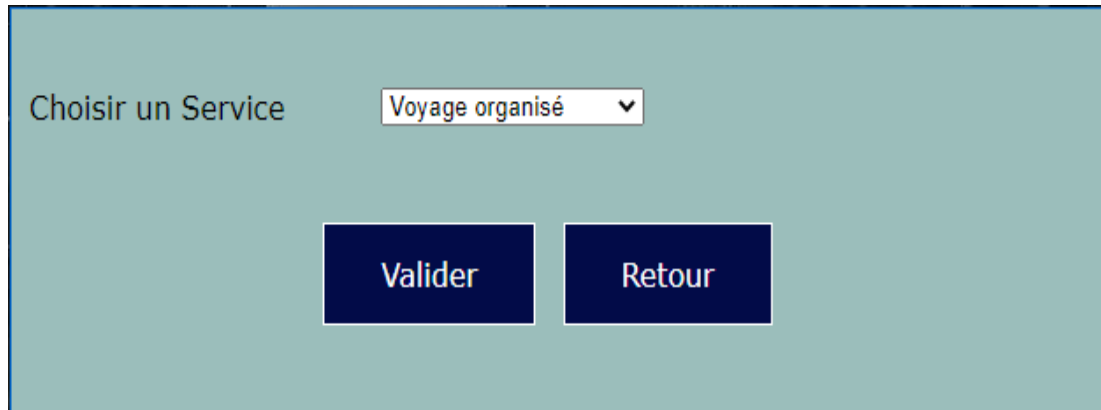


Figure 27: Page d'accueil utilisateur.

- ❖ Le client peut consulter les services disponibles, et demander l'exécution d'un service composé. L'interface suivante permet au client de choisir un service

composé, dans notre exemple le client choisi le service « voyage organisé » qui est un service composé.



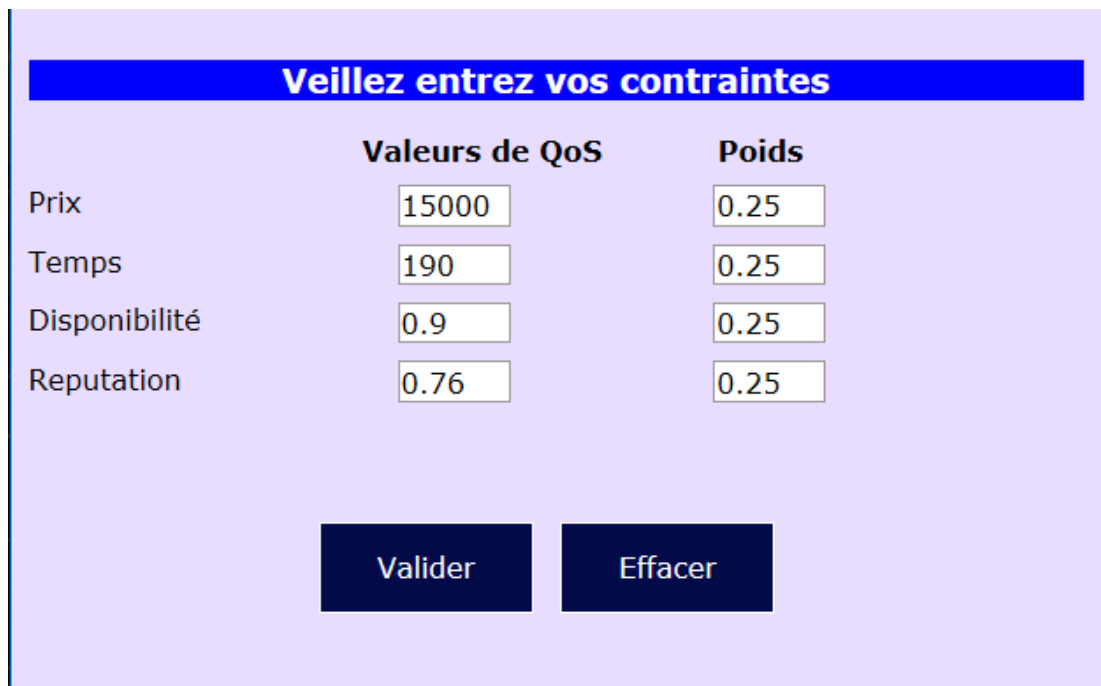
Choisir un Service

Voyage organisé ▼

Valider Retour

Figure 28: Interface de sélection d'un service.

- ❖ Après qu'il clique sur le bouton « Valider », le système affiche l'interface ci dessous, pour permettre au client de saisir ses exigences de QoS, avec les valeurs de poids. Cela permet au système de faire la sélection globale, son but est de sélectionner un service qui satisfait la demande du client.



Veillez entrez vos contraintes

	Valeurs de QoS	Poids
Prix	15000	0.25
Temps	190	0.25
Disponibilité	0.9	0.25
Reputation	0.76	0.25

Valider Effacer

Figure 29: Interface pour la spécification des valeurs de QoS.

- ❖ Après la sélection, le système retourne le plan de composition au client, qui contient la séquence des services avec les fournisseurs de ses services et les valeurs de QoS du service composé, pour donner le choix au utilisateurs d'exécuter le service, la figure suivante montre l'interface qui affiche le service composé, sélectionné après l'opération de sélection.

Service : Voyage organisé

Valeurs de Qualité de service

- Le **Cout** : 11480
- Le **temps** :219
- La **disponibilité** :0.93906945
- La **reputation** :0.7633333333333333

La sequence des services

Numero	Nom Service	Fonction du service	Fournisseur du service
1	ticket Voyage	Reservation Avion	DZCloud
2	ReserveHotel	Reservation Hotel	CloudFire
3	Bus Tourist	Reservation Transport	DZCloud

Valider Annuler

Figure 30: Information sur le service sélectionné.

