République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed KHIDER - BISKRA

Faculté des Sciences exactes et de Sciences de la nature et de la vie

Département d'Informatique

N° d'ordre : .....

N° de série:.....

Thèse

Présentée en vue de l'obtention du diplôme de doctorat en informatique

Option : Intelligence Artificielle et Systèmes Distribués

Présentée par : Mr. BARKAT Abdelbasset **Dirigée par :** Pr. KAZAR Okba

Composition de service web dans le cloud computing

Devant le jury composé de :

Pr. Benmohamed Mohamed	Président	Université de Constantine 2
Pr. Kazar Okba	Rapporteur	Université de Biskra
Dr. Sidhom Sahbi	Examinateur	université de Lorraine France
Dr. Benharzallah Saber	Examinateur	Université de Batna 2
Dr. Rezeg Khaled	Examinateur	Université de Biskra
Dr. Bennoui Hammadi	Examinateur	Université de Biskra
Dr. Ayad Soheyb	Invité	Université de Biskra

Soutenu le : / /2018

Remerciements

Je remercie sincèrement mon directeur de mémoire **Okba KAZAR** Professeur au département d'informatique à l'université de Biskra, et je remercie aussi tous les membres de mon jury de mémoire qui ont pris de leurs temps pour lire et juger ce travail ainsi que pour leur déplacement le jour de la soutenance.

Pour terminer, je remercie ma grande famille pour leur soutien et pour leur aide dans la préparation de ce mémoire et pour leurs encouragements.

Dédicace

Je dédie ce travail à ma mère et ma femme.

ملخص

تعتبر الحوسبة السحابية الخطوة التالية في تطور الإنترنت، وكلمة سحابة "سحابة باللغة الفرنسية" يعني أننا سوف تجمع كل شيء في هذه السحابة والأجهزة والتطبيقات والتخزين وحتى تكوين شبكة وعرض المستخدمين كخدمة لاستخدامها في أي مكان وزمان يريدون. بالنسبة لمستخدمي الحوسبة السحابية، يمكن الوصول إلى كل شيء من خلال خدمات الويب، وتنقسم هذه الخدمات إلى ثلاث فئات يمكن تقديمها في الحوسبة السحابية: البنية التحتية كخدمة (إاس) والمنصة كخدمة (باس) والبرمجيات كخدمة (ادارة العلاقات). وتستخدم هذه الخدمات لأداء المهام الأساسية الموجودة في سحابة واحدة، ولكن في كثير من الأحيان من الضروري وحتى المطلوبة للجمع بين خدمات متعددة تقع في سحابة المؤجلة أو في نفس السحابة ولكن لأداء مهام أكثر تعقيدا.

الهدف من هذا العمل هو اقتراح طريقة لتكوين خدمات الويب في الحوسبة السحابية للسماح للمستخدمين باستخدام العديد من الخدمات في نفس الوقت، مما سيؤدي بالتالي إلى تحسين أداء هذه السحابة.

الكلمات المفتاحية : الحوسبة السحابية, خدمة ويب, تجميع, حالة الخدمة, خوارزم القطرة الذكية

Abstract

Recently, the term cloud computing is widely used in the searching community, which shows the importance given by the scientists to this research area; Cloud computing is a new computing model provides shared resources and data based on service delivery model, where everything from infrastructures, platforms, and software are given to the user like a set of services. The users of cloud platforms deal with these services to satisfy their requests, however these requests become more complex, and they need more than one service to accomplish one request; the process of gathering a set of services to satisfy a user request is called the service composition.

In addition to the fact that one request needs a set of services to be executed, and die to the quick development of cloud technology, there are many of similar services which offer the same functionality, for each service in this set, which make the composition process needs a mechanism to choose between these infinity choices to give the user an optimal satisfaction. We present in this thesis an approach for service composition in the multi cloud environment where compose these services based on the number of cloud bases involved in the composition process.

Keywords: Web service, Cloud computing, Service composition, Intelligent water drops

Résumé

Les travaux de recherche menés autour de la composions de service web dans le cloud computing jusqu'à maintenant, représente une tentative pour résoudre le problème et il n'y pas aucune solution qui est considéré totalement optimale. La difficulté réside dans deux point : Tout d'abord, l'anticipation de tous les services nécessaires peut être un problème très difficile, surtout en cas de service de logiciel, car ils sont connus pour être des services simples et atomiques, publiés par différents éditeurs. Le deuxième défi est la sélection de l'optimal global service composé peut être considéré comme un problème d'optimisation NP-difficile. Dans cette thèse, nous concentrons sur la composition des services automatiques dans l'environnement multi clouds, par conséquent, nous utilisons l'algorithme Intelligent Water Drops, et une technique de programmation linéaire pour décider quelles bases de cloud à sélectionner pour la création du service composé.

Mots clés : Cloud computing, service, composition de service, IWD.

Sommaire

Table des figures

Liste des tables

Liste des abbreviation

NIST	The National Institute of Standards and Technology
SaaS ou SAAS	Software as a service
PaaS ou PAAS	Platform as a service
IaaS ou IAAS	Infrastructure as a service
W3C	Le World Wide Web Consortium
WSDL	Web Service Definition Language
XML	eXtensible Markup Language
SOA	Service Oriented Architecture (Architecture Orienté Service)
IWD	Intelligent Water Drops
PSO	particle swarm optimization
CB	Cloud base
UDDI	Universal Description Discovery and Integration
WSC	Web service composition
PC	personnel computers
RPC	(Remote Procedure Call
TCP/IP	Transmission Control Protocol/Internet Protocol
HTTP	Hyper Text Transfert Protocol

Introduction générale

C'est vrai que le cloud computing est un sujet d'actualité et le terme lui-même est apparu récemment, mais l'idée ou bien son principe n'est pas assez nouveau. Ces origines reviennent à l'année 1961 où *John McCarthy* a introduit le model de partage de ressources virtuel qu'il l'appelle « *Utility computing* ». Puis l'année 1965 ou *Fernando Corbató* et ces collègues ont discutés l'idée de fournir un système d'exploitation par une compagnie qui fonctionne comme les compagnies d'eau ou d'électricité. C'est-à-dire que les clients doivent brancher au réseau de cette compagnie pour bénéficier de ce système. La seule raison qui a empêché le cloud computing d'être une réalité dans cette époque c'est l'absence de bonne technologie pour concrétiser ces principes.

L'avancement technologique dans les dernières années a permis l'apparition des grands équipements comme les Datacenter qui contient des centaines et des milliers de serveurs, une très grande capacité de stockage évolutive. Cette avancée technologique comme les services web et la virtualisation, avec d'autre caractéristiques comme l'accès à la demande est le fondement qui permet de convertir l'idée des années 60 en la réalité dans nos jours.

L'utilisation du cloud computing a incroyablement évolué dans la dernière décennie. Le cloud repose sur plusieurs éléments parmi lesquels, les services web qui sont considérés comme la pierre angulaire pour construire une plate-forme de cloud computing. Généralement le cloud computing est un modèle de livraison de services aux clients, en se basant sur des infrastructures virtuelles. Les utilisateurs font appels aux services web pour exécuter des fonctionnalités sur le cloud. Dans la plupart des cas, les requêtes des utilisateurs ne sont pas réalisables avec un seul service, mais ils sont le résultat de l'exécution de plusieurs services. Pour obtenir ce résultat, les services web doivent être composés en un seul service global appelé le service composé.

A cause de la disponibilité de nombreux services web pour accomplir la même opération, publiés dans un seul ou plusieurs cloud bases, par le même ou déférents éditeurs, le nombre de possibilités pour composer ces services dans un seul service composé va être multiplié. Chaque possibilité est située dans un ensemble de cloud bases. Il est évident que s'il est possible de diminuer le nombre de cloud bases impliqués dans le processus de composition de service, le coût de la communication entre ces services aura une diminution automatique. Pour décider de quel cloud base a été sélectionné dans la composition de services, nous proposons dans cette thèse une première approche basée sur l'algorithme Intelligent Water

Drops (IWD) puis une deuxième approche pour modéliser par la programmation linéaire et résolu par la technique *Branch & bound*, pour l'optimisation du nombre de cloud bases impliqués dans le processus de composition de service

Le rapport de cette thèse est organisé comme suite

- Chapitre 1 : est consacré pour donner en détaille un état de l'art sur les services web, ce chapitre, va nous permet d'étudier et analyser les services web et leur mode de fonctionnement. Les services web sont la pierre angulaire dans notre recherche, parce ce que, ils sont les unités qui vont être composés. Le chapitre est terminé par une présentation de l'architecture orienté service, qui est l'extension logique des services web, car elle représente un environnement où ces services sont exécutés, évolués et même composés.
- Chapitre 2 : ce chapitre intitulé « Emergence de services web dans le cloud computing », dans lequel on va expliquer c'est quoi le cloud computing et quel est sa relation avec les services web.
- Chapitre 3 : On va donner une présentation détaillée suivi par une étude et analyse d'un ensemble des travaux intérieurs qui traitent le même problème, cette étude va nous aider à ne pas proposer une solution déjà existé, et même permettre à comparer nos résultats avec celles des travaux intérieurs.
- Chapitre 4 : est devisé en deux parties, la première partie est consacré pour la présentation détaillé de notre contribution dans ce domaine, par la proposition une nouvelle approche pour résoudre le problème traité dans cette thèse. L'approche est illustrée en utilisant un ensemble de figures et diagrammes, comme l'architecture globale et le diagramme de composants. Au début de ce chapitre on présente quelques algorithmes et méthodes utilisées dans ce travail. Ces algorithmes sont présentés comme un processus standard tel qu'il a été défini par ces auteurs. Dans la deuxième partie, l'ensemble des résultats démontrés pour la validation de nos contributions sont présentés dans ce chapitre. Ces résultats sont commentés, analysés et comparées pour leur donner une bonne interprétation. Nous avons synthétisé ces résultats sous forme de tableaux et de courbes pour faciliter le processus d'analyse et de comparaison.

La fin de cette thèse, une conclusion générale est présenté comme une récapitulation des travaux réalisés, associant la recherche menée et les résultats démontrées durant le cursus

doctorat. Un ensemble de perspectives sont aussi explorées pour conclure la thèse. Ces perspectives seront traitées et étudiées dans l'avenir de nos travaux de recherche.

Chapitre 1

Les services web : état de l'art

1.1. Introduction

Au début de l'informatique les applications dédiées pour être exécuter sur les mainframes sont considérées la meilleure solution pour traiter une large quantité de données. Avec l'apparition de personnel computers (PC), les applications qui fonctionnent avec ces PC's sont devenues très populaire à cause des coûts faibles et la facilité d'utilisation. Avec le très grand nombre des applications de PC qui fonctionnent sur une machine indépendante, la communication entre ces applications est désormais plus en plus complexe. Un autre challenge est rajouté à l'interaction entre application-application.

Ensuite, avec l'apparition des réseaux informatiques et en se basant sur l'invocation des méthodes à distance (Remote Procedure Call ou RPC) sur un protocole de transmission appelé TCP/IP (Transmission Control Protocol/Internet Protocol), les communautés ont prouvé qu'il peut être une solution très acceptable pour la communication entre les applications. A partir de cette étape, les applications qui s'exécutent sur différentes plateformes, différentes systèmes d'exploitation et différents réseaux, font face à quelques défis quand ils ont besoin de communiquer ou de partager des données. Ces besoins ont conduit à la notion d'applications distribuées et le calcul distribué.

1.2. Définition de Services Web

Les services web (en anglais web services) représentent un mécanisme de communication entre les applications distantes à travers les réseaux internet indépendant de tout langage de programmation et de toute plate-forme d'exécution. En utilisant le protocole HTTP (Hyper Text Transfert Protocol) comme un moyen de transport, la communication s'effectue sur un support universel, maîtrisé et généralement non filtré par les pare-feu. En employant une syntaxe basée sur la notation XML pour décrire les appels de fonctions distantes et les données échangées, cela a permis d'organisant les mécanismes d'appel et de réponse.

Grâce aux services web, les applications peuvent être vues comme un ensemble de services métiers, structurés et correctement décrits, communicant selon un standard international plutôt qu'un ensemble d'objets et de méthodes entremêlées [1].

Selon le World Wide Web Consortium (W3C), un service web est conçu pour supporter l'interaction machine à machine interopérables sur un réseau. Il a une interface décrit dans un format compréhensible par une machine. D'autres systèmes interagissent avec le service Web

de la manière prescrite par sa description à l'aide de messages SOAP, généralement transmis à l'aide du protocole HTTP avec une sérialisation XML en conjonction avec d'autres standards Web. [2]

L'idée générale derrière un service web est de fournir une fonctionnalité pour un ensemble de clients. Parce que les services web sont conçu pour être trouvés, ils doivent être publiés des informations concernant leurs descriptions. Le model de service web est composé de ces trois entités : le fournisseur de services, le client et l'annuaire (voir la figure 1 [3]).

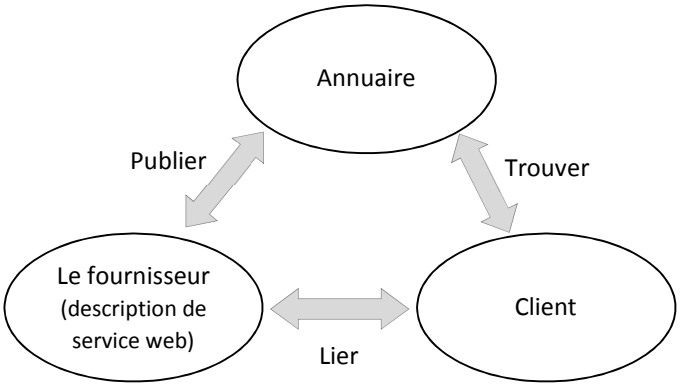

Figure 1 le modèle de service web

La plupart des cas réels, les clients demandent des fonctionnalités complexes qui ne sont pas fournies par un seul service web, cette fonctionnalité est le résultat d'exécution de plusieurs services, alors on est obligé de combiner plusieurs services web pour satisfaire ces demandes, dans ce cas on parle de la composition de services.

1.3 WSDL Web Service Definition Language

La première version de WSDL a été publiée en Septembre 2000 par Microsoft, IBM et Ariba. Brièvement après l'annonce de la spécification UDDI avec 36 autres entreprises, cette version de WSDL a été basée sur deux précédents langages de description : Network Accessible Services Specification Language (NASSL) et (SOAP Contract Language SCL), de IBM et Microsoft respectivement. Plus tard en Mars 2001, les mêmes entreprises avec quelques autres ont soumis la spécification WSDL 1.1 au W3C [4] [5].

L'objectif de WSDL est de décrire un service web quel que soit le langage d'implémentation et la plateforme de déblayement de ce service. Pour que le WSDL soit compréhensible par les différents langages d'implémentation et soit compatible avec les différentes plateformes utilisées, il est totalement écrit en XML.

Alors, un fichier WSDL est un ensemble de balises XML, organisée de telle sorte qu'il décrit un service web de manière indépendante de tout langage ou plateforme. Et comme tous les fichiers xml, un fichier WSDL est une arborescence d'un ensemble des éléments sont : <definition>, <types>, <message>, <portType>, <binding>, < port> et <service>.

L'élément *defintion* représente l'élément racine d'un fichier WSDL, et le reste des éléments sont tous des éléments fils pour cette racine, de telle façon que la forme générale d'un fichier WSDL soit comme suite (voir la figure 2) :

```
<definitions Déclaration de « namespace »…>
        <types>
                Définitions des types….
        </types>
        <message>
                Définitions d'un message….
        </message>
        <portType>
                 <operation>
                        Définitions d'une opération….
                </operation>
        </portType>
        <binding>
                Définitions d'un binding….
        </binding>
        <service>
                Définitions d'un service….
        </service>
</definitions>
```

Figure 2 La structure d'un document WSDL 1

Nous allons expliquer ces différents éléments avec un exemple illustrant un service web implémenté en java et qu'on enrichit le fichier WSDL par un élément à chaque fois.

Pour simplifier l'explication, nous choisissons un service web qui fait l'addition de deux entiers, dont le code est présenté ci-dessous (cf figure 3):

```
package com.calculator;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
/**
 * @author barkat
 */

@WebService
public class Addition {
   @WebMethod
   public int add(@WebParam(name = "par1")int n1,@WebParam(name = "par2") int n2)
   {

      return n1+n2;
   }
}
```

Figure 3 Un service web implémenté en java

Sans entrer dans les détails, la classe java, représente une classe *(Addition)* avec une seule méthode *(add)* qui fait l'addition de deux paramètres (par1 par2), et renvoie le résultat, avec les annotations appropriées pour les services web dans le langage java : *@WebService* , *@WebMethod* et *@WebParam* qui annotent respectivement la classe comme un service web, la méthode comme opération d'un service web et les paramètres comme des paramètres d'une opération dans un service web.

L'élément <definition> : c'est l'élément racine d'un fichier WSDL, contient essentiellement le nom de service web et la déclaration du « *namespaces* » utilisés dans le fichier entier.

L'élément <type> : pour définir tous les types de données utilisées dans le fichier WSDL, le langage WSDL ne spécifie pas aucun type à utiliser malgré que le schéma XML est le type de données par default.

L'élément <portType> : cet élément regroupe la définition abstraite de l'ensemble des opérations offertes par ce service. Chaque opération est la définition abstraite d'une méthode dans l'implémentation de service, pour laquelle élément <operation> spécifie le nom de l'opération ainsi qu'un message d'entrée et un message de sortie pour l'exécution de l'opération.

Dans notre exemple, il y a une seule méthode *add*, alors le fichier WSDL doit contenir un seul élément opération comme suite (cf. figure 4) :

```
<definitions name="additionService">
        <types>
                Définitions des types….
        </types>
        <message> Définitions d'un message….</message>
        <portType>
                 <operation name="add">
        <input wsam:Action="http://my.org/ns/addition/addRequest" message="tns:add"/>
        <output wsam:Action="http://my.org/ns/addition/addResponse" message="tns:addResp
        onse"/>
                </operation>
        </portType>
        <binding> Définitions d'un binding….</binding>
        <service> Définitions d'un service….</service>
</definitions>
```

Figure 4 La structure d'un document WSDL 2

L'élement <message> : Chaque opération désigne deux messages, un pour les entrées et l'autre pour les sorties (les éléments input et output dans la figure 4), dans l'élément <opération> exige de donner juste les noms de deux messages, le reste des informations comme le types, les noms de paramètres doivent décrit dans l'élément message.