- ❖ Pour la validation du service le client demande doit cliquer sur le bouton « Valider » pour pouvoir bénéficier du service composé.

4. Simulation et expérimentations

Pour mesurer la performance de notre algorithme de sélection, voici les étapes et les méthodes que nous avons suivies :

- ✓ Nous avons développé un générateur qui génère des descriptions des services en XML, en utilisant la bibliothèque JDOM pour générer les services pour les différents cas de test avec des différentes valeurs de QoS.

Chapitre IV : Expérimentation

- ✓ Nous avons généré aléatoirement les valeurs de QoS pour les services avec les plages des valeurs [2, 15], [20, 1500], [0.95, 1], [0.4, 1] pour le cout, le temps, la disponibilité et la réputation respectivement.
- ✓ Pour les tests, nous avons deux scenarios de simulation :
 - ✚ Un scenario de test qui contient 10 tâches, avec pour chaque cas de test, nous avons varié les services pour chaque tâche en 10, 100, 500 et 1000 services. Le tableau suivant montre les différents paramètres pour les cas de tests de ce scenario.

Cas de test	Taille de population	Nombre des tâches	Nombre des services
1	10	10	10
2	50	10	100
3	200	10	500
4	500	10	1000

Tableau 7: Premier scénario de simulation.

- ✚ Pour le deuxième scenario nous avons fixé le nombre des services pour chaque tâche en 20 et nous avons varié 30,50 et 80 tâches. Le tableau suivant montre les valeurs de paramètres de simulation pour ce scenario.

Cas de test	Taille de population	Nombre des tâches	Nombres des services
1	20	10	10
2	20	30	20
3	20	50	20
4	20	80	20

Tableau 8: Deuxième scénario de simulation.

- ✓ Nous avons exécuté tous les cas de test de chaque scenario, avec 20 itérations pour tous les cas, et pour tous les algorithmes qui nous les avons simulés.

- ✓ L'implémentation est faite avec le langage java, en utilisant java servlet sous une machine Windows 10 professionnel, avec 6 GB de RAM et un processeur Intel i5 d'une cadence de 2.70 GHz, les services sont stockés dans une base de données MySQL.
 - ✓ Nous avons utilisé les valeurs de fitness et le temps d'exécution avec un passage à l'échelle comme mesures pour l'évaluation de notre solution, et pour comparer avec deux autres algorithmes qui sont l'algorithme SFLA proposé dans [24] et l'algorithme GA-S qui est proposé dans [32]. Les mêmes cas de tests sont utilisés pour tester tous les algorithmes.
 - ✓ Les résultats obtenus seront présentés et discutés dans la section résultats et discussions.
- Pour l'exécution de composition, nous avons fait l'exécution du service composé avec un exemple de séquence de deux services :

- ✚ Un service qui calcule la moyenne d'un étudiant, en supposons qu'il y a trois modules. Ce service contient la méthode `calculMoyenne` qui calcule la moyenne.

```
public double calculMoyenne(int t[],double f[])
```

Ce service a deux entrées et une sortie :

- Un tableau d'entiers comme entrée qui contient les coefficients des modules.
 - Un tableau des doubles aussi est une entrée qui contient les notes des modules.
 - Une sortie qui est une valeur double qui représente la moyenne.
- ✚ Un service qui donne l'évaluation à partir de la moyenne obtenue, avec la méthode `réévaluation()`. Par exemple si la moyenne = 14.5, le service affiche que l'étudiant admis et avec mention bien.

```
public String[]getEvaluation(double moy)
```

Ce service a une entrée et une sortie.

- L'entrée est la valeur du moyenne (double).
- La sortie est un tableau de chaînes de caractères(String) qui représente

Pour simuler ce processus nous avons déployé les deux services dans des conteneurs Docker. Pour faire ça, nous avons installé Docker desktop pour Windows.

- ✓ Nous avons installé image pour un serveur web tomcat en utilisant la commande suivante dans l'invite de commande cmd :
`#docker pull tomcat :9.0.39-jdk14`
Cette commande permet d'installer image pour serveur tomcat version 9.0.39 avec une version JDK 14.
- ✓ Nous avons utilisé eclipse Docker tooling, pour permettre la gestion des images et des conteneurs docker à travers eclipse. Pour installer il faut rechercher eclipse docker tooling dans eclipse Marketplace dans la l'item « Help » dans le menu et l'installer dans eclipse.
- ✓ Après l'installation, il faut configurer la connexion vers Docker daemon, en précisant le nom et l'URI de la connexion comme suit :