Dans notre exemple, il n'y a que deux messages *(add, addresponse)* pour les entrés et les sorties de notre seule méthode add.

```
<definitions name="additionService">
        <types>
                Définitions des types….
        </types>
        <message name="add">
        <part name="parameters" element="tns:add"/>
        </message>
        <message> name="addResponse">
        <part name="parameters" element="tns:addResponse"/>
        </message>

        <portType>
                 <operation name="add">
        <input wsam:Action="http://my.org/ns/addition/addRequest" message="tns:add"/>
        <output wsam:Action="http://my.org/ns/addition/addResponse" message="tns:addResp
        onse"/>
                </operation>
        </portType>
        <binding> Définitions d'un binding….</binding>
        <service> Définitions d'un service….</service>
</definitions>
```

Figure 5 La structure d'un document WSDL 3

Les deux messages spécifient l'élément qui représente les données échangées par ce message. L'élément *tns :add* et *tns :addResponse* pour les messages *add* (message d'entrée) et *addResponse* (message de sortie) respectivement. Ces deux éléments sont des éléments de type complexe, dont la définition détaillée se trouve dans le schéma *tns (figure 5).*

L'élement <binding> : contient une définition concrète de types de données et les protocoles pour un ensemble d'opérations et de messages définis dans un porttype.

1.4. L'annuaire UDDI

Après la description du service web, les clients ne pouvant pas l'utiliser parce qu'ils ignorent l'existence de ce service. C'est pour cela on a besoin d'un moyen qui sert comme une publicité qui annonce la création d'un nouveau produit. Les chercheurs ont défini un annuaire appelé UDDI (Universal Description, Discovery, and Integration) qui sert comme un ensemble de normes pour la publication et la localisation de services web.

UDDI a deux parties : un registre de toutes les métadonnées d'un service Web (y compris un pointeur vers la description WSDL d'un service), et un ensemble de définitions de type de

port WSDL pour la manipulation et la recherche de ce registre. La dernière spécification UDDI est la version 2.0. UDDI n'est pas la seule option pour la découverte de services. IBM et Microsoft ont récemment annoncé le langage d'inspection des services Web (WS-Inspection), un langage basé en XML qui fournit un index de tous les services Web disponibles sur le Web [6].

Modèle de données UDDI

Les informations d'enregistrement UDDI comprennent les cinq types de structure de données suivants [7] :

- *businessEntity*, la structure de niveau supérieur, décrivant l'entreprise ou autre entité pour laquelle les informations sont enregistrées. Les autres structures sont reliées par des références de cette structure.
- *businessService*, le nom et la description du service en cours de publication.
- *bindingTemplate*, informations sur le service, y compris une adresse du point d'entrée pour accéder au service.
- *tModel*, une empreinte digitale ou une collection d'informations identifiant de manière unique le service. Cette structure de données prend également en charge les recherches de niveau supérieur.
- *publisherAssertion*, une structure relationnelle mettant en association deux ou plus de structures businessEntity selon un type spécifique de relation, tel en tant que filiale ou département de.

La figure 6 regroupe ces cinq éléments et montre leurs relations avec la description de service web en WSDL [8].

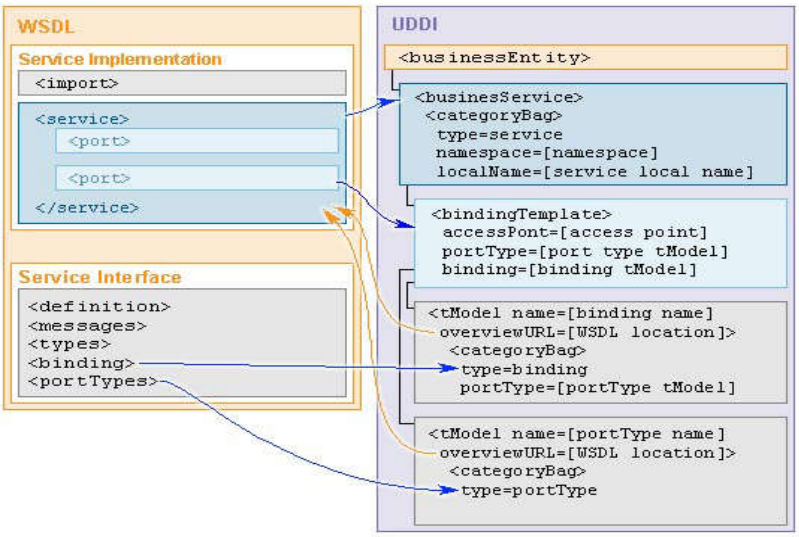

Figure 6 WSDL vers UDDI

1.5. SOAP - Simple Object Access Protocol

SOAP est un protocole de transmission de messages basé sur l'XM. Il définit un ensemble de règles pour structurer des messages principalement pour exécuter des dialogues requête-réponse de type RPC (ou Remote Procedure Call). Un message SOAP respecte les règles de XML et il est composé de trois éléments essentiels : l'enveloppe, header et body du message (cf. figure 7).

SOAP est un protocole basé sur XML pour l'échange d'informations entre ordinateurs. Bien que SOAP puisse être utilisé dans divers systèmes de messagerie, il peut être fourni via un variété de protocoles de transport. L'objectif initial de SOAP est de permettre des appels de procédure à distance transporté via HTTP. SOAP permet donc aux applications clients de se connecter facilement aux services distants et invoquer des méthodes distantes. Par exemple, un application cliente peut immédiatement ajouter la traduction de la langue à son ensemble de fonctionnalités par la localisation et ensuite l'invocation de la correcte service SOAP [9].

Malgré que les avantages de SOAP qui sont nombreux, on cite principalement quelques exemples qui sont considérés importants comme :

- Protocole de communication entre applications
- Basé sur XML et les namespaces;
- Communication par le Web (HTTP / SMTP / ...) ;
- Indépendant de la plateforme (windows, unix, mac, ...) ;

- Simple et extensible.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <ns2:Add xmlns:ns2="http://ServicesPackage/">
         <entier1>1</entier1>
         <entier2>2</entier2>
      </ns2:Add>
   </S:Body>
</S:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:AddResponse xmlns:ns2="http://ServicesPackage/">
         <return>3</return>
      </ns2:AddResponse>
   </S:Body>
</S:Envelope>
```

Figure 7 Exemple pour un requête et une réponse Messages SOAP

1.6. Les caractéristiques de services web

Les services web, en tant que des entités technologiques, ont un ensemble de caractéristiques. On cite dans les prochains paragraphes les caractéristiques les plus importantes qui figurent dans les littératures

1.6.1 Types de services Web

Sur le plan topologique, les services Web peuvent prendre deux formes (cf. figure 8). Services Web Informatif ou type I, qui ne prennent en charge que les opérations simples de requête / réponse et d'attendre une demande. Les services Web complexes ou de type II, implémentent une certaine forme de coordination entre les opérations entrantes et sortantes. Chacun de ces deux modèles présentent plusieurs caractéristiques importantes et sont à leur tour subdivisés en plusieurs sous-catégories spécialisées [10].

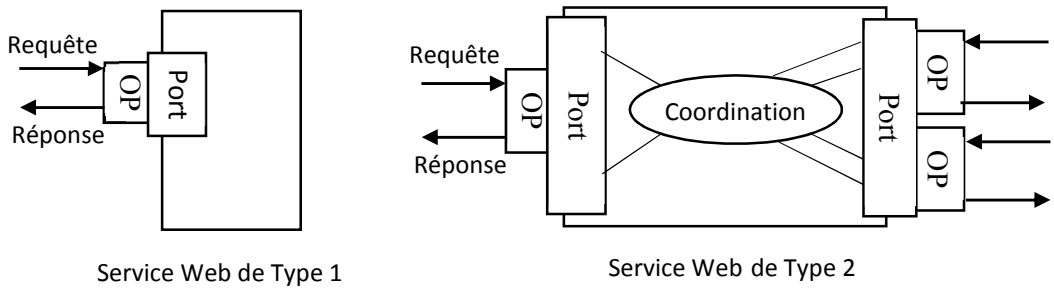

Figure 8 Types de service web

Donc selon le type d'opération traiter par le service web, ce dernier est devisé en deux catégories : les services Web informatif si l'opération est simple, et les services web complexes si elle ne l'est pas.

1.6.2. Aspects fonctionnel et non fonctionnel

Les services web sont décrits en termes de langage de description. Une description de service a deux principaux composants : ses caractéristiques fonctionnelles et non fonctionnelles. La description fonctionnelle détaille les caractéristiques opérationnelles qui définissent le comportement global du service, c'est-à-dire, définit les détails de la manière dont le service est invoqué, l'emplacement où il se trouve, etc. Cette description se concentre sur les détails concernant la syntaxe des messages et comment configurer les protocoles du réseau pour livrer des messages. La description non fonctionnelle se concentre sur les attributs de qualité de services, tels que la métrique de service et de coût, les mesures de performance (le temps de réponse, la précision, les attributs de sécurité, l'autorisation, authentification, intégrité (transactionnelle), fiabilité, évolutivité et disponibilité). [10].

Les attributs de qualité de services jouent un rôle très important dans la composition de services web.

1.6.3. Propriétés de l'État de services web

Les services web peuvent être avec ou sans état. Si les services peuvent être invoqués à plusieurs reprises sans avoir à maintenir le contexte ou l'état, ils sont appelés sans état. A l'opposé, les services qui exigent à préserver leur contexte d'une invocation à l'autre sont appelés avec état. Le protocole d'accès aux services est toujours sans connexion, c'est-à-dire que le protocole n'a pas le concept de session et ne fait aucune hypothèse à propos de la livraison éventuelle [10].

1.6.4. Faible couplage

Les services Web interagissent les uns avec les autres de manière dynamique et utilisent des technologies standards de l'Internet. Ce qui permet de créer des ponts entre les systèmes qui nécessiteraient autrement un grand effort de développement. Le terme couplage indique le degré de dépendance que deux systèmes ont l'un sur l'autre [10].

1.6.5. Granularité du service

Les services Web peuvent varier en terme de fonction, des demandes simples ou complexes pour accéder et combiner des informations provenant de plusieurs sources. Malgré que les services simples peuvent avoir une réalisations complexes. Les services simples sont de nature discrète, présentent normalement un mode de fonctionnement de demande/réponse, et sont de granularité fine, c'est-à-dire qu'ils sont de nature atomique. En revanche, les services complexes n'ont pas une granularité fine parce que l'exécution de ces services impliquent des interactions avec d'autres services et éventuellement des autres utilisateurs en un seul ou plusieurs sessions [10].

1.6.6. La Synchronicité

Nous pouvons distinguer deux styles de programmation pour les services : le style synchrone comme l'appel de procédure à distance (RPC), et le style asynchrone comme le passage de message (ou document).

-Pour les services synchrones : Les clients des services synchrones expriment leur demande en tant qu'appel de méthode avec un ensemble d'arguments, qui retourne une réponse contenant une valeur de retour. Cela implique que lorsqu'un client envoie un message de demande, il s'attend à un message de réponse avant de poursuivre son calcul.

-Pour les services asynchrones : les services asynchrones sont des services de type document ou messagerie. Lorsqu'un client appelle un service de type message, le client envoie un document entier (tel qu'un bon de commande) plutôt qu'un ensemble de paramètres. Le service accepte le document, le traite et peut renvoyer ou ne pas renvoyer un message de résultat. Un client qui appelle un service asynchrone n'a pas besoin d'attendre une réponse avant de continuer avec le reste de son application. S'il y a une réponse pour ce service, il peut apparaître dans les heures ou même dans les jours plus tard [10].

Le principe de synchronicité est très connu avec les anciens systèmes. Un système peut fonctionner en mode synchrone comme le système téléphonique ou bien en mode asynchrone comme par exemple une messagerie d'é-mail.

1.6.7. Un service bien défini

L'interaction de services doit être bien définie. WSDL permet aux applications de décrire pour d'autres applications les règles d'interfaçage et d'interaction. Il fournit aussi un mécanisme uniforme pour décrire les interfaces abstraites de services et les spécificités de protocole de liaisons qui supporte le service. Les descriptions de services focalisent sur la façon dont les opérations interagissent avec un service : comment les messages invoquent des opérations ? les détails de construire de tels messages, et des détails sur l'endroit où envoyer des messages pour traitement, c'est-à-dire, déterminer les points d'accès au service [10].

1.6.8. Contexte d'utilisation du service

En plus des types et caractéristiques des services Web mentionnés dans les paragraphes précédents, il est également utile de diviser les services en différentes catégories en fonction de la perspective du demandeur du service Web. Ici, la distinction est faite entre les services remplaçables et les services critiques.

-Un service Web remplaçable est un service fourni par plusieurs fournisseurs et l'action de remplacer un fournisseur avec un autre n'affecte pas la fonctionnalité de l'application tant que les interfaces de services sont identiques [10].

-Un service Web critique est un service éventuellement fourni par un seul fournisseur, et l'action de remplacer ce service compromet gravement la fonctionnalité d'une application entière. Si le service est indisponible pendant un certain temps, cela réduirait considérablement la productivité de l'application. Ce type de service aurait typiquement certaines données commerciales et être intégré au niveau du processus.[10].

Cette caractéristique est très importante dans la composition de services, parce que le processus de composition est réalisé selon les perspectives du client, et le fait qu'il y un service remplaçable pose un problème de décision ou bien de choix entre plusieurs services qui fournit la même fonctionnalité.

1.7. Les attributs de qualité de service (OoS)

La qualité de service couvre toute une gamme de techniques qui correspondent aux besoins des demandeurs de service avec ceux des fournisseurs de services en fonction des ressources réseau disponibles. Par QoS, nous nous référons aux propriétés non-fonctionnelles des services Web telles que les performances, la fiabilité, la disponibilité et la sécurité [11].

La qualité de service est un ensemble de propriétés opérationnelles du service que l'on doit constater dans la réalisation de la prestation. Formalisées dans le contrat, ces propriétés représentent un ensemble d'exigences concernant la mise en œuvre du service et constituent donc un engagement de niveau de service (Service Level Agreement ou SLA). Ces propriétés peuvent être réparties en six groupes [12]:

- Les propriétés relatives au périmètre de prestation comme la limite de prestation et les droits et obligations du client ;
- Les propriétés relatives au qualité de fonctionnement qui touche le service quand il est en état de fonctionnement comme la performance et l'accessibilité ;
- Les propriétés relatives au sécurité de service comme l'authentification, l'autorisation, et la confidentialité ;
- Les propriétés relatives au robustesse de service qui définissent le comportement de service face aux défaillances comme la fiabilité et la disponibilité ;
- Les propriétés relatives à la gestion de service et la gestion de changement.

Trois aspects différents doivent être considérés en parlant de services web : (1) fonctionnel, (2) comportement et (3) non fonctionnel. La description fonctionnelle contient la spécification formelle de ce que le service peut faire exactement. La description du comportement porte sur la façon dont la fonctionnalité du service peut être atteint en termes d'interaction avec le service, et aussi en termes de fonctionnalités requises par des autres services Web. Enfin, les descriptions non fonctionnelles capturent les contraintes sur les deux aspects précédents [13].

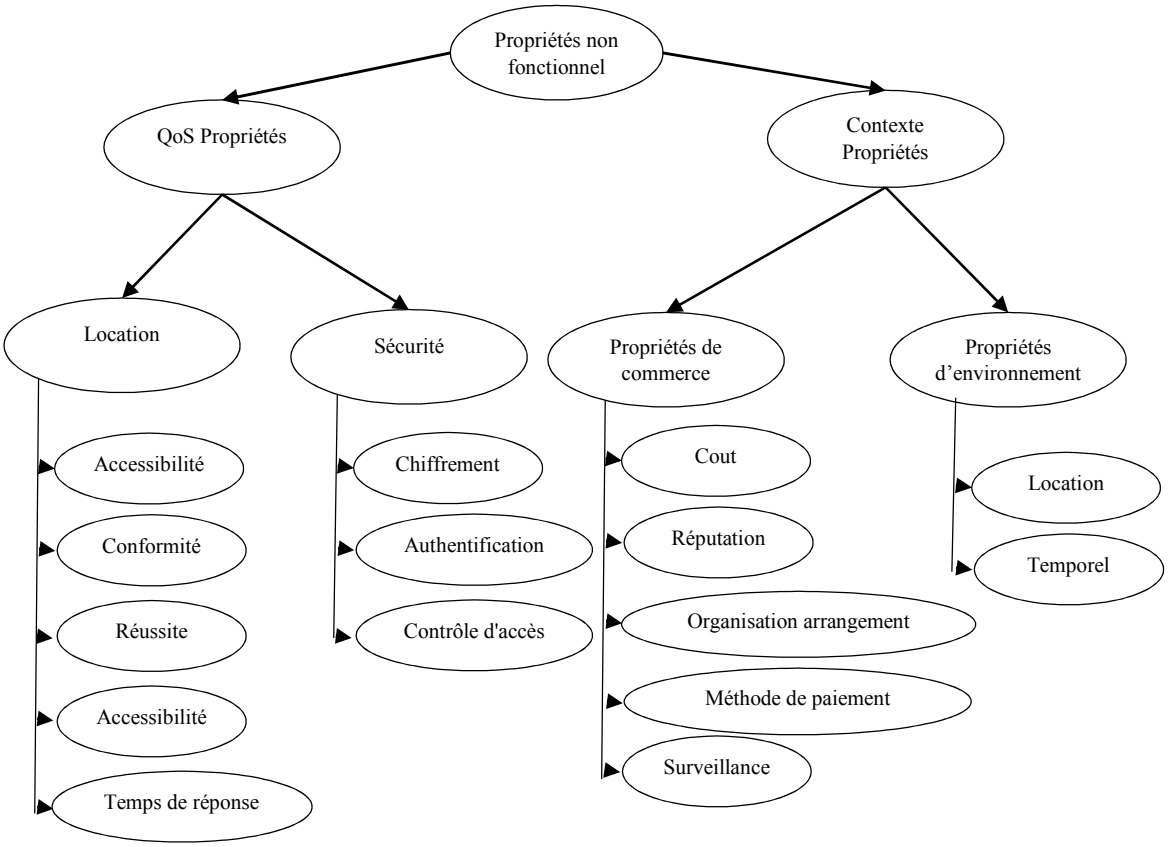

Figure 9 Les propriétés non fonctionnel de service web

Un autre avis discuté par Youakim Badr et ces collègues [14], expliquent qu'un service Web peut être entièrement décrit par deux ensembles de propriétés : (i) fonctionnelle et (ii) non fonctionnelle [14], les propriétés non fonctionnelles peuvent être deviser en deux catégories, la première liée au QoS et la deuxième liée au contexte du service (cf. figure 9).

Dans ce qui suit, le tableau 1 regroupe la définition de l'ensemble des propriétés non fonctionnelles présentées par l'étude de la référence [14]

Tableau 1 Les propriétés non fonctionnelles de services web

QoS Propriétés	Location	Temps de réponse	le temps écoulé depuis la soumission d'une demande au moment où le la réponse est reçue
		Accessibilité	représente le degré que Le service Web est en mesure de répondre à une demande
		Conformité	Représente la mesure dans laquelle Le document WSDL suit WSDL spécification
		La réussite	Représente le nombre de demander des messages qui ont été répondus.
		Disponibilité	Représente le pourcentage de temps que fonctionne un service
	Sécurité	Chiffrement	La capacité d'un service Web à maintenir le cryptage des messages
		Authentification	la capacité d'un Web service pour offrir des mécanismes sert à l'identification de la partie qui a invoqué le service
		Contrôle d'accès	Si le service Web fournit des moyens de contrôle d'accès pour limiter l'invocation de l'opération et l'accès à information aux partis autorisés
Contexte Propriétés	Propriétés de commerce	Cout	Représente l'argent qu'un consommateur d'un service Web doit payer pour l'utiliser.
		Réputation	mesure la réputation de Services Web basés sur les commentaires des utilisateurs.
		Organisation arrangement	Inclut les préférences et l'historique (en cours partenariats)
		Méthode de paiement	Représente les méthodes de paiement acceptées par un service Web, c'est-à-dire banque de transfert, carte Visa etc.
		Surveillance	nécessaire pour un certain nombre d'objectifs, y compris l'optimisation des performances, vérification de l'état, débogage et dépannage.
	Propriétés d'environnement	Location	L'emplacement de service web.
		Temporel	Est-ce que le service est permanant ou temporel.

Avant de terminer la première partie qui examine sur les services web, on mentionne qu'il y a deux stratégie pour la création de services web. Soit on commence de coder le service dans un langage de programmation (java, c++…) ensuite on génère le fichier WSDL. On appelle cette stratégie *Buttom up*. Ou bien on commence par la création de fichier WSDL et on génère le code à partir de ce fichier. C'est la stratégie *Top down*.

Les services web ne sont pas la seule solution pour développer des systèmes distribués, mais il y a des raisons pour lesquels ils sont considérés meilleurs que les autres solutions :

- Les services web sont une technologie client/serveur en architecture distribuée (comme CORBA, RMI, EJB, …) ;
- Plus précisément, un service web représente la partie serveur, destinée à fournir des services (traitements) à des clients distants ;

- La particularité forte des services web est d'utiliser le langage XML pour l'échange d'informations entre les clients et les serveurs ;
- Tout type de langage (Java, C#, PHP, etc…) permet le développement de services web, aussi bien côté serveur que côté client.

1.8. Architecture orientée service

Une Architecture orienté service (Service Oriented Architecture ou SOA en anglais) est un Framework stratégique de technologie qui permet à tous les systèmes intéressés, à l'intérieur et à l'extérieur d'une organisation, d'exposer et d'accéder à des services bien définis, et à des informations liées à ces services, qui peuvent être encore abstraites pour traiter les couches et les applications composées pour le développement de solutions. En substance, SOA ajoute l'aspect d'agilité à l'architecture, ce qui nous permet de faire face aux changements dans les systèmes en utilisant une couche de configuration plutôt que d'avoir constamment à redévelopper ces systèmes [15].

Un service**,** au sens SOA, met à disposition aux acteurs (humains ou logiciels) intervenants dans des processus métiers, un accès vers une ou plusieurs fonctions métiers [16].

Une architecture orientée services est une architecture logicielle s'appuyant sur un ensemble de services simples. L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'applications, pour faire en sorte de construire une architecture logicielle globale décomposées en services correspondant aux processus métiers de l'entreprise.

L'architecture orientée services (Service-Oriented Architecture) est une approche du développement logiciel qui décompose les applications opérationnelles en fonctions distinctes ou "services" – par exemple, vérifier la solvabilité, créer un compte, etc. – utilisables indépendamment des applications et des plateformes qui les exécutent. Dès lors que les fonctions individuelles des applications sont ainsi disponibles sous la forme de composants quasi invisibles, l'entreprise a la faculté de les intégrer et de les regrouper différemment pour créer des fonctionnalités entièrement nouvelles (cf. figure 10).

1.8.1. Modèles et architectures de type SOA

Plusieurs modèles et architectures sont proposés par les chercheurs pour capturer et illustrer au maximum les mécanismes de fonctionnement d'un SOA. Parmi eux, on présente ici un modèle proposé par des chercheurs de l'entreprise IBM [17].

Ce modèle est basé sur un style architecturel qui définit l'interaction entre les clients, les fournisseurs, et l'intermédiaire qui fournit et maintint l'annuaire des services. Une architecture SOA est une architecture informatique à l'échelle de l'entreprise permettant de relier des ressources à la demande. Elle consiste en un ensemble de services informatiques alignés sur les activités qui remplissent collectivement les processus et les objectifs d'une organisation. Il est possible de chorégraphier ces services dans des applications composites et de les invoquer par des protocoles standards. Un service est une ressource logicielle découvrable grâce à une description de services. Cette description est disponible pour la recherche, la liaison et l'invocation par un consommateur de service. Le fournisseur implémente la description de services et fournit également les exigences de qualité de service au consommateur.

Une vue abstraite de SOA se représente comme une architecture en couches de service composés qui correspondent aux processus métier. La figure 10 illustre cette représentation pour ce type d'architecture.

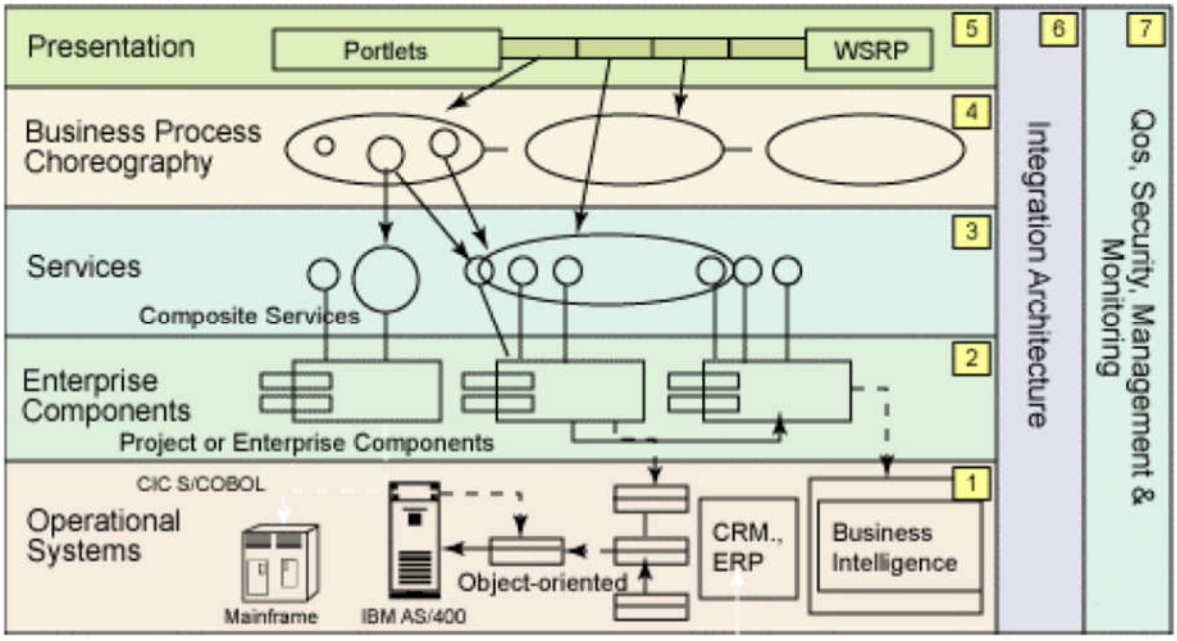

Figure 10 Architecture Orienté Services