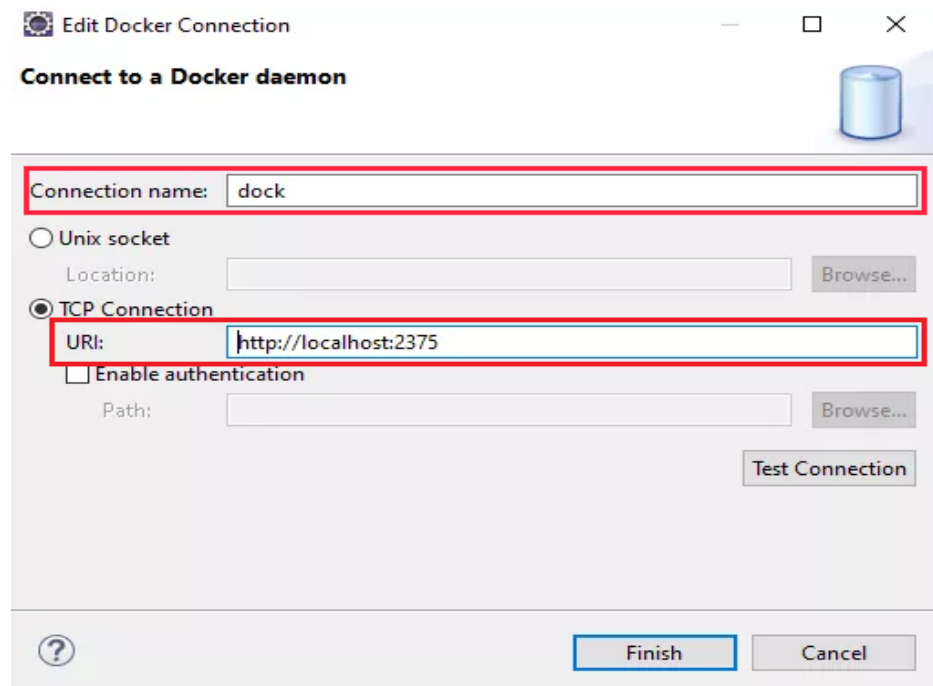


Figure 31: Configuration de la connexion eclipse docker tooling.

- ✓ Pour le déploiement de service dans un conteneur docker, il faut d'abord créer un fichier war pour le service. Le fichier war est un fichier qui contient tous les fichiers exécutables nécessaires pour l'exécution de l'application web, pour permet la portabilité de l'application web et pour permet le déploiement dans n'importe qu'un serveur.
- ✓ Après la création du fichier war, nous avons créé un fichier nommé Dockerfile (doit avoir ce nom), et nous avons ajouté le code illustré ci-dessous :

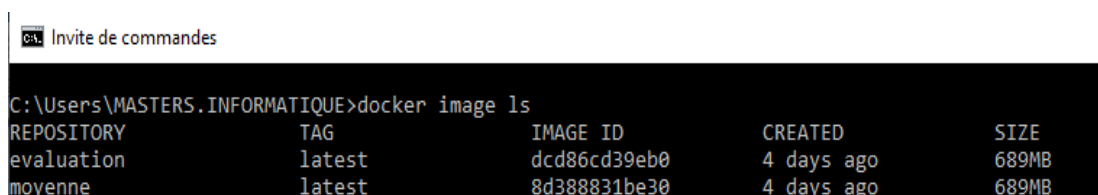
```
1 FROM tomcat:9.0.39-jdk14
2 COPY ./build/calculMoyenne.war /usr/local/tomcat/webapps/
3 CMD ["catalina.sh","run"]
4 |
```

Figure 32: Configuration Dockerfile pour la création de l'image du service.

Ce code permet de créer une image à partir de l'image de tomcat installé, et dans cette image la commande COPY permet le déploiement de fichier war à partir de chemin où il est stocké dans le répertoire webapps dans le serveur tomcat pour être exécuté.

- ✓ L'exécution de fichier avec 'Docker build image' permet la création de l'image en précisant le nom de l'image (cette étape peut être réalisé par la commande #docker build). Avec la commande #docker image ls nous pouvons voir la liste des images que nous avons créé.

La figure suivante montre le résultat d'exécution de la commande #docker image ls qui montre les deux images créées.



```
Invite de commandes
C:\Users\MASTERS.INFORMATIQUE>docker image ls
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
evaluation          latest      dcd86cd39eb0     4 days ago       689MB
moyenne            latest      8d388831be30     4 days ago       689MB
```

Figure 33: Commande 'docker image ls'.

- ✓ La création des conteneurs se fait à partir d'une image, la commande #docker run permet de créer et exécuter le conteneur à partir de l'image identifier par IMAGE ID. Par exemple pour la création d'un conteneur

à partir de l'image évaluation qui a IMAGE ID = dcd86cd39eb0 et accessible via le port 8085 la commande de création sera :

```
#docker run -p 8085 :8080 dcd86cd39eb0
```

- ✓ Pour voir les conteneurs qui sont en cours d'exécution nous utilisons la commande `#docker container ls`, la figure suivante montre le résultat d'exécution de cette commande.

```
C:\Users\MASTERS.INFORMATIQUE>docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
26ac428ee03d   dcd86cd39eb0   "catalina.sh run"       4 days ago    Up 4 days    0.0.0.0:8085->8080/tcp
7361efa421f1   8d388831be30   "catalina.sh run"       4 days ago    Up 4 days    0.0.0.0:8081->8080/tcp
```

Figure 34: Commande 'docker container ls'.

- ✓ Dans notre application, nous avons créé les programmes clients pour les deux services, en utilisant la description WSDL pour les services.
- ✓ Dans une servlet, nous avons exécuté le service compose dans une méthode POST, qu'elle reçoit les données à partir de l'utilisateur, et invoque la séquence, par la création des objets des stubs de clients, en précisant les end points (la localisation du services) pour faire l'appel aux services et récupérer les résultats.

L'exécution du service calculMoyenne, retourne la moyenne qu'elle va être utilisé pour l'exécution du service EvaluationMoyenne, et les résultats d'exécutions seront transmis vers l'utilisateur.

La figure suivante, représente le code du servlet qui contient la méthode POST qu'elle exécute le service composé.

```
package com.pfe.servlets;

import java.io.IOException;

@WebServlet(urlPatterns= {"/Exec"})
public class Composition extends HttpServlet{
    private compositionForm form;
    private EvaluationMoyenneProxy emp;
    private MoyenneProxy mp;

    public void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {
        form=new compositionForm();
        int t[]=form.getCoefs(request);
        double f[]=form.getNotes(request);
        mp=new MoyenneProxy("http://localhost:8081/calculMoyenne/services/Moyenne");
        double moy=mp.calculMoyenne(t, f);
        // System.out.println("Moyenne =" +moy);
        emp=new EvaluationMoyenneProxy("http://localhost:8085/ServiceEvaluation/services/EvaluationMoyenne");
        String str[]=emp.getEvaluation(moy);
        request.setAttribute("moyenne", moy);
        request.setAttribute("decision", str[0]);
        request.setAttribute("mention", str[1]);
        //System.out.println("evaluation =" +str[0]);
        this.getRequestDispatcher("vuesClient/result.jsp").forward(request, response);
    }
}
```

Figure 35: Code de la servlet qui exécute service compose.

- ✓ L'utilisateur saisi les notes des modules avec leurs coefficients à partir de l'interface dans la figure suivante. Supposons qu'il y a trois modules juste pour tester les services et le processus de composition.

	Notes des modules	Coefficients
Module 1	<input type="text" value="14.5"/>	<input type="text" value="2"/>
Module 2	<input type="text" value="13.0"/>	<input type="text" value="2"/>
Module 3	<input type="text" value="15.0"/>	<input type="text" value="1"/>

Figure 36: Interface pour la saisie des notes des modules.

- ✓ Le système exécute le service composé, et il retourne le résultat d'exécution, la figure ci-dessus montre le résultat après l'exécution du service composé.



Figure 37: Résultat après l'exécution du service composé

5. Résultats et discussions

Dans cette section, nous allons présenter et discuter quelques résultats, obtenus lors de la simulation de notre solution proposée pour la sélection dans le processus de composition. Nous avons exécuté chaque cas de test 5 fois, et nous avons pris la moyenne des valeurs obtenues pour chaque cas.

La figure suivante présente les valeurs de fitness par rapport au nombre des services candidats pour chaque tâche dans le premier scénario, où nous avons utilisé une série des services candidats (10,100,500,1000) pour 10 tâches.

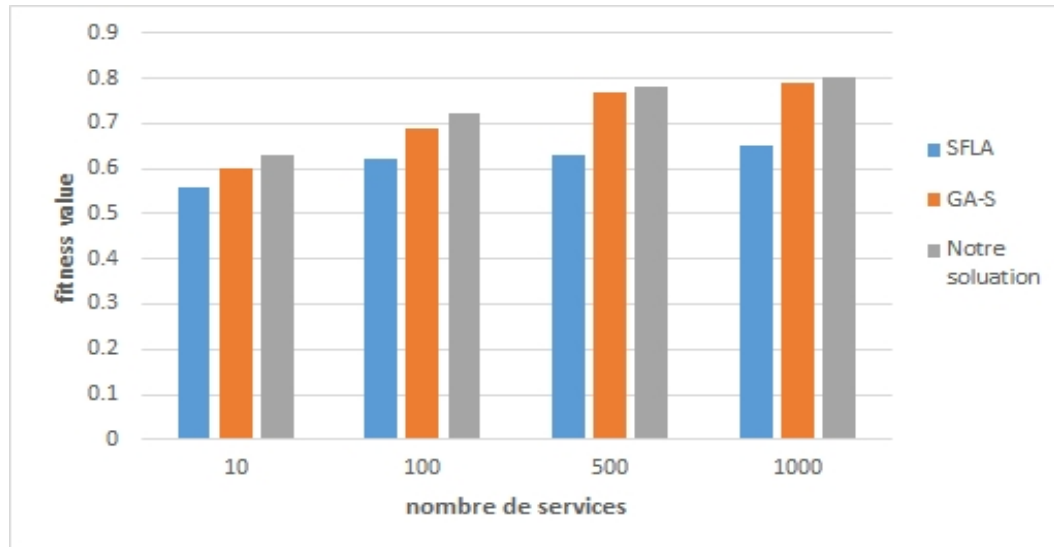


Figure 38: Valeurs de fitness par rapport au nombre des services candidats.

Cette figure, montre que notre algorithme obtient des hautes valeurs de fitness par rapport aux autres algorithmes de simulation, même si les nombres des services candidats augmentent, ça montre que notre algorithme est capable de trouver un meilleur plan de composition qui satisfait les exigences des utilisateurs.

Nous avons mesuré les valeurs du fitness, pour les trois algorithmes dans le deuxième scénario, où nous avons varié le nombre des tâches pour chaque cas de test, avec un nombre des services fixe et égal à 20 services pour chaque tâche.

La figure suivante montre les résultats obtenus de cette expérimentation :

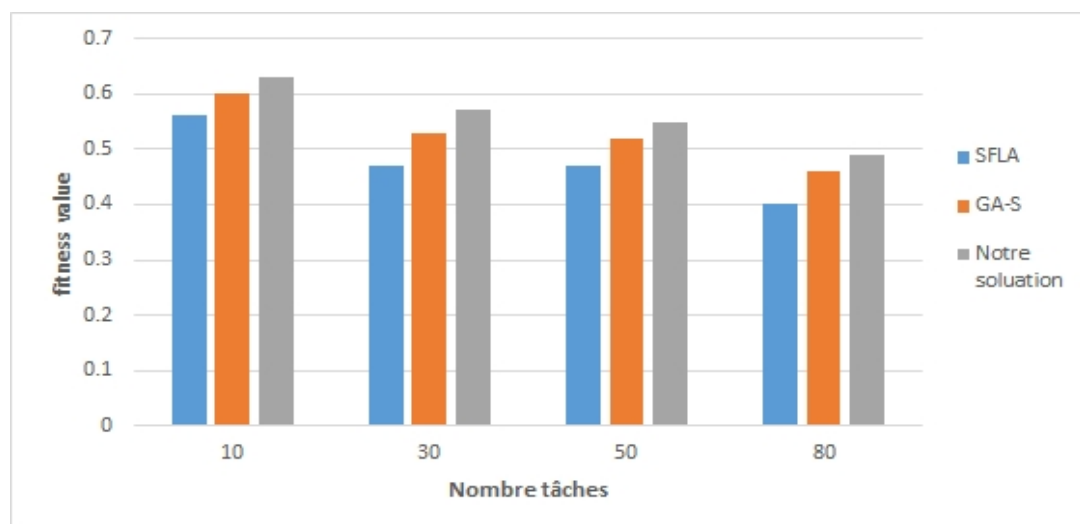


Figure 39: Valeurs de fitness par rapport au nombre de tâches.