1.8.2. La chorégraphie de services

La chorégraphie de services est une description globale des services participants, qui est définie par l'échange de messages, de règles d'interaction et d'accords entre deux points de terminaison ou plus. La chorégraphie utilise une approche décentralisée pour la composition de services, c'est-à-dire qu'il n'y pas un contrôleur central, et chaque service qui participe à la chorégraphie doit avoir un comportement autonome.

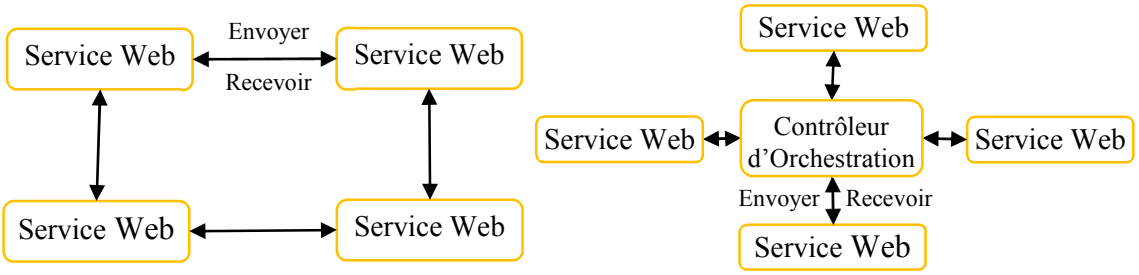

Figure 11 Composition de services web

Au moment de la conception, la chorégraphie assure l'interopérabilité entre un ensemble de services d'un point de vue global, ce qui signifie que tous les services participants sont traités de manière égale, d'une manière *peer-to-peer*. Un modèle de chorégraphie décrit une collaboration multipartite et met l'accent sur l'échange de messages. Chaque service Web impliqué dans une chorégraphie sait exactement quand il doit exécuter ses opérations et avec qui interagir (cf. figure 11) [18].

1.8.3. L'orchestration de services

Pour les solutions basées web, la logique métier doit être distribué dans différents services. Ces services doivent être organisés et composés, qui est communément appelé comme une orchestration. Un processus d'orchestration présente différents services qui peuvent être composés efficacement à travers un flux afin d'exécuter un processus métier. Les différents services impliqués sont coordonnés grâce à un service de contrôleur. Le service résultant peut-être intégré hiérarchiquement dans une autre composition en utilisant son interface WSDL[18].

Dans l'orchestration, les services web concernés sont sous le contrôle d'un seul processus central (un autre service Web). Ce processus coordonne l'exécution des différentes opérations du services Web qui participent au processus (cf. figure 11).

1.9. Conclusion

Nous avons survolé, dans ce chapitre le domaine de services web, qui sont considérés comme la dernière technologie de type client-serveur. Les services web sont utilisés dans plusieurs domaines informatique, pour améliorer des solutions ou bien pour fonder des nouvelles technologies, comme par exemple le cloud computing, qui est un nouveau paradigme pour organiser la relation entre le client et le fournisseur. Dans le prochain chapitre, on va parler sur le cloud computing pour comprendre la relation entre ce concept et les services web.

Chapitre 2

Emergence de services web dans le cloud computing

2.1. Introduction

Si vous voulez cherchez l'origine du terme *cloud computing*, vous allez trouver plusieurs allégations, entre Amazon.com, Google, passant par Microsoft. La plupart de ces compagnies se considèrent comme le principal acteur dans ce domaine de technologie, Mais toutes les compagnies sont d'accord que le cloud est devenu un *buzz Word* à partir des années 2000. Depuis cette date, les chercheurs ont essayé de définir et standardiser les concepts en relation avec le cloud computing.

Dans la suite du chapitre on va explorer le concept du cloud computing et tous ce qui en relation avec ce domaine.

2.2. Définition du cloud computing

Il n'existe pas une définition standard pour le cloud computing, chaque chercheur ou groupe de recherche propose sa propre définition. Pour cette raison, Vaquero et ces collègues[19] ont essayé de construire une définition complète pour le cloud computing en étudions et analysons 22 définitions qui existent dans la littérature. A la fin de ce travail, ils ont proposé la définition suivante ;

Les clouds sont une grande réserve de ressources virtuelles faciles à accéder et à utiliser (comme les matériels, les plateformes de développement). Ces ressources peuvent être reconfigurer dynamiquement pour s'adapter à des situations variables, permettant aussi une utilisation optimale de ces ressources. Typiquement cette réserve de ressources est exploitée par un modèle de paiement à l'utilisation dont laquelle des garanties sont offertes par le fournisseur de l'infrastructure.

A partir de l'année 2009, le cloud computing et sa communauté ont trouvé un consensus total sur la définition par l' Institut national des normes et de la technologie (National Institute of Standards and Technology ou NIST), qui réclame que :

Le cloud computing est un modèle Informatique qui permet un accès facile et à la demande par le réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service [20].

Pour la raison citée ci-dessus, nous adoptons dans ce mémoire la définition de NIST pour le cloud computing. C'est pour cela nous allons étudier en détail cette définition dans les paragraphes suivants.

Le modèle de Cloud Computing définit par le NIST est composé de cinq caractéristiques essentielles, trois modèles de services (modèles de livraison) et quatre modèles de déploiement [20].

2.3. Emergence du cloud computing

Le cloud Computing est le résultat d'un avancement technologique de modèle de calcul. Il est considéré comme la sixième étape de développement de ce modèle[21]. Après l'ancien mainframe dont lequel plusieurs utilisateurs sont connectés à une grande machine appelée mainframe via un terminal. Ensuite l'apparition de PC qui est un petit ordinateur mais assez puissant pour satisfaire les besoins des utilisateurs. Ces PCs sont connectés entre eux et aussi avec un autre type spécialisé d'ordinateurs appelés serveurs pour créer un réseau. Consécutivement, les réseaux sont liés pour construire un réseau mondiale appelée l'Internet. Sur cette base technologique, le model de calcul grid computing et le model de calcul Cloud Computing sont apparues (cf. figure 12).

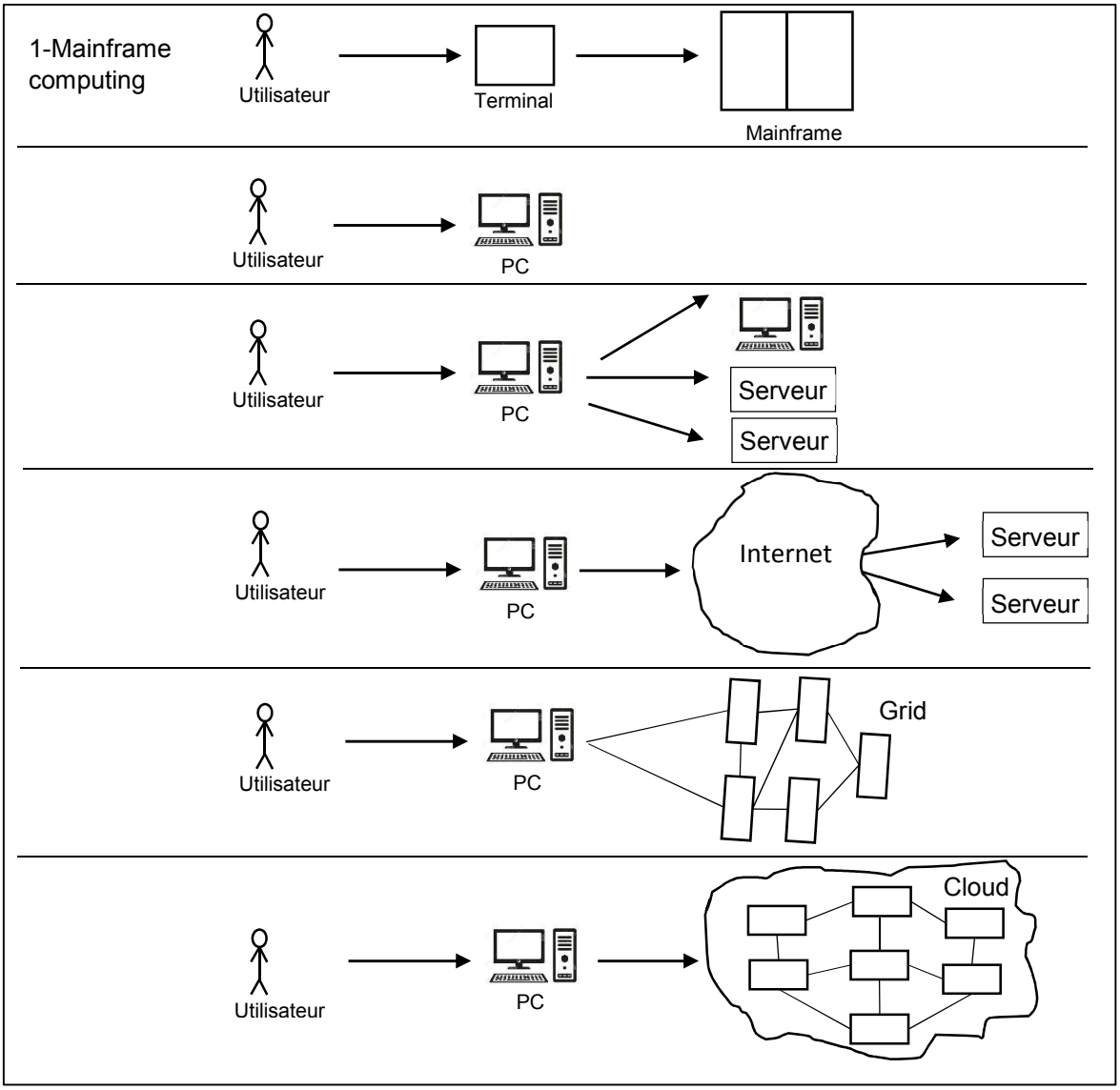

Figure 12 Modèles de calcul

2.4. Grid Computing

Le Grid Computing permet l'agrégation des ressources distribuées pour y accéder d'une manière transparente. La plupart des Grids cherchent à partager des ressources de calcul et de stockage réparties entre différents domaines administratifs. Leur principal objectif est d'accélérer une large gamme d'applications scientifiques telles que la modélisation climatique, la conception de médicaments et l'analyse des protéines [22].

Donc, le Grid Computing c'est un modèle et architecture pour rassembler un ensemble de ressources informatiques (Serveurs, puissance de calcul, capacité de stockage…etc), pour construire virtuellement une machine puissante dédiée à aboutir pour un objectif commun.

2.5. Les caractéristiques de Cloud Computing

Le Cloud Computing modèle définit par le NIST contient essentiellement cinq caractéristiques[20], à savoir :

2.5.1. Accès à la demande par le consommateur

Un consommateur peut utiliser des ressources informatiques, telles que le temps du calcul et la capacité de stockage, automatiquement au besoin sans nécessiter d'interaction humaine avec chaque fournisseur de services.

2.5.2. Large accès au réseau

Les ressources de Cloud Computing sont accessibles via l'Internet et en utilisant des techniques standardisées. Ce qui donne la possibilité de s'en servir en utilisant un téléphone portable, une tablette ou bien un PC.

2.5.3. Réservoir de ressources (Resource pooling)

Les ressources informatiques sont partagées pour servir de multiple consommateurs. Elles peuvent être physiques ou virtuelles, et elles sont dynamiquement allouées et libérées selon les demandes des consommateurs.

2.5.4. Redimensionnement rapide (élasticité)

En fonction de la demande, les ressources et les capacités peuvent être vendues rapidement et même dans certains cas automatiquement, provisionnées et libérées élastiquement. Pour le consommateur, les capacités disponibles pour l'approvisionnement semblent souvent illimitées et peuvent être appropriées à n'importe quelle quantité à n'importe quel moment.

2.5.5. Paiement à l'usage

La facturation est calculée en fonction de la durée et de la quantité de ressources utilisées. Cette caractéristique permet au consommateur et même au fournisseur de contrôler l'utilisation d'un service de cloud (il n'y a pas généralement un coût de mise en service c'est l'utilisateur qui réalise les opérations).

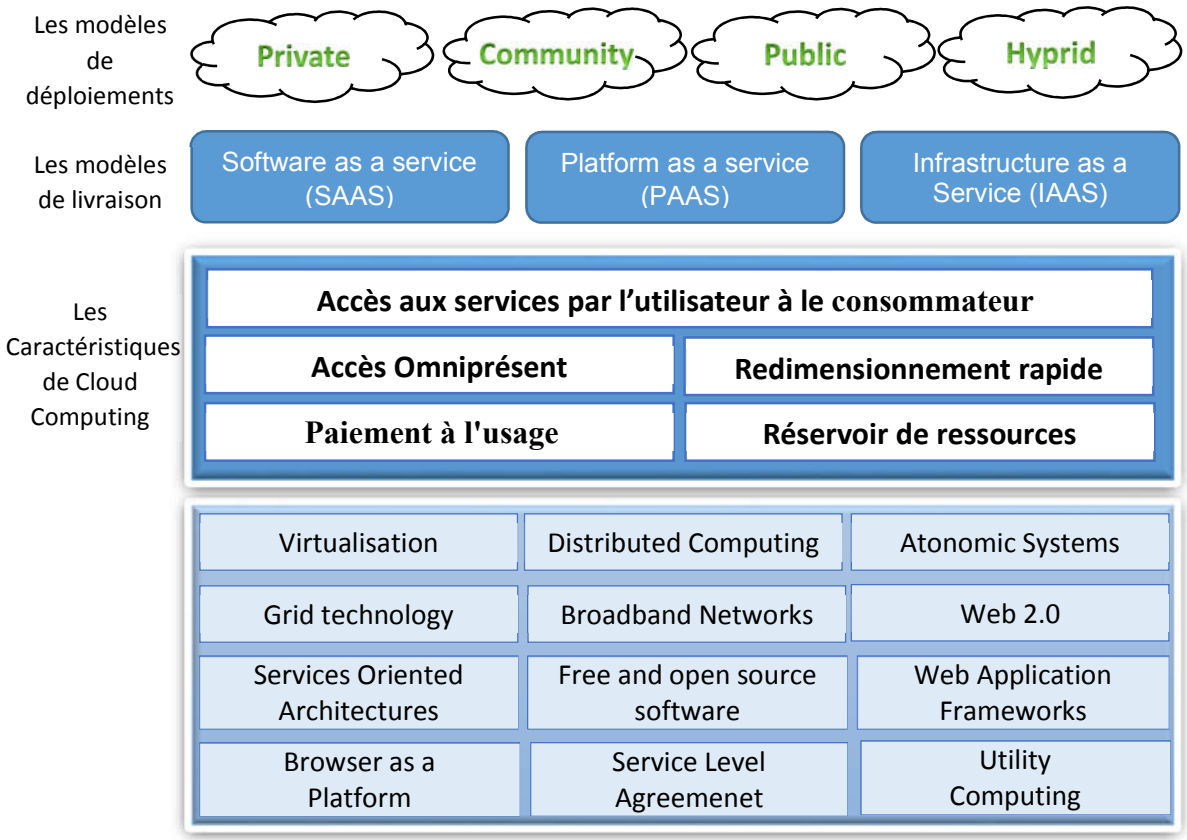

Figure 13 le modèle de cloud computing

2.6. Les modèles de livraison

Dans le cloud computing les fonctionnalités sont offertes aux consommateurs comme des services. Ces services peuvent être devisés selon la nature du service livré en trois grands groupes (cf. figure 13), à suivre :

2.6.1. Software as a Service (SAAS)

Sont des services pour les clients de ce cloud. Ces services peuvent être des applications traditionnelles et simples comme le traitement de texte. C'est le modèle de livraison de service le plus utilisé dans le cloud, tel que SaaS qui permet aux consommateurs d'exploiter des applications à distance du cloud[20]. Par exemple les applications de gestions des emails (Gmail de google, Yahoo mail) sont une sorte de SaaS.

En plus de partager la plateforme matérielle ou ces services qui sont exécutées, le software lui-même en cours d'être délivré est une ressource partagée entre les consommateurs. SaaS donne la possibilité pour un consommateur d'utiliser les applications du fournisseur qui

sont fournies sur une infrastructure cloud. Les applications sont accessibles à partir de différents périphériques clients par une interface légère comme un navigateur Web (par exemple, un email Web) ou une interface de programme. Le consommateur ne gère pas et ne contrôle pas l'infrastructure de cloud, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage ou même les capacités d'application individuelles, à l'exception possible des paramètres de configuration d'application spécifiques à l'utilisateur[20].

SaaS élimine la nécessité d'acheter, d'installer et de gérer les softwares afin de les utiliser. Il a aussi l'avantage que les softwares sont rendues accessibles de partout avec une connexion Internet. L'exemple le plus connu dans le monde pour le modèle SaaS est à travers le Salesforce.com, qui est une entreprise qui utilise le Cloud Computing pour délivrer ces services spécialisées dans la gestion de la relation client (GRC). Les applications nécessaires à une entreprise pour gérer ces relations avec les clients comme, la gestion des ressources humain et la planification des ressources, sont toutes fournies par SalesForce.com aux autre entreprises sur une plateforme de cloud. Ces entreprises éliminent l'immense effort de s'équiper en matériels informatiques spécialisés et d'installer des logiciels spécialisés, par le fait d'avoir une légère infrastructure et un compte chez SalesForce.com pour utiliser ces services.

2.6.1.1. Les avantages de SaaS

Coût : Du côté matériel, le coût est notablement réduit car l'entreprise remplace la complexe infrastructure par un simple réseau local connecté avec l'Internet. De côté logiciel, au lieu de payer l'application d'un seul coup, avec SaaS le coût est réglé chaque période de temps et le paiement est à l'usage. Le coût de maintenance est diminué, parce que il est partagé entre tous les clients de ce service.

Temps : en comparaison avec les traditionnels modèles, dans SaaS les matériels et les applications (services) sont déjà installés et configurés qui permet de gagner un temps considérable.

Misa à niveau facile : parce que les services sont hébergés chez les fournisseurs, la mise à niveau des applications ne nécessite aucun changement de matériel ou bien téléchargement de logiciels complémentaires chez les clients et toutes les charges sont sur les fournisseurs.

2.6.2. Platform as a Service (PaaS)

PaaS permet aux consommateurs de développer et déployer sur une infrastructure cloud des applications en utilisant des langages et outils de programmation fournis par le fournisseur.

Le consommateur ne gère pas et ne contrôle pas l'infrastructure interne de cloud tel que le réseau, les serveurs, les systèmes d'exploitation, le stockage, mais il a un control sur les applications déployées [20].

PaaS offre un environnement de développement pour créer des applications pour les utiliser sur le cloud. La différence essentielle avec un SaaS est que le SaaS est destiné aux consommateurs, par contre le PaaS est destiné aux développeurs. Les fournisseurs de PaaS les plus connus sont :

Google App Engine : permet de développer des applications en utilisant Java ou Python comme langage de programmation dans un environnement cloud [22]. Les applications créées par App Engine caractérisent par une élasticité automatique, et permet d'allouer plus de ressources selon le nombre de requêtes par les consommateurs.

Microsoft Azure : Microsoft Azure Cloud Services, offre la possibilité de développer des applications en utilisant la pile des langages Dot Net (C#, VB et ASP.Net), le système d'Azure est composé de nombreux éléments, le Windows Azure Fabric Controller fournit une fiabilité et une mise à niveau automatique, et il gère les ressources mémoire et l'équilibrage de charge. Le .Net Service Bus registres connecte les applications entre elles [22].

2.6.3. Infrastructure as a Service (IaaS)

C'est la possibilité pour les consommateurs d'utiliser des ressources liées aux traitement, le stockage, les réseaux et d'autres ressources informatiques fondamentales. Le consommateur sera capable de déployer et d'exécuter des logiciels arbitraires, ce qui peut inclure des systèmes d'exploitation et des applications. Le consommateur ne gère pas et ne contrôle pas l'infrastructure de cloud, mais il contrôle le système d'exploitation, le stockage et les applications déployées avec un contrôle limité des composants du réseau sélectionnés (par exemple, des pare-feu hôtes) [20].

IaaS offre des ressources comme le stockage, les technologies de communication et d'autres ressources matérielles de manière virtuelle. L'exemple le plus connu pour les IaaS c'est l'Amazone Web Services, qui a commencé en 2006, et il est composé de plusieurs services de cloud, parmi lesquels on cite : EC2 pour des serveurs virtuels, SimpleBD pour les bases de données structurés, S3 pour le stockage.

2.7. Les modèles de déploiement

Le modèle de déploiement est habituellement divisé en quatre modes : public, privé, communautaire et le cloud hybride. Leurs descriptions seront présentées dans ce qui suit :

2.7.1. Le Cloud Privé

L'infrastructure du cloud privé est provisionnée pour une utilisation exclusive par une seule organisation avec plusieurs consommateurs. Le cloud privé peut être appartenir, être géré et être exploité par l'organisation, par un tiers ou par une combinaison de ces derniers, et il peut exister à l'intérieur ou à l'extérieur des locaux de l'organisation [20].

2.7.2. Le Cloud public

L'infrastructure de cloud public est accessible par l'Internet. Elle est ouverte au public ou à de grands groupes industriels. Il peut appartenir, être géré et être exploité par une entreprise de commerce, par une organisation académique, ou par une organisation gouvernementale. Il existe sur les lieux du fournisseur du cloud public [20].

2.7.3. Le Cloud communautaire

L'infrastructure du cloud communautaire est provisionnée pour une utilisation exclusive par une communauté spécifique de consommateurs qui partagent les même domaines d'intérêts. Il peut appartenir, être géré et être exploité par une organisation ou plus dans la communauté, par un tiers ou par une combinaison d'eux [20].

2.7.4. Le Cloud hybride

Le cloud hybride est une composition de deux ou plusieurs types de clouds (privé, public et communautaire) [20]

Au cours des dernières années, la technologie du cloud computing est devenue plus indispensable. Les racines de cette technologie remontent au début des années 1960, où John Mc- Carthy était le premier qui a discuté l'idée de l'informatique utilitaire [23]. Cette idée a été reformée et devenue le cloud computing aujourd'hui. Les architectures de cloud et ses complexes technologies sont transparentes pour les utilisateurs, car ses fonctionnalités sont offertes aux clients en tant que services. Les utilisateurs peuvent demander des opérations sur les clouds en utilisant ces services. Selon la nature de ces services, on distingue trois modèles