Cette figure, montre que toujours notre solution qui obtient les meilleures valeurs du fitness, dans tous les cas de test de ce scénario par rapport aux algorithmes GA-S et SFLA, nous pouvons conclure que notre algorithme peut retourner un plan de composition avec les meilleures valeurs de QoS, quel que soit le nombre de tâches du service composé.

Aussi, nous avons mesuré le temps d'exécution pour les trois algorithmes dans le premier scénario, nous avons testé le temps d'exécution des trois algorithmes après le chargement de tous les données dans la mémoire depuis la génération de la première population jusqu'à la fin d'exécution. La figure suivante présente les résultats obtenus :

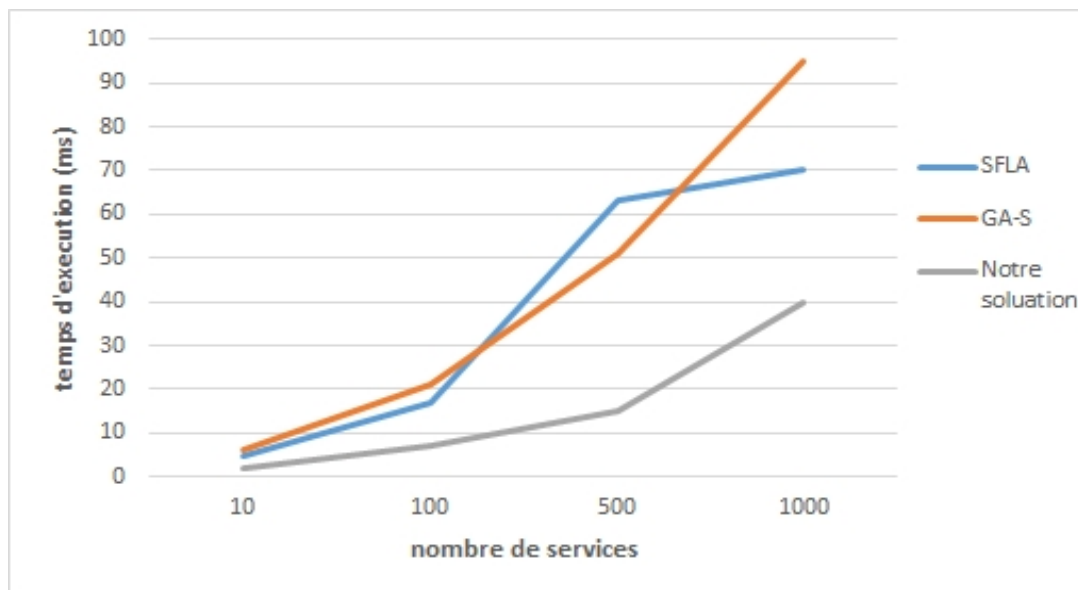


Figure 40: Temps d'exécution par rapport au nombre de services.

Dans tous les cas de test de scénario, notre algorithme s'exécute dans un temps réduit par rapport aux autres algorithmes de cette simulation, cette résultat à cause de réduction des opérations évolutifs dans notre algorithme avec le changement de seulement le mauvais et le bon individu du chaque groupe (deux opérations dans chaque groupe) avec un changement du 40% au maximum dans la population pour chaque itération.

Avec l'utilisation de l'ensemble des service non dominé notre algorithme converge plus rapidement vers la solution qui satisfait la demande du client, ça aussi réduit le temps d'exécution de l'algorithme.

Chapitre IV : Expérimentation

La complexité de notre algorithme est égale $O(N*P)$, avec N représente le nombre des itérations nécessaires pour trouver la meilleure solution, et P la taille de la population, Cette complexité montre aussi que cet algorithme réduit le temps de la sélection surtout avec l'utilisation des services non dominés.

Après les expérimentations et les tests, nous pouvons conclure que notre solution est capable de donner des meilleurs résultats qui satisfait les demandes des clients, les expérimentations montre les meilleures performances pour cette solution.

À la fin nous allons récapituler ces résultats concernant notre solution dans le tableau de l'étude comparative présenté dans le deuxième chapitre.

C1 : approche proposé dans un environnement mono-cloud.

C2 : approche proposé dans un environnement multiple cloud.

C3 : solution tenir en compte les critères de QoS.

C4 : solution à base d'une sélection locale.

C5 : solution à base d'une sélection globale.

C6 : solution avec un algorithme non heuristique

C7 : solution avec un algorithme heuristique.

C8 : solution méta-heuristique.

C9 : solution basée sur les agents

Approche	C1	C2	C3	C4	C5	C6	C7	C8	C9
Notre solution	✓	✓	✓	✗	✓	✗	✗	✓	✗

Tableau 9: Récapitulatif de l'approche proposée.