de livraison de services : le premier, le *Software as a Service* (SaaS) ou les logiciels ordinaires comme le traitement de texte, image sont fourni aux consommateurs comme des services de cloud. Le deuxième modèle, c'est le *Platform as a Service* (PaaS) les consommateurs des services dans cette cas ne sont pas des utilisateurs habituels mais plutôt des développeurs, parce que le PaaS offre un environnement de développement complet à héberger dans un cloud et à livrer aux développeurs comme un ensemble de services. Le troisième modèle, c'est l'*Infrastructure as a Service* IaaS où la nature des ressources est beaucoup plus reliée avec l'infrastructure du cloud, comme la capacité de calcul et de stockage.

D'après le mode de déploiement ou bien le type de consommateurs cibles, il existe quatre types de cloud : le cloud public destiné au grand publique, le cloud privé pour une utilisation exclusive par une organisation unique, le cloud communautaire pour une utilisation exclusive par une communauté et en dernier le cloud hybride qui peut être une combinaison des trois types mentionnés.

Pour qu'une infrastructure informatique soit considérée comme une infrastructure de cloud, elle doit vérifier cinq caractéristiques essentielles : premièrement les consommateurs doivent utiliser les services à la demande, et être capables de les invoquer à n'importe où et à n'importe quel moment. Les ressources de cloud doivent être provisionnées d'une manière transparente et par les quantités désirées par les consommateurs. Ces ressources sont partagées et peuvent être physiques ou virtuelles, et en dernier, les consommateurs les utilisent gratuitement ces ressources ou bien ils payent leurs usages.

2.8. L'architecture du Cloud Computing

L'architecture du cloud computing peut être devisée en deux sections : la première section s'appelle le Front End. Qui est la partie visible pour les consommateurs, et la seconde c'est le Back End qui est la partie cachée pour les consommateurs et elle est gérée par le fournisseur. L'interaction entre le Front End et le Back End est assurée via l'Internet.

2.9. Le modèle cubique du coud computing

Pour assurer une collaboration sûre dans les plateformes de Clouds, le forum de Jericko a défini quatre critères pour différencier les formations de Clouds entre eux[24]. Ces critères

sont présentés comme quatre dimensions qui forment un modèle cubique d'une plateforme de Cloud. Les dimensions qui forment le modèle cubique sont présentés :

-Dimension Interne (I) / Externe (E) : C'est la dimension qui définit l'emplacement physique des données (où le Cloud existe) à l'intérieur ou à l'extérieur des frontières de l'organisation.

-Dimension : Propriétaire (P) / Ouvert (O) : C'est la dimension qui définit l'état de propriété de la technologie du cloud, les services, les interfaces, etc. et le degré d'interopérabilité, ainsi que l'habilitation "Transmission de données / applications" entre les systèmes propres et autres formes de Clouds. La possibilité de retirer des données à partir d'une forme de Cloud ou de la déplacer vers une autre forme sans contrainte. Elle indique également toute contrainte sur cette dimension et la capacité de partager des applications.

-Dimension **:** Perimeterised (Per) / De-perimeterised (D-p) Architectures

Cette troisième dimension représente la « mentalité architecturale » - opérez-vous à l'intérieur de votre périmètre informatique traditionnel ou à l'extérieur ? De-périmétrie a toujours été lié à échec graduel / suppression / rétrécissement / effondrement informatique traditionnelle basée sur le silo périmètre.

-Dimension : Insurcié/Externalisé (Insourced / Outsourced I/O)

C'est la réponse à la question : qui est le fournisseur de services du Cloud ? s'ils sont fournis par une troisième partie alors c'est externalisé /Outsourced (O). Par contre s'ils sont fournis par le propriétaire lui-même alors c'est Insurcié /Insourced (I).

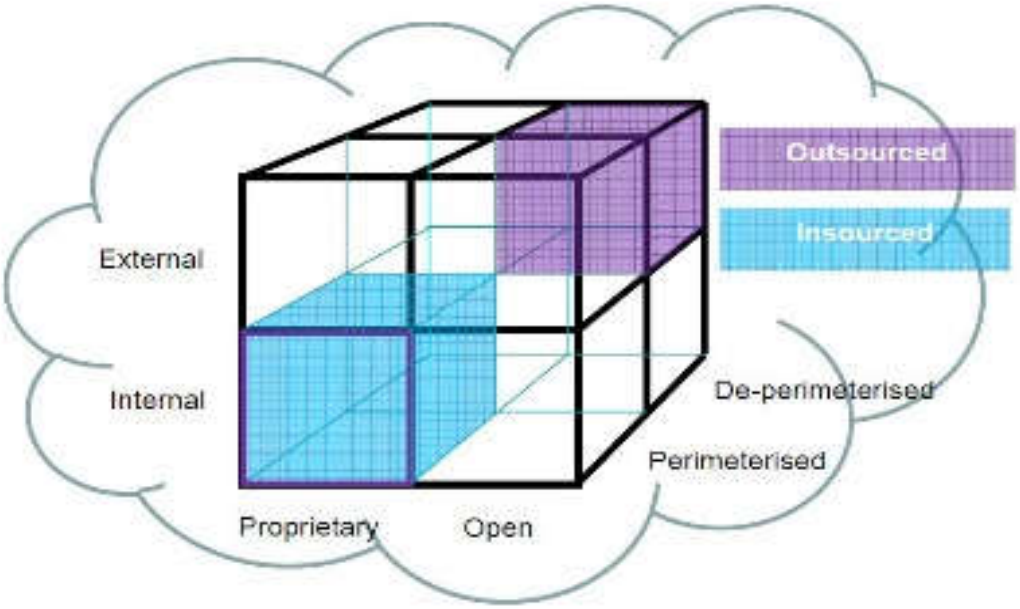

Figure 14 Le modèle cubique de cloud computing

Si une plateforme de Cloud Computing peut être posée dans le cube droit, haut en front, alors cette plateforme peut être jugée qu'elle est Externel, Open, Perimeterised et Outsourced.

2.10. Les services web VS les services de Cloud

Il y'a souvent confusion entre les deux termes scientifiques services web et « cloud service » (ou service de cloud). Comme montre la figure 15 (la source de la figure [25]) qui représente un diagramme de Venn illustre les relations entre l'architecture orientée services, cloud computing et les services web. On voit clairement que les services web encapsulent le cloud computing, par contre l'architecture orientée services peut exister sans les services web, malgré qu'ils représentent le meilleur choix pour construire une SOA.

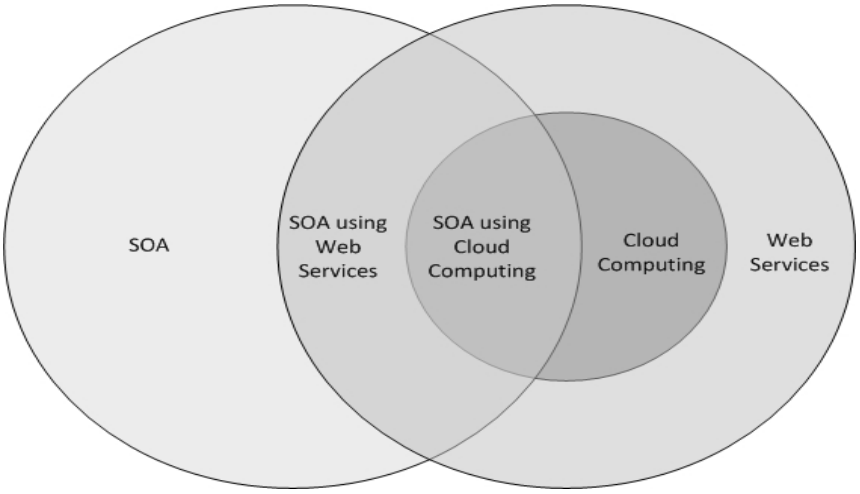

Figure 15 Diagramme de Venn entre les services Web, SOA et le cloud computing

2.11. Conclusion

Le cloud computing est considéré comme la prochaine étape dans l'évolution de l'internet. Le mot cloud « nuage » en français signifie qu'on va rassembler tout dans ce nuage : le matériel, les applications, le stockage et même la configuration du réseau et d'offrir aux utilisateurs des services pour les utiliser à n'importe où et à n'importe quand. En fonction des utilisateurs cibles de cloud computing peut faire la différence entre : le cloud public destiné au public, le cloud privé à utilisation destiné exclusive et le cloud hybride.

Chapitre 3

Processus de composition de services dans le cloud computing

3.1. Introduction

Au cours des dernières années, le cloud computing devient une nouvelle approche pour distribuer l'informatique. Dans le cloud computing on se sert des services pour utiliser des ressources ou de bien lancer des applications. Mais des fois on veut avoir les résultats de plusieurs services en même temps pour améliorer la performance de ce cloud. Par conséquent, on est besoin d'une approche pour la composition de services Web dans le cloud computing, et plusieurs travaux de recherche sont déjà réalisés pour traiter ce sujet. Dans ce chapitre nous allons examiner un ensemble de travaux connexes sur la composition de services web dans le cloud computing. On termine le chapitre par une contribution dans le domaine.

3.2. Le problème de la composition de services

Les utilisateurs des plates-formes clouds utilisent des services pour satisfaire leurs demandes. Mais ces demandes deviennent très complexes, et ils ont besoin plus d'un seul service pour accomplir une demande. Le processus de collecte d'un ensemble des services pour satisfaire une demande de l'utilisateur est appelé la composition de services.

En générale, l'architecture de cloud est composée de deux parties : la première, appelée Front End, qui est visible pour les clients, et généralement représentées par un ordinateur ou tout type de terminal. La deuxième partie, appelée Back End, qui rassemble les technologies complexes du cloud qui sont transparents aux clients. La connexion entre deux parties est raccordée via Internet, où les clients utilisent le frontal 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 sur le cloud même. 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.

Le processus de construction d'un service à partir de l'ensemble des services sélectionnés pour donner à l'utilisateur le meilleur résultat est appelée le Cloud Computing Service Composition (CCSC) [26]. La figure 16 montre le mécanisme de fonctionnement d'un environnement cloud durant l'exécution d'une requête envoyée par un utilisateur.

Egalement, la figure 16 montre qu'il y a une transparence totale pour l'utilisateur durant l'exécution de sa requête. L'opération de composition de services est achevée sans l'intervention de l'utilisateur. Aussi, on voit clairement que le processus de composition de services conduit à la sélection d'un ensemble de cloud bases, où les services impliquées dans le processus de composition sont situés. L'exemple donné dans la figure 16 montre la requête de l'utilisateur qui est représenté par la liste des taches $\{t_1, t_2, t_3, t_4\}$ exécute par les services $\{S_1, S_2, S_3, S_4\}$, les compositions possibles pour cette exemple sont les suivantes (on ne cite pas ici toutes les compositions, mais juste des instances) :

Tableau 2 La relation entre la composition de service et la sélection des clouds

La composition de services	**le combinaison de cloud bases sélection**
$C_1/S_1 – C_2/S_2$ - $C_1/S_3 – C_3/S_4$	$C_1 – C_2 – C_3$
$C_1/S_1 – C_2/S_2 – C_4/S_3 – C_3/S_4$	$C_1 – C_2 – C_3 – C_4$
$C_1/S_1 – C_3/S_2$ - $C_1/S_3 – C_3/S_4$	$C_1 – C_3$
$C_1/S_1 – C_2/S_2$ - $C_1/S_3 – C_2/S_4$	$C_1 – C_2$

Il est évidemment que, s'il est possible de diminuer le nombre des cloud bases impliqués dans le processus de composition, cela permet de réduire le coût de la communication entre ces services. Cette réduction apparait clairement dans les points suivants :

1. Temps de réponse : si le nombre de cloud bases impliqués dans le processus de composition est élevée, le temps de réponse sera plus grand, en particulier si les services sont interdépendants entre eux (la sortie d'un service est l'entrée d'un autre).
2. Prix d'accès : si l'accès à l'ensemble des cloud bases n'est pas gratuit, il est clair que l'utilisateur préfère utiliser le minimum de nombre de ces bases.
3. La bande passante du réseau : à partir du point (1), il est clair que si le processus de composition arrive à trouver un nombre minimal de cloud bases, ce qui permettra de réduire la quantité de données échangées sur le réseau.

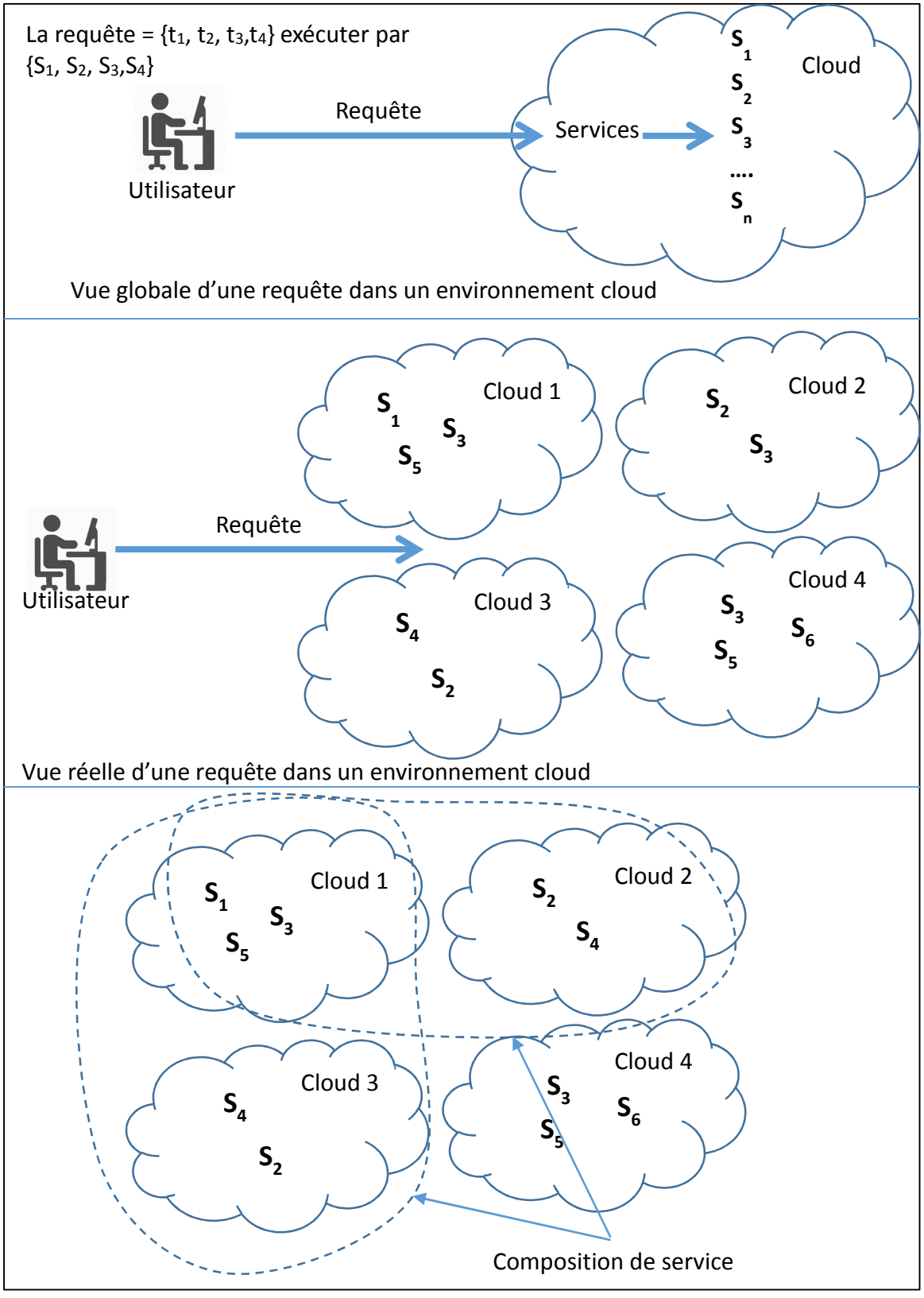

Figure 16 la composition de services dans le cloud computing

Les services sont conçus pour effectuer un ensemble d'opérations atomiques, où une opération ne satisfait pas le demande du client mais juste une partie de celui-ci. Pour satisfaire

complètement la demande du client, nous avons besoin de composer plusieurs services pour construire un service composé capable d'effectuer des tâches complexes. Par exemple, si un client veut voyager pour passer ses vacances dans un pays étranger, son agence lui proposera de réserver un vol, de réserver un hôtel, de louer une voiture et de participer à certaines activités organisées, Ces tâches sont proposées comme des services sur des plateformes cloud, et pour chaque tâche, il existe plusieurs services pour effectuer une tâche donnée, Parce que ces multiples services sont différents en plusieurs termes, l'agence a besoin de stratégies ou de mécanismes pour sélectionner un service pour chaque tâche et pour composer les services sélectionnés dans un seul service afin de maximiser la satisfaction du client.

3.2.1 Cycle de vie de la composition de services web

Comme illustré dans la figure 17 en schématisant le cycle de vie de la composition des services Web, ces activités consistent à [27]:

1. découvrir les services Web susceptibles de coopérer pour répondre à la requête traitée ;
2. vérifier leur compossibilité et générer les plans (ou solutions) de composition possibles ;
3. sélectionner le ou les meilleurs plans qui répondent aux exigences et préférences du client en termes de propriétés non-fonctionnelles (les exigences du client) ;
4. exécuter le service composite.
5. Publier les services composites

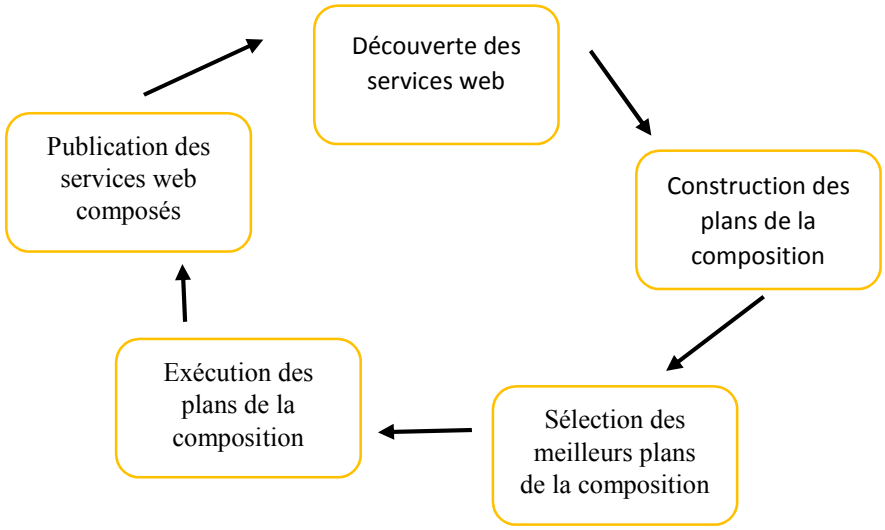

Figure 17 Cycle de vie de la composition de services web

3.3. Etudes des anciens travaux de la composition de services

Le sujet de la composition de services web dans le cloud computing a été largement traitée au cours des dernières années. Un grand travail peut être consulté dans [1], qui est une revue systématique de la littérature sur le problème et les solutions proposées jusqu'à 2014.

Le problème de la composition de services est considéré comme un axe de recherche récent. C'est pour cela il existe plusieurs articles publiés dans le domaine. Dans les paragraphes suivants on va présenter un échantillon de ces travaux qu'on considère pertinents pour notre recherche.

3.3.1. La composition de services comme un système multi-agents

Certains travaux modélisent le problème de la composition de services web dans le cloud computing en tant qu'un système mutli-agents. Comme les travaux présentés dans [28] et [29], les auteurs dans [28] tirent profit des caractéristiques des agents coopératifs pour proposer une approche de la composition de services en trois étapes: 1) l'auto-organisation des agents dans un réseau social, 2) le calcul distribué de la composition abstraite et optimale des services et 3) la sélection de clouds pour la composition de services dans une base de multiple clouds. En revanche, Anouer Bennajeh dans [29] tente d'augmenter le bénéfice, et de réduire le coût et le temps de réponse dans le processus de composition en se fondant sur les caractéristiques de la technologie d'agents logiciels.

3.3.1.1. Agent-based Service Composition in Cloud Computing

Les auteurs du travail intitulé « *Agent-based Service Composition in Cloud Computing »* ont proposé un système multi-agents pour la composition de services dans le cloud computing. L'architecture du système est présentée dans la figure 18 [30] :

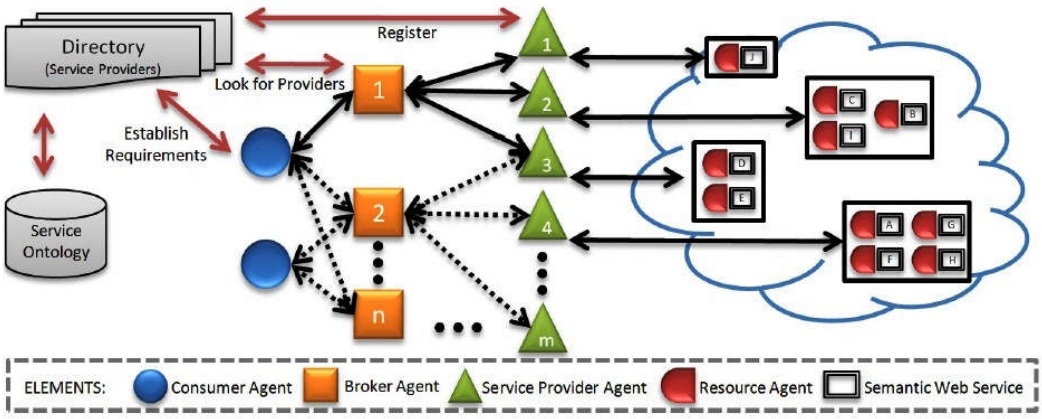

Figure 18 Composition de services basée agents dans le cloud.

Tel que :

- *service ontology :* est une représentation formelle des services et les tâches atomiques que ces services peuvent remplir.
- *directory* : est une liste d'agents fournisseurs des services disponibles. Chaque entrée est associée à un fournisseur de services qui contient son nom, son emplacement (par exemple , l'adresse URI) , et des capacités .
- *semantic web service* (SWS) est un service Web dont la définition est mappé à une ontologie , qui caractérise la fonction et de son domaine d'application.
- *consumer agent* (CA) soumet les intérêts d'un consommateur aux broker agents.
- *resource agent* (RA) orchestre un service web sémantique. En outre, un RA contient une référence à une tache atomique, qui peut être remplie par ses SWS .
- *service provider agent* (SPA) gère un ensemble de RA . Sa fonction principale est de coordonner et synchroniser les RA.
- *broker agent* (BA) accepte les demandes des agents des consommateurs CA et crée une seule service virtualisé par le biais de services composants à déployer sur un ou plusieurs clouds. Un BA utilise les exigences des consommateurs à la recherche de SPA, ensuite il démarre le processus de composition en adoptant le protocole de contrat net bien connu pour la sélection des services et l'allocation des tâches.

3.3.1.2. Self-Organizing Agents for Service Composition in Cloud Computing

Dans une communication intitulé « *Self-Organizing Agents for Service Composition in Cloud Computing »,* ces auteurs présentent un système multi-agents pour la composition de services dans le cloud computing (cf. figure 19)[31], dont lequel chaque ressource est

représentée et instanciée par un agent. La composition de service est supportée par deux techniques coopératives basées sur un agent de résolution de problèmes, à savoir :

-Acquaintance networks : liste d'agents de système et ses capacités à utiliser pour traiter les informations incomplètes sur les locations de ressources distribuées du cloud.

-Le protocole contrat net CNP : est utilisé pour sélectionner dynamiquement des services en se basant sur les frais de services.

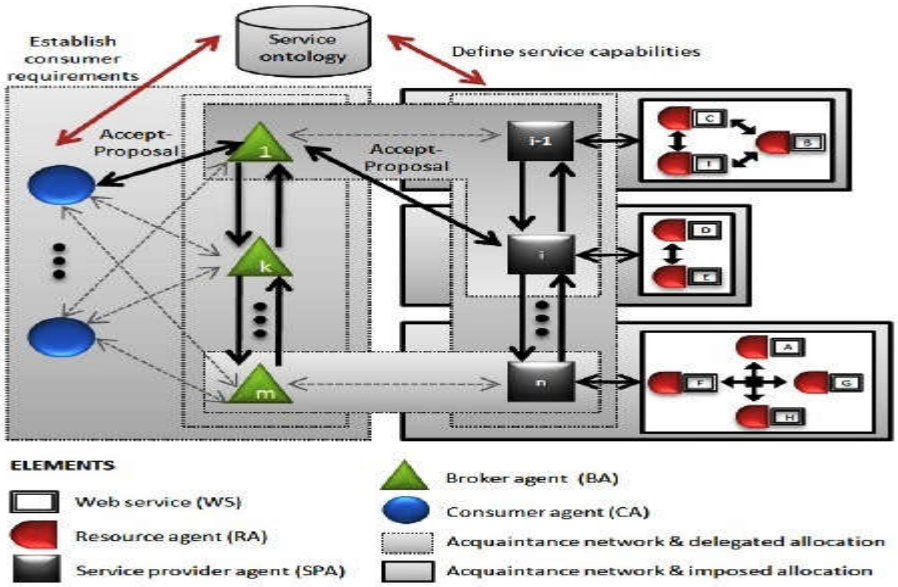

Figure 19 Architecture de système.

-*Service web* : est une interface pour accéder à une application à des ressources qui sont accessibles à distance.

-*Ressource Agent RA* : est utilisée comme gestionnaire d'un service web.

-*Service Provider Agent SPA* : est utilisée pour administrer un ensemble de ressources agents.

-*Broker Agent BA :* est l'intermédiaire entre les agents de consommateurs et les services provider agents.

-*Consumer Agent CA* : représente un consommateur dans le *cloud*.

3.3.2. Optimisation combinatoire pour multi cloud composition du service

Le travail réalisé par Heba Kurd [32], a proposé un algorithme d'optimisation combinatoire pour plusieurs cloud bases appelé COM2. Le principe de l'algorithme est de trier la liste des clouds dans l'ordre décroissant, pour assurer que le cloud avec le nombre maximal de services est sélectionné en premier lieu. Il est clair que le résultat de l'algorithme dépend de

la position des services demandés par l'utilisateur dans les clouds. En d'autres termes, si les services demandés sont dans les plus petits clouds, l'algorithme sera coûteux en terme de temps, et vice versa.

```
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 ← φ //B is the Combiner List of clouds sorted in decreasing order of number of services
6: n ← number of clouds in the MCE
7: P ← φ //P is the Composer List of services
8: Get the users request R.
9: Select the cloud Cn that contains the largest number of services
10: //Check if Cn contains a new service that can fulfil the user request:
11: If ((Cn∩R) - P==φ) // 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 + Cn  //add the cloud to the Combiner List
15:     P = P∪(Cn ∩ 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 20 Pseudo code de l'algorithme COM2

L'algorithme en figure 20 montre le pseudo code de l'algorithme COM2 proposé par les auteurs de ce travail, cette algorithme fonctionne comme suite :

- Après les initialisations des variables (la requête de l'utilisateur et des informations concernent le multi cloud environnement), l'algorithme accepte la demande de l'utilisateur pour une composition de services et génère une suggestion en sélectionnant le premier cloud qui contient le plus grand nombre de services.
- Pour plus de simplicité, les auteurs ont supposé que les clouds sont triés par ordre décroissant en fonction du nombre de fichiers de services. Ce tri aide à identifier rapidement le cloud avec le plus grand nombre de fichiers de services.
- Par la suite, l'algorithme détermine si l'un des fichiers de cloud énumérés en premier peut répondre à la demande de l'utilisateur. Si aucun fichier de clouds n'est identifié, le cloud suivant est vérifié jusqu'à ce qu'un cloud approprié sont obtenu. Sinon, l'algorithme se termine lorsque le dernier cloud est atteint.

- Une fois un cloud approprié est trouvé, il est ajouté à Combiner List B, et ses fichiers de services sont ajoutés à la liste Composer P. Si la demande de l'utilisateur n'est pas satisfaite, le cloud suivant qui contient de nouveaux services et qui peut répondre à la demande de l'utilisateur est sélectionné.
- Pour s'assurer que le cloud sélectionné contient des services qui ne figuraient pas auparavant dans la liste des compositeurs, l'algorithme soustrait le contenu de la liste des compositeurs des nouveaux services du cloud sélectionné (Cn \ R). Si de nouveaux services ne sont pas disponibles, le cloud sélectionné est ignoré et le cloud suivant de la liste est pris en compte.
- Les étapes du processus sont répétées jusqu'à ce que la demande de l'utilisateur soit satisfaite. La Liste des Combinateurs est ensuite envoyée au compositeur.

3.3.3. Optimisation des colonies de fourmis appliquée aux compositions de services

Un autre travail dans le domaine de la composition du services a été proposé par Qiang Yu [33], dans lequel il propose deux algorithmes : le premier est un algorithme glouton appelé Greedy-WS. Son principe est de sélectionnez le cloud qui contient la plupart des services demandés. Le deuxième algorithme est un algorithme d'optimisation de colonies de fourmis appelé ACO-WSC. L'algorithme prend en entrée un digraphe (cf. figure 21), ses nouds sont les différentes cloud bases, et les fourmis voyagent sur le graphe pour trouver une combinaison de clouds.

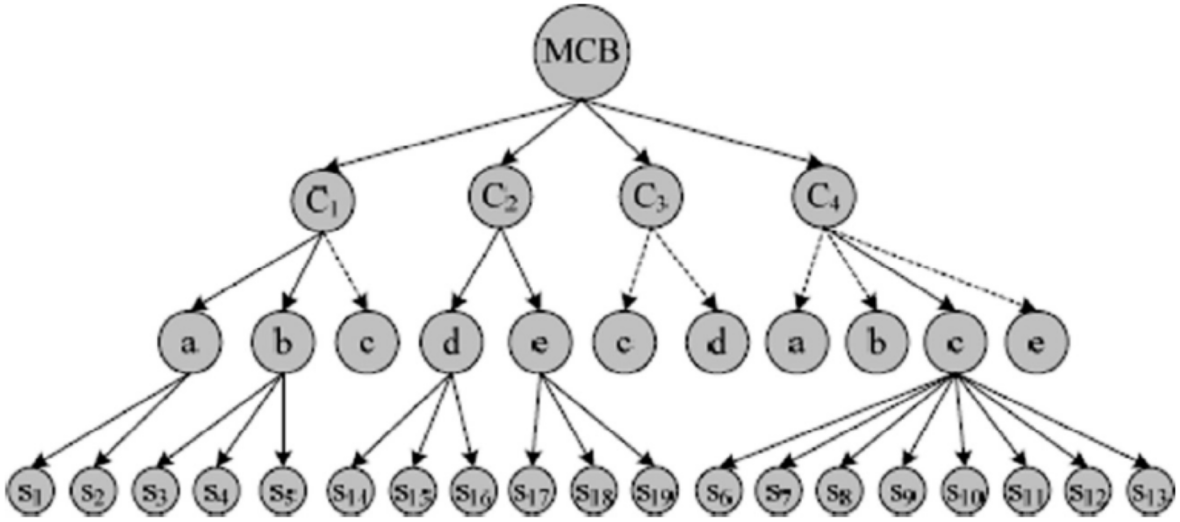

Figure 21 Arbre de la multiple cloud bases

Comme le montre l'algorithme Greedy-WSC en figure 22, les auteurs présentent un algorithme de composition de services web (Greedy-WSC) pour trouver la combinaison de clouds optimale pour un ensemble de fichiers d'un service donné. L'algorithme sélectionne d'abord le cloud de $\{C_1, C_2, \ldots, C_n\}$ qui contient la plupart des services requis.

Après avoir marquer le cloud sélectionné, l'algorithme continue à sélectionner le cloud non marqué qui contient la plupart des autres services requis. L'algorithme répète ce processus de sélection de clouds jusqu'à ce que les clouds sélectionnés couvrent tous les services requis.

```
Algorithm 1. Greedy-WSC;
Input: C1,C2, . . .,Cn: set of cloud bases, each Ci = {si1, si2,. . ., sik} is a set of services;
        n: number of cloud bases;
        req-set: user's requested services composition;
Output: Sc: resulting cloud combination;
Begin
        Sr = req-set; Sc = U;
        While Sr – U do
                Max-gain = 0;
                For each unmarked cloud base Ci do
                        gain = jCi \ Sr j;
                                if gain> Max-gain then
                                m = i; Max-gain = gain; mark cloud base Ci;
                                end if
                end for
                Sc = Sc [ fCmg; Sr = Sr _ Cm;
        End while
End
```

Figure 22 Pseudo code de l'algorithme Greedy-WSC

ACO est un système multi-agents où les communications entre fourmis artificielles entraînent un comportement de rétroaction positive qui guide la colonie de fourmis à converger vers la solution optimale. Les informations sur les phéromones (qui simule la substance chimique que les vraies fourmis reposent sur un chemin qu'elles ont passé) sont assignées aux arcs d'un graphe. Dans ACO, les fourmis artificielles voyagent dans le graphe pour rechercher les chemins optimaux et suivent les informations de phéromones et les informations des heuristiques locales spécifiques au problème. La phéromone sur chaque arc est évaporée à un certain rythme à chaque itération. Il est aussi mis à jour en fonction de la qualité des chemins contenant cet arc. Les fourmis artificielles sont généralement associées à une liste qui enregistre leurs actions précédentes, et ils peuvent appliquer certaines opérations supplémentaires (telles que la recherche locale, le croisement et la mutation) pour améliorer la qualité des résultats (cf. figure 23).

Comparé aux algorithmes génétiques, ACO a quelques avantages tels que le feedback positif, l'informatique distribuée, et une recherche heuristique gloutonne constructive.

```
Algorithm ACO-WSC;
Input: C = {C1,C2, . . .,Cn}: set of cloud bases, where Ci = {si1, si2, . . ., sik} is a set of services;
        n: number of clouds;
        m: number of ants;
        req-set: user's composition request;
Output: Sc: composition sequence;
Begin
        While not terminate condition do
                For i = 1 to M do //the i-th ant//
                        Randomly select a cloud base Ck;
                        Sc (i) = {Ck}; Sr(i) = req-set–Ck;
                        While Sc (i) – U do
                                Select a cloud base Cj according to the probability pi
                                kj defined in (1);
                                Sc (i) = Sc[ fCjg; Sr(i) = Sr(i) _ Ck;
                        End while;
                End for
                Update the pheromone on each edge;
                Sc = the best solution obtained so far;
        End while
End
```

Figure 23 Pseudo code de l'algorithme ACO-WSC

3.3.4. AI Planning and Combinatorial Optimization for Web Service Composition

Dans le travail intitulé « *AI planning and combinatorial optimization for web service composition in cloud computing »* [34]. Ses auteurs abordent le problème de la composition du services Web dans des environnements avec plusieurs clouds. En raison du coût très élevé de communication entre les services Web de différents clouds, le but de ce travail est de minimiser efficacement le nombre de clouds impliqués dans une séquence de composition de services. Qu'il soit considéré un problème difficile car il couple la planification à cause des contraintes de fournisseurs de services et l'optimisation pour la sélection de clouds.

L'idée clé de ce travail est de convertir ce problème en un ensemble de modèles de couverture, et de trouver une combinaison sous-optimale de clouds en utilisant un algorithme d'approximation. En utilisant IA planification pour composer des services Web. Plus précisément, l'idée-clé modélise la multiple base de clouds comme un arbre, et utilise un algorithme d'approximation pour optimiser la sélection du clouds, en employant une IA système de planification pour la composition du service.

Les auteurs présentent leur travail en quatre algorithmes :

1-All Clouds Combinitons : Dans cet algorithme, ils couvrent d'abord tous les services Web impliqués dans MCB (multiple cloud bases) et la demande de composition dans un domaine de composition et un problème de composition, respectivement. Ensuite, il exécute un

planificateur de composition pour trouver une solution de planification. S'il existe une séquence de plan de composition, elle est renvoyée au demandeur de service.

2-Base Cloud Combination : cet algorithme permet de trouver une combinaison optimale de clouds avec un nombre minimum de clouds impliqués dans la composition. Cet algorithme, énumère récursivement toutes les possibilités de combinaison de clouds jusqu'à ce qu'une solution de composition soit trouvée dans une combinaison.

3-Smart Cloud Combination : La méthode de Smart Cloud combinaison de clouds commence par modéliser d'un MCB en tant qu'arborescence, puis de trouver un ensemble minimum de clouds en effectuant une recherche dans l'arborescence MCB.

Cet algorithme est constitué de quatre étapes : la première étape consiste à trouver MRS (Minimum Request Set), s'il n'existe pas de combinaison de clouds satisfaisant la requête de l'utilisateur si son MRS ne peut pas être trouvé dans cette étape. Dans la deuxième étape, il faut prétraiter chaque cloud dans MCB pour réduire l'espace de ses fichiers de service et définir son coût. Plus précisément, pour chaque fichier de services sf impliqué dans un cloud C, s'il est également inclus dans MRS, alors sf est ajouté dans le cloud réduit C'. Autrement, sf est retiré du cloud réduit et le nombre de services impliqués dans sf est ajouté au coût du cloud réduit C'. Après que, tous les clouds sont transformés en leurs correspondants clouds réduits, qui sont stockés dans un ensemble réduit de réduction de clouds. Dans le même temps, chaque cloud réduit C' a un coût de cloud SN(C').

La troisième étape consiste à trouver une combinaison sous-optimale de clouds en utilisant l'ensemble réduit de clouds et les coûts de clouds réduits. Ils implémentent un algorithme approximatif en considérant le coût réduit du cloud dans le processus de sélection de clouds. Chaque cloud réduit doit être vérifier, si le nombre de ses fichiers de services inclus est plus grand que le cloud réduit maximum actuel, alors il est sélectionné comme cloud réduit maximum actuel. Pendant ce temps, si le nombre de fichiers de services est égal au cloud réduit maximal actuel.

3.4. Conclusion

Récemment, le terme cloud computing est largement utilisé dans la communauté de recherche, qui montre l'importance donnée par les scientifiques à ce domaine de recherche. Le cloud computing est un nouveau modèle informatique qui fournit la possibilité de partager des

ressources et des données en se basant sur un modèle de livraison de services, tels que des infrastructures, des plates-formes, et des logiciels sont offrent à l'utilisateur un ensemble de services.

On a commencé ce chapitre par une présentation détaillée du problème traité dans cette thèse, suivi par une étude d'un ensemble de travaux de recherche qui essayent de résoudre ce problème. On a remarqué que le sujet a attiré beaucoup d'attention, c'est pour cela on a trouvé que les chercheurs ont tenté plusieurs méthodes et paradigmes. Certains ont préféré utiliser le paradigme des agents, et des autres utilisent la programmation linéaire, sans oublier les travaux qui sont fondés sur l'optimisation combinatoire et les méta heuristiques. Le tableau 3 représente une récapitulation avec une comparaison entre un ensemble de travaux qui traitent le problème de composition de services. On a remarqué que les recherches ciblent l'optimisation du nombre de cloud bases impliqués dans le processus de composition, ou la satisfaction de l'utilisateur en terme de QoS. On a rajouté d'autres critères de comparaison comme l'année de la proposition, le langage de programmation utilisé pour l'implémentation.

Tableau 3 Comparaison entre les approches de composition de services

	Année	Optimisation de cloud bases	Basé sur les QoS	Basé Agent	met heuristique	Implémentation
COM2 [32]	2015	X			X	Inconnu
Gready-WSC et ACO-WSC [33]	2015	X			X	C++ et Matlab
AGAGA-WSC [35]	2010		X		X	Inconnu
SCM-CCA [28]	2014	X		X		Java (NetBeans)
GP-based approach [36]	2014		X		X	Inconnu
DGBAC Algorithme [37]	2014		X		X	Matlab
WSC Middleware [38]	2004		X		X	Java
Anouer B et Hela H [29]	2015		X	X		Inconnu

Chapitre 4

Problèmes de composition de services web dans le cloud

Critique des approches proposées et proposition d'améliorations

4.1. Partie 1 : Modélisation de problème de composition de services

4.1.1. Introduction

Après l'études des approches et méthodes existantes qui traitent le problème de composition de services web dans le cloud, on a remarqué que les chercheurs ont entamé le problème de deux directions différentes : la première direction vise la minimisation du nombre de cloud bases impliqués dans le processus de composition. On a expliqué dans cette thèse que le service composé est un assemblage d'un ensemble de services, chaque service dans l'ensemble eux réside sur une cloud base. Donc le service composé à son tour est distribué sur un ensemble de cloud bases, et on a expliqué que la minimisation de nombre de cloud bases où ces services sont distribués améliorera le fonctionnement de ce service composé et augmentera la satisfaction de l'utilisateur.