6. Conclusion

À la fin de ce dernier chapitre de ce mémoire, nous rappelons que nous avons présenté et implémenté notre solution avec les différents modules et nous avons montré les différentes étapes de la composition depuis la requête utilisateur jusqu'au retour de résultat à l'utilisateur.

Aussi nous avons testé et simulé notre proposition de solution du problème de composition et nous avons montré par les expérimentations et les simulations la performance de notre solution et qu'elle est capable de retourner de meilleurs résultats de composition qui satisfont les requêtes d'utilisateurs.

Conclusion générale

Le Cloud Computing offre des milliers de services qui ont différentes fonctionnalités et différentes valeurs de QoS. De nombreux services offrent les mêmes fonctionnalités, mais ils ont différentes valeurs de QoS. D'un autre côté, les demandes des utilisateurs peuvent être complexes et nécessitant plusieurs services.

Les requêtes des utilisateurs peuvent être non satisfaites par un seul service, où il est nécessaire d'avoir une combinaison des services pour les satisfaire. C'est pour ça la composition des services est très importante pour répondre aux demandes complexes. Le processus de composition est un processus compliqué, surtout s'il considère les critères de QoS lors la sélection des services, où ce problème est considéré comme un problème NP-Complet.

Dans ce contexte, nous avons proposé une solution pour ce problème dans les Cloud de type SaaS à base d'une sélection globale, qui prend en compte les critères de QoS. Avant la réalisation de cette solution, nous avons défini des généralités sur le contexte général, qui est le Cloud Computing, et nous avons étudié le problème de composition dans le Cloud Computing et dans l'environnement SaaS, avec la comparaison des différents travaux connexes dans ce contexte. Ça nous a aidés à modéliser et définir notre solution, puis présenté et discuter notre solution après une série des tests et expérimentations.

Notre solution est basée sur un algorithme de sélection globale, et l'idée de cette solution a été prise de l'algorithme SFLA. Elle utilise l'ensemble des services non dominés. Nous avons utilisé Java EE pour tester et simulé cette solution avec deux autres solutions pour les comparer avec notre solution. Les expérimentations montrent que notre solution présente une performance meilleure que l'algorithme SFLA et l'algorithme génétique.

Ce travail ouvre des perspectives pour la satisfaction des requêtes complexes des utilisateurs, qui ne peuvent pas être réalisées grâce à un seul service et des perspectives pour la sélection des services avec les meilleures valeurs de QoS avec pris en considération la latence du réseau pour bien optimiser le temps.

Bibliographie

- [1] Microsoft, “Qu’est-ce que le cloud computing ?” [Online]. Available: <https://azure.microsoft.com/fr-fr/overview/what-is-cloud-computing/>. [Accessed: 10-Dec-2019].
- [2] NIST, “Final Version of NIST Cloud Computing Definition Published,” 2011. [Online]. Available: <https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published>. [Accessed: 10-Dec-2019].
- [3] V. J. R. Winkler, “La Sécurité dans le Cloud,” *Pearson Educ. Fr.*, pp. 1–8, 2011.
- [4] N. Jain and S. Choudhary, “Overview of virtualization in cloud computing,” in *2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016*, 2016.
- [5] P. Mell and T. Grance, “The NIST definition of cloud computing, NIST Special Publication 800,” *Natl. Inst. Stand. Technol.*, 2011.
- [6] A. Fu, “7 Différents types de structures de calcul en nuage que vous devriez savoir,” 2017. [Online]. Available: <https://www.uniprint.net/fr/7-types-cloud-computing-structures/>. [Accessed: 20-Dec-2019].
- [7] A. Jula, E. Sundararajan, and Z. Othman, “Expert Systems with Applications Cloud computing service composition : A systematic literature review,” *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3809–3824, 2014.
- [8] F. Stroud, “What is Anything-as-a-Service (XaaS)?” [Online]. Available: https://www.webopedia.com/TERM/A/anything-as-a-service_xaas.html. [Accessed: 20-Dec-2019].
- [9] J. Dykstra and A. T. Sherman, “Acquiring forensic evidence from infrastructure-as-a-service cloud computing : Exploring and evaluating tools , trust , and techniques,” *Digit. Investig.*, vol. 9, pp. S90–S98, 2012.
- [10] akamai.com, “Architecture du Cloud Computing.” [Online]. Available: <https://www.akamai.com/fr/fr/resources/cloud-computing-architecture.jsp>. [Accessed: 21-Dec-2019].

- [11] Y. Demchenko, "Defining the Big Data Architecture Framework (BDAF)," 2013. [Online]. Available: https://bigdatawg.nist.gov/_uploadfiles/M0055_v1_7606723276.pdf. [Accessed: 10-Dec-2019].
- [12] RightCloud, "What are the jobs related to cloud computing?," 2018. [Online]. Available: <https://www.quora.com/What-are-the-jobs-related-to-cloud-computing>. [Accessed: 21-Dec-2019].
- [13] M. Rouse, "Que signifie Cloud Broker?," 2016. [Online]. Available: <https://www.lemagit.fr/definition/Cloud-Broker>. [Accessed: 21-Dec-2019].
- [14] M. Aazam and E. N. Huh, "Inter-cloud media storage and media cloud architecture for inter-cloud communication," *IEEE Int. Conf. Cloud Comput. CLOUD*, pp. 982–985, 2014.
- [15] Y. Jadeja, "Cloud Computing - Concepts , Architecture and Challenges," pp. 877–880, 2012.
- [16] M. A. Olaru, "Advantages and challenges of adopting cloud computing from an enterprise perspective," *Procedia Technol.*, vol. 12, pp. 529–534, 2014.
- [17] B. K. Rani, B. P. Rani, and A. V. Babu, "Cloud Computing and Inter-Clouds - Types , Topologies and Research Issues .," *Procedia - Procedia Comput. Sci.*, vol. 50, pp. 24–29, 2015.
- [18] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–47, 2014.
- [19] Q. She, X. Wei, G. Nie, and D. Chen, "QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence," *Expert Syst. Appl.*, vol. 138, p. 112804, 2019.
- [20] M. Yuan, Z. Zhou, X. Cai, C. Sun, and W. Gu, "Service composition model and method in cloud manufacturing," *Robot. Comput. Integr. Manuf.*, vol. 61, no. July 2019, p. 101840, 2020.
- [21] C. S. Wu and I. Khoury, "Tree-based search algorithm for web service composition in SaaS," *Proc. 9th Int. Conf. Inf. Technol. ITNG 2012*, pp. 132–138, 2012.