La deuxième direction consiste à composer un service optimal en terme des critères de QoS, pour satisfait le désire de l'utilisateur. On a discuté le concept de critères de QoS, qui sont un ensemble de critères qui caractérisent le fonctionnement d'un service web. Par conséquence, le service composé va avoir un ensemble de critères de QoS qui sont représentés par la composition des critères de QoS qui constitués le service composé, et on doit assurer que l'ensemble des critères de QoS pour le service composé soit quasi-optimal.

On a utilisé un ensemble de techniques pour proposer nos approches dans ce domaine. Certaines techniques sont classiques, très connues, et n'ont pas besoins d'une explication détaillée comme la programmation linéaire et la technique *branche & bound.* D'autre sont des techniques récentes et ne sont pas très utilisées par les chercheurs. C'est pour cela elles ont besoin d'une explication détaillée dans cette thèse comme l'algorithme Intelligent Water Drops (IWD)

4.1.2. Le processus standard de l'algorithme Intelligent Water Drops

L'algorithme Intelligent Water Drops (IWD) est une nouvelle technique d'optimisation inspirée de la nature, qui a été proposée par Hamed Shah-Hosseini dans 2007. L'idée de l'algorithme tient de l'observation du comportement de l'eau dans une rivière : les gouttes d'eau qui se déplacent le sol dans le lit de la rivière pour créer un chemin ; ce chemin est pris ensuite par les gouttes d'eau qui arrivent après. L'algorithme IWD travaille sur un graphe G=(N,E) (N est un ensemble de nœuds, E est un ensemble d'arcs) représente le sol de la rivière, et les gouttes

d'eau voyagent d'un nœud à un autre en passant par les arcs du graphes pour trouver finalement un chemin optimal [39] [40].

Les paramètres de l'algorithme sont de deux types : le premier type est un ensemble de paramètres constants durant tout l'algorithme nommé les paramètres statiques, Le deuxième type est un ensemble de paramètres qui change de valeurs pour chaque itération nommé les paramètres dynamiques, les étapes de l'algorithme définies par l'auteur sont constituées par un processus standard comme suit [39] :

1. Initialisation des paramètres statiques : le graphe *(N,E)* du problème est initialise, la fonction de qualité $q(T^{TB})$ de la totale meilleure solution est initialisée par la plus mouvais valeur = -∞. Le maximum nombre d'itération $iter_{max}$ est défini et le compteur d'itération $iter_{count}$ est initialisé par zéro. Le nombre de gouttes d'eau est souvent égal au nombre de nœuds dans le graphe. Les paramètres pour le changement de *Velocity* (vitesse par laquelle voyage une goutte d'eau) sont $a_v = 1, b_v = 0.01\ and\ c_v = 1$, et pour le changement de soul sont $a_s = 1, b_s = 0.01\ and\ c_s = 1$. Finalement, le soul initial (InitSoil) et l'initial *velocity* (InitVel), sont donnés par l'utilisateur.

2. Initialisation des paramètres dynamiques : chaque IWD a une liste de nœuds visités, qui est initialement vide. Aussi la *velocity* et la quantité de sol pour chaque IWD sont initialisées par *intivel* et *intisoil* respectivement. Et pour chaque arc (i; j) dans le graphe, tel que i et j sont des nœuds le soil(i; j) = *intisoil*.

3. Distribuer les IWDs aléatoirement sur les nœuds de graphe.
4. Modifier la liste des nœuds visités pour chaque IWD en ajoutant le nœud juste visite.
5. Répéter les étapes de (a) jusqu'a (d) pour les IWDs avec une solution incomplète :
(a) Pour l'IWD qui est déjà au niveau de nœud i, choisir le nœud suivant qui ne viole aucune contrainte du problème, et qui n'est pas dans la liste des nœuds visités selon la probabilité suivante :

$$P_i^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \in V_c(IWD)} f(soil(i,k))} \quad \text{Tel que}$$

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))} \text{ et} g(soil(i,j)) \begin{cases} soil(i,j)\ if\ min_{l \in V_c} & (soil(i,l)) \geq 0 \\ soil(i,j) - min_{l \in V_c} & (soil(i,l))\ else \end{cases}$$

(b) Si un IWD se déplace d'un nœud i à un autre j, de modifier sa velocity par :

$$Vel^{IWD}(t+1) = Vel^{IWD}(t) + \frac{a_v}{b_v + c_v + soil^2(i,j)}$$

ou V elIWD(t + 1) est le nouveau velocity pour ce IWD.

(c) Pour un IWD qui se déplace de i vers j, de calculer la quantité de soul qui le déplace

$\Delta soil(i,j) = \frac{a_s}{b_s + c_s.time^2(i,j;Vel^{IWD}(t+1))}$ tel que

$$time^2(i,j;Vel^{IWD}(t+1)) = \frac{HUD(j)}{Vel^{IWD}(t+1)}$$

Telle que la fonction heuristique est définie selon le problème traité.

(d) A ce point, le soul sur le chemin pris par l'IWD, et aussi le soul de l'IWD elle-même doit être modifie avec les équations :

$$soil(i,j) = (1 - p_n) * soil(i,j) - p_n \Delta soil(i,j)$$
$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j)$$

6. Trouver la meilleure solution pour la courante itération en utilisant :

$$T^{IB} = \arg max_{\forall T^{IWD}} q(T^{IWD})$$

ou la fonction q donne la qualité de solutions.

7. modifier le sol sur le chemin qui constituée la meilleure solution pour cette itération :

$$soil(i,j) = (1 + P_{IWD}) * soil(i,j) - P_{IWD} * \frac{1}{(N_{IB} - 1)} * soil_{IB}^{IWD} \; \forall \, (i,j) \in T^{IB}$$

ou N_{IB} est le nombre de nœuds dans la solution T^{IB}

8. modifier la totale meilleure solution en comparant l'ancienne solution avec la solution trouvée dans cette itération

$$T^{TB} = \begin{cases} T^{TB} \; if \; q(T^{TB}) > q(T^{IB}) \\ T^{IB} \; otherwise \end{cases}$$

9. incrémenter le compteur des itérations

$Iter_{cou\ nt} = Iter_{cou\ nt} + 1$ allez à l'étape 2 si $Iter_{cou\ nt} < Iter_{max}$

10. L'algorithme se termine ici avec la totale meilleure solution TTB

L'organigramme en figure 24 résume le processus standard de l'IWD algorithme.

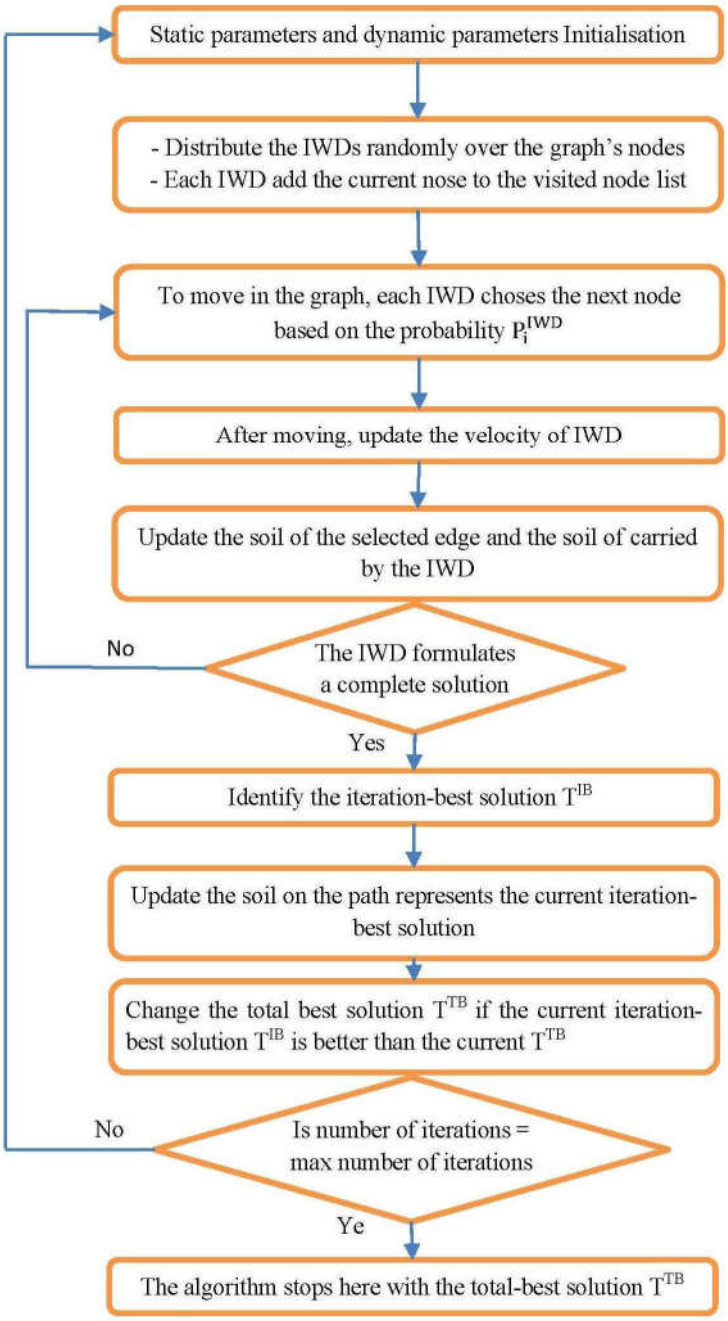

Figure 24 le processus standard de IWD algorithme

4.1.3. Contribution : La sélection d'un cloud combinaison

Un service web est une application modulaire définie et invoquée par les standards technologies du web [41] comme l'XML. Dans la plupart des cas, le service web est simple et atomique et ne peut pas satisfaire la demande d'un utilisateur. C'est pour cela, les services doivent être composés pour construire un service capable de répondre à la demande de l'utilisateur.

La figure 25 montre qu'il y a deux composants (combinateur de cloud, sélectionneur de services) très importants dans ce mécanisme. Les deux composants s'introduisent dans le processus de composition, et l'amélioration de ce processus passes par un changement au niveau de ses composants. Le processus commence par l'introduction de la requête de l'utilisateur, ensuite le cloud combinateur équipé par un algorithme d'optimisation combinatoire sélectionne un cloud combinaison approprié parmi les cloud bases disponibles

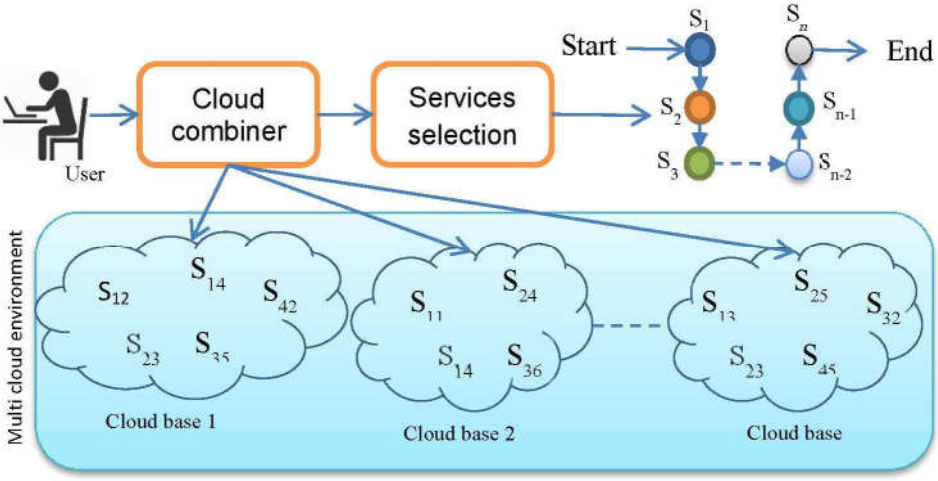

Figure 25 Processus de composition de services

Comme le montre la figure 25, il existe plusieurs services avec la même fonctionnalité sur différent cloud bases. L'action de choisir ce service implique implicitement le choix de la cloud base ou réside ce service. En d'autre termes, si la requête de l'utilisateur est d'exécuter la suite des taches (t_1, t_2….t_n), et pour exécuter une tache j, il y a un nombre de services candidats (S_{1j},S_{2j} …S_{nj}) tels que ces services offrent la même fonctionnalité mais sont localisés sur des cloud bases différents. Dans le processus de composition de services, il est nécessaire de sélectionner un nombre minimal de cloud bases. Par exemple si nous avons cinq cloud bases CB_i(i = 1…5) tels que CB_1 = [S_1, S_3], CB_2 = [S_3, S_2], CB_3 = [S_3, S_4], CB_4 = [S_1, S_2, S_5], CB_5 = [S_4] et la requête de l'utilisateur est [S_1,S_2,S_3,S_4]. L'ensemble des cloud combinaison pour couvrir la requête de l'utilisateur est : {{CB_1, CB_2, CB_3}, {CB_1, CB_2, CB_5}, {CB_1, CB_4}, {CB_2, CB_3, CB_4}, {CB_2, CB_4, CB_5}}. Parmi ces combinaisons, sauf qui ont deux éléments sont considérées comme des combinaisons optimales.

Architecture globale de l'approche proposée

Le processus de service composition dans un environnement multi cloud implique implicitement la sélection d'un ensemble de cloud bases. Parce que chaque service réside forcément sur un cloud base, la figure 26 présente cette coloration entre la composition de services et la sélection d'un cloud combinaison :

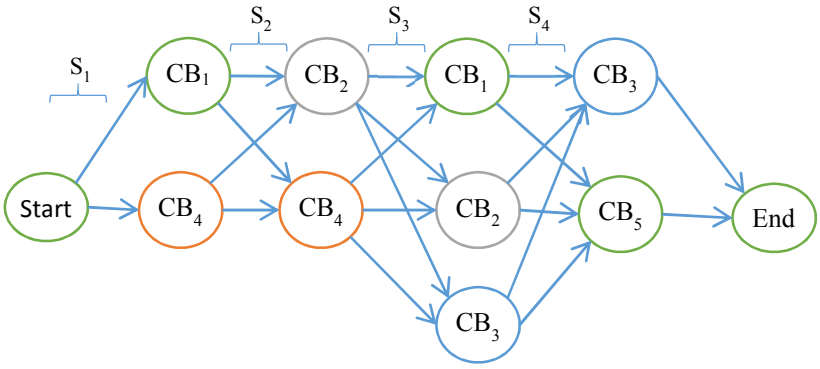

Figure 26 La sélection d'une cloud combinaison durant la composition de services

Chaque fois qu'on sélectionne un service, un cloud base est sélectionné en même temps, c'est-à-dire que dans le pire des cas, le nombre de cloud bases sélectionnés est égale au nombre de services.

Généralement, la requête de l'utilisateur passe par un ensemble d'étapes, avant de retourner les résultats au demandeur. La figure 27 présente ces étapes, dont laquelle la requête passe premièrement par une étape de décomposition pour identifier l'ensemble des taches qui constituent la requête. Ces taches sont des opérations simples et atomiques et supposé d'être exécutés par un ou plusieurs services, en suite pour chaque tâche, il y a un ensemble de services qui assure l'exécution de cette tâche, ces services sont désignés dans l'étape de l'identification de services, pour créer des relations de type un à plusieurs entre les taches et les services. Durant l'étape de l'identification des bases de cloud, et pour tout service déterminé à l'étape précédente, une collection de bases de cloud est identifiée, tel que ce service appartient à ces bases de cloud. La dernière étape, consiste à la sélection d'une combinaison de bases de cloud optimale qui regroupe l'ensemble des services qui assurent la satisfaction de l'utilisateur.

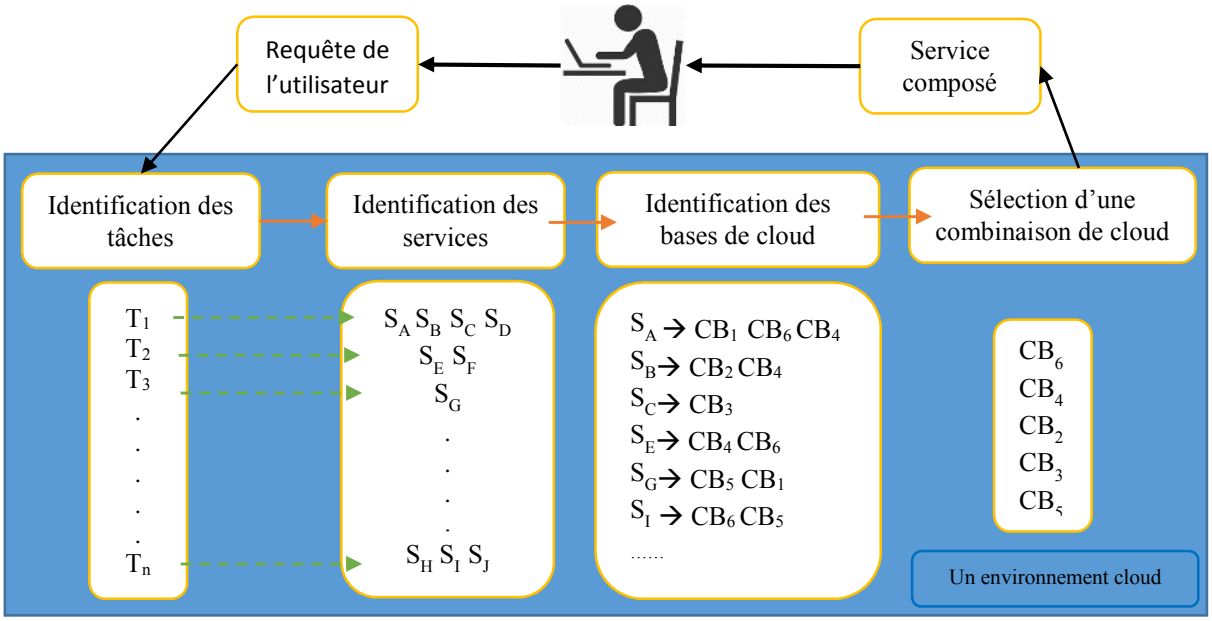

Figure 27 Architecture globale de l'approche proposée

Diagramme de composants

Dans le langage de modélisation unifiée, un diagramme de composants décrit la manière dont les composants sont câblés ensemble pour former des composants plus importants ou des systèmes logiciels. Ils sont utilisés pour illustrer la structure des systèmes arbitrairement complexes. Un composant est une ressource requise pour exécuter une fonction de stéréotype. Les exemples de stéréotypes dans les composants comprennent les exécutables, les documents, les tables de base de données, les fichiers et les fichiers de bibliothèque [42].