- [22] A. A. Wakrime and S. Jabbour, "Formal approach for QoS-aware cloud service composition," *Proc. - 2017 IEEE 26th Int. Conf. Enabling Technol. Infrastruct. Collab. Enterp. WETICE 2017*, pp. 30–35, 2017.
- [23] S. Pang, Q. Gao, T. Liu, H. He, G. Xu, and K. Liang, "A Behavior Based Trustworthy Service Composition Discovery Approach in Cloud Environment," *IEEE Access*, vol. 7, pp. 56492–56503, 2019.
- [24] A. Younes, M. Essaaidi, and A. El Moussaoui, "SFL Algorithm for QoS-based Cloud Service Composition," *Int. J. Comput. Appl.*, vol. 97, no. 17, pp. 42–49, 2014.
- [25] R. Guerfel, Z. Sbai, and R. Ben Ayed, "On service composition in cloud computing: A survey and an ongoing architecture," in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2015*, vol. 2015-Febru, no. February, pp. 875–880.
- [26] I. El Kassmi and Z. Jarir, "Toward a Smart Cloud Service Composition: Popularity-Driven Approach," *Proc. - 14th Int. Conf. Signal Image Technol. Internet Based Syst. SITIS 2018*, vol. 69, pp. 522–528, 2018.
- [27] R. Khanam, R. R. Kumar, and B. Kumari, "A novel approach for cloud service composition ensuring global QoS constraints optimization," *2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018*, pp. 1695–1701, 2018.
- [28] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Appl. Intell.*, vol. 42, no. 4, pp. 661–678, 2015.
- [29] G. Yi, H. Hu, S. Zhang, and L. Sun, "Cloud Service Composition with Multiple QoS Constraints for Manufacturing Resource," *Proc. - 2018 IEEE 15th Int. Conf. E-bus. Eng. ICEBE 2018*, pp. 158–163, 2018.
- [30] V. Hayyolalam and A. A. Pourhaji Kazem, "A systematic literature review on QoS-aware service composition and selection in cloud environment," *Journal of Network and Computer Applications*, vol. 110. Elsevier Ltd, pp. 52–74, 2018.
- [31] H. Kurdi, A. Al-Anazi, C. Campbell, and A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition," *Comput. Electr. Eng.*, vol. 42, pp.

107–113, 2015.

- [32] D. Wang, Y. Yang, and Z. Mi, “A genetic-based approach to web service composition in geo-distributed cloud environment,” *Comput. Electr. Eng.*, vol. 43, pp. 129–141, 2015.
- [33] G. Zou, Y. Chen, Y. Xiang, R. Huang, and Y. Xu, “AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing,” pp. 28–35, 2010.
- [34] S. B. Bhushan and P. C. H. Reddy, “A hybrid meta-heuristic approach for QoS-aware cloud service composition,” *Int. J. Web Serv. Res.*, vol. 15, no. 2, pp. 1–20, 2018.
- [35] N. Xi, C. Sun, J. Ma, and Y. Shen, “Secure service composition with information flow control in service clouds,” *Futur. Gener. Comput. Syst.*, vol. 49, pp. 142–148, 2015.
- [36] J. O. Gutierrez-Garcia and K. M. Sim, “Agent-based service composition in cloud computing,” *Commun. Comput. Inf. Sci.*, vol. 121 CCIS, pp. 1–10, 2010.
- [37] Z. Z. Liu, D. H. Chu, C. Song, X. Xue, and B. Y. Lu, “Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition,” *Inf. Sci. (Ny)*, vol. 326, pp. 315–333, 2016.
- [38] A. Naseri and N. Jafari Navimipour, “A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm,” *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 5, pp. 1851–1864, 2019.
- [39] A. Singh, D. Juneja, and M. Malhotra, “A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 1, pp. 19–28, 2017.
- [40] Q. Yu, L. Chen, B. Li, and J. Li, “Ant colony optimization applied to web service compositions in cloud computing,” *Comput. Electr. Eng.*, vol. 41, no. C, pp. 18–27, 2015.
- [41] D. Ivanović and M. Carro, “Transforming service compositions into cloud-friendly actor networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8831, pp. 291–305, 2014.
- [42] H. Wang, X. Wang, X. Hu, X. Zhang, and M. Gu, “A multi-agent reinforcement learning approach to dynamic service composition,” *Inf. Sci. (Ny)*, vol. 363, pp. 96–

119, 2016.

- [43] A. Barkat, K. Okba, and S. Bourekkache, "Service composition in the multi cloud environment," *Int. J. Web Inf. Syst.*, vol. 13, no. 4, pp. 471–484, 2017.
- [44] S. Asghari and N. J. Navimipour, "Review and Comparison of Meta-Heuristic Algorithms for Service Composition in Cloud Computing," *Majlesi J. Multimed. Process.*, vol. 4, no. 4, pp. 28–34, 2015.
- [45] Z. Ye, X. Zhou, and A. Bouguettaya, "Genetic algorithm based QoS-aware service compositions in cloud computing," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6588 LNCS, no. PART 2, pp. 321–334.
- [46] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *J. Intell. Manuf.*, vol. 29, no. 8, pp. 1773–1792, 2018.
- [47] H. Kurdi, F. Ezzat, L. Altoaimy, S. H. Ahmed, and K. Youcef-Toumi, "MultiCuckoo: Multi-cloud service composition using a cuckoo-inspired algorithm for the internet of things applications," *IEEE Access*, vol. 6, pp. 56737–56749, 2018.
- [48] H. Mezni and M. Sellami, "Multi-cloud service composition using Formal Concept Analysis," *J. Syst. Softw.*, vol. 134, pp. 138–152, 2017.
- [49] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti, "CSA-WSC: cuckoo search algorithm for web service composition in cloud environments," *Soft Comput.*, vol. 22, no. 24, pp. 8353–8378, 2018.
- [50] L. Liu, S. Gu, M. Zhang, and D. Fu, "A hybrid evolutionary algorithm for inter-cloud service composition," *Proc. 2017 9th Int. Conf. Model. Identif. Control. ICMIC 2017*, vol. 2018-March, no. Icmic, pp. 482–487, 2018.
- [51] S. Bharath Bhushan and C. H. Pradeep Reddy, "A Qos aware cloud service composition algorithm for geo-distributed multi cloud domain," *Int. J. Intell. Eng. Syst.*, vol. 9, no. 4, pp. 147–156, 2016.
- [52] P. Kendrick, T. Baker, Z. Maamar, A. Hussain, R. Buyya, and D. Al-Jumeily, "An Efficient Multi-Cloud Service Composition Using A Distributed Multiagent-based,

- Memory-driven Approach,” *IEEE Trans. Sustain. Comput.*, vol. XXX, no. XXX, pp. 1–1, 2018.
- [53] “An Energy-aware Service Composition Algorithm for Multiple Cloud-based IoT Applications,” pp. 96–108, 2017.
- [54] S. Bharathan, C. Rajendran, and R. P. Sundarraj, “Penalty Based Mathematical Models for Web Service Composition in a Geo-Distributed Cloud Environment,” *Proc. - 2017 IEEE 24th Int. Conf. Web Serv. ICWS 2017*, pp. 886–889, 2017.
- [55] Y. Chen, J. Huang, C. Lin, and X. Shen, “Multi-objective service composition with QoS dependencies,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 537–552, 2019.
- [56] M. F. Zaki Brahmi, “Service Composition in a Multi-Cloud environment based on Cooperative Agents,” in *SRMC 2014: International Workshop on Scheduling and Resource Management in Cloud*, 2014, no. December.
- [57] X. Wang, J. Liu, B. Cao, and M. Tang, “A global optimal service selection approach based on QoS and load-aware in cloud environment,” *Proc. - 2013 IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2013 2013 IEEE Int. Conf. Embed. Ubiquitous Comput. EUC 2013*, pp. 762–768, 2014.
- [58] Y. Q. Samuel, “Software-as-a-Service Composition in Cloud Computing using Genetic Algorithm,” 2018.
- [59] L. Qi, W. Dou, X. Zhang, and J. Chen, “A QoS-aware composition method supporting cross-platform service invocation in cloud environment,” *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1316–1329, 2012.
- [60] H. Hassan, A. El-Desoky, and A. Ibrahim, “An Economic Model for Cloud Service Composition Based on User’s Preferences,” *ICENCO 2017 - 13th Int. Comput. Eng. Conf. Boundless Smart Soc.*, vol. 2018-Janua, pp. 195–201, 2018.
- [61] A. Jula, Z. Othman, and E. Sundararajan, “A Hybrid Imperialist Competitive-Gravitational Attraction Search Algorithm to Optimize Cloud Service Composition,” vol. 3, pp. 37–43, 2013.
- [62] A. S. Kurdija, M. Silic, G. Delac, and K. Vladimir, “Fast Multi-Criteria Service Selection for Multi-User Composite Applications,” *IEEE Trans. Serv. Comput.*, vol.

PP, no. c, p. 1, 2019.

- [63] C. Jatoth, G. R. Gangadharan, and R. Buyya, “Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm,” *Futur. Gener. Comput. Syst.*, vol. 94, pp. 185–198, 2019.
- [64] Y. Li, X. Yao, and M. Liu, “Cloud Manufacturing Service Composition Optimization with Improved Genetic Algorithm,” *Math. Probl. Eng.*, vol. 2019, 2019.
- [65] A. Merizig, “Approche de composition de services web dans le Cloud Computing basée sur la coopération des agents Web service composition approach in Cloud Computing based on agent ’ s cooperation,” pp. 2017–2018, 2018.
- [66] A. Jula, Z. Othman, and E. Sundararajan, “Imperialist competitive algorithm with PROCLUS classifier for service time optimization in cloud computing service composition,” *Expert Syst. Appl.*, vol. 42, no. 1, pp. 135–145, 2015.
- [67] D. Wells, “Extreme Programming,” 2013. [Online]. Available: <http://www.extremeprogramming.org>. [Accessed: 01-Sep-2020].
- [68] A. Tomcat, “Apache Tomcat,” 2019. [Online]. Available: <http://tomcat.apache.org/>. [Accessed: 01-Sep-2020].
- [69] redhat.com, “Docker : définition, fonctionnement et avantages,” 2019. [Online]. Available: <https://www.redhat.com/fr/topics/containers/what-is-docker>. [Accessed: 01-Sep-2020].