Les composants sont câblés ensemble en utilisant un connecteur d'assemblage pour connecter l'interface requise d'un composant avec l'interface fournie d'un autre composant

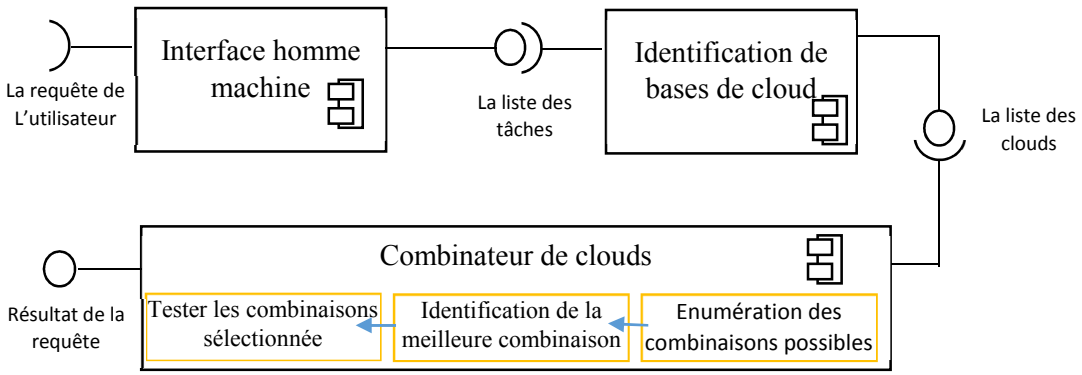

Figure 28 Diagramme de composants de l'architecture proposée

La figure 28 montre le diagramme de composants pour notre approche. On voit clairement que le composant nommé *combinateur de clouds* représente le cœur de fonctionnement pour notre approche. Ce composant est constitué de trois classes : une classe pour énumérer l'ensemble des solutions possibles, une classe pour choisir la meilleure solution et une dernière classe pour tester la solution sélectionnée.

4.1.3.1. Modélisation mathématique du problème de composition de services

La recherche d'une cloud combinaison pour couvrir la requête de l'utilisateur est identique avec le problème classique de mathématique nommé *set covering problem.* Il est considéré comme une question de mathématique combinatoire [43], dont lequel il existe un ensemble nommé l'univers U = {u_1, u_2…..u_n} qui représente la requête de l'utilisateur dans notre problème original, et *m* sous-ensembles de U : S_1,S_2….S_m tel que l'union de tous les S_i = U, ces S_i représentent les cloud bases dans notre problème.

Le problème de set covering consiste à trouver le plus petit nombre de ces sous-ensembles tel que leur union égale à l'univers U. Cet énoncé est identique avec la description de notre original problème avec une petite différence est que l'on cherche le plus petit nombre de cloud bases tel que leur union inclus l'univers et ne pas exactement égale. Malgré cette différence, les méthodes, les techniques et toute la base théorique utilisées pour résoudre le problème set covering peuvent être utilisés pour trouver un nombre optimal de cloud bases pour satisfaire les besoins des utilisateurs.

4.1.3.2 Modélisation du problème à base de graphe

Le problème de composition de services peut être vu d'une manière abstraite comme un graphe indirecte complet (cf. figure 29) tel que *G = (CB, E)* où :

$CB = \{CB_1, CB_2 \cdots CB_m\}$ est un ensemble de cloud bases et représente les nœuds du graphes.

$E = \{(CB_i, CB_j) tel\ que\ CB_i\ et\ CB_j\ \in CB\ , (i \neq j)\}$est un ensemble d'arcs.

Tel que chaque cloud base (nœud) regroupe un ensemble de services $CB_i = \{S_1, S_2 \cdots S_n\}$. La figure 29 montre la modélisation à base de graphes pour l'exemple donnée. Les cinq cloud bases sont modélisés comme des nœuds, et l'existence d'un arc entre deux nœuds (CB_1, CB_2) signifie que si a un instant quelconque nous avons choisis le cloud base représenté par CB_1, il est possible de choisir le cloud base représenté par CB_2 dans la prochaine étape.

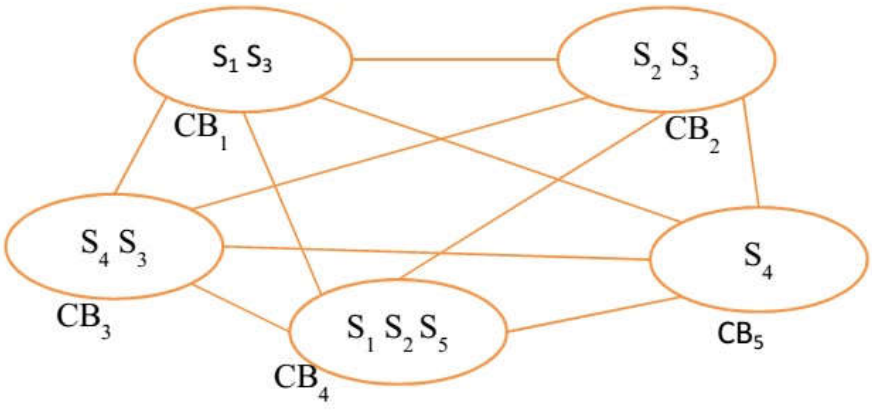

Figure 29 Modalisation du problème de composition à base de graphes.

4.1.3.3. La composition des services comme un programme linéaire

La programmation linéaire en nombres entiers est un outil mathématique pour modéliser et résoudre les problèmes d'optimisation, dont laquelle la fonction objective est une fonction linéaire et les variables de décision prennent des valeurs entières.

Dans notre problème, la décision est autour de sélectionner ou non un cloud base. C'est pour cela, chaque cloud base CB_i est représenté par une variable de décision entier Xi, et la fonction objective sera de minimiser la somme de ces variables. Pour chaque service si qui est dans la requête de l'utilisateur, au moins un cloud base dont lequel ce service réside est sélectionné, c.-à-d. la somme des variables de décision pour les cloud bases où ce service existe doit être supérieur ou égale à 1 (voir l'équation) :

$$minmiser \quad \sum_{i=1}^{m} x_i$$

$$Sachant\ que \quad \sum_{i=1}^{m} Ax_i \geq 1$$

$$x_i \in \{0,1\} \quad i = 1 \cdots m$$

La matrice A représente la distribution des services demandés par l'utilisateur sur les cloud bases, tel que l'élément a_{ij} signifie que le service S_j est un élément de cloud base CB_i. Pour l'exemple, nous avons cinq cloud bases $CB_1 = [S_1, S_3]$, $CB_2 = [S_3, S_2]$, $CB_3 = [S_3, S_4]$, $CB_4 = [S_1, S_2, S_5]$, $CB_5 = [S_4]$ et la requête de l'utilisateur est $[S_1, S_2, S_3, S_4]$, le problème peut être modélisé comme un programme linéaire en nombre entiers, comme suit :

$$\begin{array}{ll} minmiser & x_1 + x_2 + x_3 + x_4 + x_5 \\ & x_1 \quad + x_4 \geq 1 \\ & \quad x_2 \quad + x_4 \geq 1 \\ Sachant\ que & x_1 + x_2 + x_3 \quad \geq 1 \\ & \quad x_3 + x_4 + x_5 \geq 1 \\ & x_i \in \{0,1\} \end{array}$$

4.1.3.3.1. La relaxation de problème et la borne inferieure

Pour décider quels sous-problèmes sont intéressants à résoudre (ou à conquérir) et lesquels doivent être ignorer, nous devons identifier une limite inferieure pour notre problème. La technique la plus connu pour calculer la limite inferieure est la relaxation du problème, où on supprime la contrainte qui exige que toute variable de décision doit être un nombre entier. Ce processus transforme le programme linéaire en nombres entiers (ILP), à un problème linéaire (LP). La relaxation de l'ILP donnée comme exemple est comme suit :

La résolution du programme linaire présenté comme exemple, donnera la solution avec $x_1 = 1/3,\ x_2 = 1/3,\ x_3 = 1/3,\ x_4 = 2/3,\ x_5 = 0$ et $\sum x_i$ (i=1..5)= 5/3, la valeur 5/3 représentent la limite inferieure que nous ne pouvons pas obtenir mieux qu'elle avec des valeurs entières. Alors nous commençons le processus de ramification en arrondissant la valeur d'une variable de décision pour être un entier et calculer à nouveau la relaxation du problème. L'ensemble des étapes du processus sont comme suit :

- Si le problème n'a pas de solution possible, arrêter la ramification.
- Si le problème a une solution entière, arrêtez le branchement et comparer cette solution avec la meilleure solution jusqu'à présent.
- Si le problème a une solution, mais il n'est pas mieux que l'autre solution trouvée avant, arrêter le branchement
- Si le problème a une solution mixte (valeurs entières avec des valeurs réelles) mieux que les autres solutions, commencer le branchement et répétez.

4.1.3.4 L'algorithme IWD pour la sélection de cloud combinaison

En plus du processus standard de l'algorithme IWD présenté entièrement, l'algorithme prend aussi comme entré, des entrés en relation avec notre problème : une liste représente la requête de l'utilisateur L_{req}, et la liste des cloud base, en fixant les services disponibles pour chaque base.

Ensuite l'algorithme génère le graphe (N,E), tel que chaque nœud représente un cloud base, et pour chaque deux nœuds Ni et N_j , l'arc (N_i;N_j) appartient à E et ($i \neq j$). C.-à-d. que le graphe généré est complet, et qu'à partir d'un nœud quelconque, il est possible de visiter n'importe quel autre nœud.

Les IWDs se déplacent d'un nœud à un autre (d'une cloud base à une autre) pour tenter de satisfaire la requête de l'utilisateur. Chaque fois qu'une water drop visite un cloud base, elle supprime les services disponibles sur cette base de la requête liste, et elle continue le voyage jusqu'à ce que la requête liste sera vide. La prochaine cloud base est sélectionné selon une fonction de probabilité P^{IWD}_i (j) ; cette fonction sert comme un moyen pour un IWD résidant dans le nœud i, de choisir le prochain nœud j, qui ne viole pas les contraintes du problème et n'est pas dans la liste des nœuds déjà visités Vc(IWD) du IWD [44][45] (cf. Figure 30).

Input: - $CB = \{CB_1, CB_2 \cdots CB_n\}$ un ensemble de cloud bases
- L_{req} un ensemble représente la requête de l'utilisateur.

Output: - S_c l'ensemble de bases de cloud impliqué dans le processus de composition trouvé par l'IWD numéro i.
- T^{TB} La meilleur ensemble de cloud bases impliqué dans la composition trouvée par l'algorithme

Begin
Initialisation des paramètres statiques et dynamiques
while not atteint le nombre max d'itérations **do**
for i = 1 to nd (the i-th drop) **do**
Aléatoirement sélectionner une base de cloud CB_j
while ($L_{req} \neq 0$) **do**
if ($CB_j \cap L_{req}$) $\neq 0$ **then**
$L_{req} = L_{req} - CB_j$
$S_c = S_c \cup CB_j$
end
Sélectionner une base cloud CB_j en basant sur la probabilité dans 3
Modifier le velocity de l'IWD courant en utilisant l'équation 4
Modifier le soil de chaque arc du graphe en utilisant l'équation 5
Modifier the soil de l'IWD courant en utilisant l'équation 6
end
Modifier la totale meilleure solution (T^{TB}) si le résultat de ce IWD est mieux.
End
End

Figure 30 L'algorithme IWD adapté pour la composition de services

L'algorithme IWD est une série bien définie d'étapes. Les seules parties que nous devions les définir en fonction des caractéristiques de notre problème sont : la fonction heuristique HUD(x), la fonction objective f(X) et la fonction de la qualité q(.) [44].

Dans ce travail, un IWD se déplace d'un cloud base CB_i vers un autre CB_j parmi beaucoup d'autres choix ; si la cloud base (CB_j) contient un nombre maximal de services qui sont aussi dans la requête liste. c-à-d. le HUD(CB_j) = le nombre d'éléments communs entre la requête liste et la cloud base (CB_j), ou la cloud base avec une grande valeur pour HUD est considéré comme un candidat préférable à visite:

$$HUD(CB_x) = \left|CB_x \cap L_{req}\right|$$

Chaque IWD doit trouver une solution complète pour satisfaire la demande de l'utilisateur. La fonction mathématique qui modélise cet objectif, est que l'intersection entre la requête liste et l'union des services qui sont membre dans les cloud bases sélectionnés par le courant IWD doit être égale à la requête liste initial. Par exemple pour une solution quelconque $S = \{CB_{i1}, CB_{i2} \cdots\ CB_{in}\}$, cette fonction mathématique f(s) est :

$$f(S) = \left((\cup\ CB_x\ ou\ CB_x \in S\ ) \cap\ L_{req}\right) =\ L_{req})$$

Enfin, chaque IWD cesse de voyager dans le graphe si la fonction objectif est satisfaite. Dans ce cas, on peut dire que cet IWD a formulé une solution au problème, donc à la fin de chaque itération, nous aurons N_{iwd} solution, et pour comparer ces solutions, nous définissons une fonction de qualité q(S) qui prend la valeur de la cardinalité de S, ou S est une solution donnée (un ensemble de cloud bases) :

$$q(S) = |S|$$

4.1.5 Conclusion

Cette première partie de ce chapitre, représente le fruit de notre travail, dans lequel nous avons présenté notre contribution pour résoudre le problème de composition de services dans le cloud. Le chapitre commence par une présentation détaillée de l'algorithme Intelligent Water Drops (IWD), tel que on a présenté son processus standard comme il a été défini par son auteur. Ensuite on a commencé à exposer notre contribution qui concerne la sélection d'un cloud combinaison durant le processus de composition de service. On a proposé une architecture globale pour l'approche proposée, et un diagramme de composants pour montrer l'ensemble des composants qui composent notre approche.

Ensuite, on a passé à l'étape de modélisation, qui montre la relation entre le problème traité dans cette thèse et le problème classique dans les mathématiques appelé *set covering problem*. A la suite, on a proposé deux modélisations pour notre problème, la première

modélisation est basée sur la théorie des graphes et la deuxième formulée comme un programme linéaire en nombres entiers.

Pour les solutions, on a utilisé la technique Branch & Bound pour résoudre le problème modélisé comme un programme linéaire, et une adaptation de l'algorithme IWD pour résoudre le problème modélisé comme un graphe.

4.2. Partie 2 Discussion du résultat

4.2.1 Introduction

Dans la deuxième partie de ce chapitre, nous exposons l'ensemble des résultats trouvés durant notre recherche. Ces résultats sont obtenus en exécutant un processus de trois étapes. La première étape consiste en l'implémentation des algorithmes et techniques proposés. La deuxième étape est consacrée pour l'élaboration des benchmarks et des donnés pour les tests. Dans la dernière étape, elle est consacrée pour commenter, analyser, expliquer et comparer les résultats

Afin d'implémenter les deux approches proposées dans le chapitre précédent, nous avons utilisé un ordinateur avec le system d'exploitation Windows, alimenté par le processeur Intel Core i5, et le .net c sharp comme un langage de programmation, intégré dans l'environnement de développement Visual Studio 2010. Pour créer le modèle ILP et mettre en œuvre la technique *branche and bound*, nous avons utilisé le puissant langage de techniques de calcul MATLAB la version 7.1 sur la même machine.

4.2.2. Les outils de développement

Afin d'implémenter l'approche proposée, nous avons choisi d'utiliser un ensemble d'outils et d'environnements de développement. Dans ce travail, on va utiliser l'environnement de développement de Microsoft nommé visual studio et l'outil de calcul scientifique Matalb distribué par MathWorks. Le choix d'utiliser ces outils est d'un côté purement technique : parce que d'une part ils sont appropriés pour cette implémentation, et d'autre part, ça revient à mon expérience personnelle d'utiliser ces outils.

4.2.2.1. La plateforme .net et Microsoft visual studio

La plateforme .NET existe depuis 2002 avec sa version de la CLR (Common Language Runtime). Plusieurs langages de programmation sont développés et intégrés avec cette plateforme parmi lesquelles on cite : C++, C#, visual basic et Asp.net[46]. La plateforme .net est la première expérience de Microsoft avec le code manager, tel que tous ces langages sont interprétés dans un langage intermédiaire appelle Microsoft Intermediate Language (MSIL) qui

rendre les applications développées sous la plateforme .net exécutable sur plusieurs système d'exploitation.

Microsoft fournit un très bon environnement de développement qui supporte le plateforme .net et les langages cités en ci-dessus. Les solutions et les projets sont les conteneurs utilisés par Visual Studio pour héberger et organiser le code que vous écriviez dans l'EDI. Les solutions sont des conteneurs virtuels ; ils regroupent et appliquent des propriétés à travers un ou plusieurs projets. Les projets sont à la fois virtuels et physiques. En plus de fonctionner comme des unités organisées pour le code, ils mappent également un à un avec les cibles du compilateur. Autrement dit, Visual Studio tourne un projet en code compilé. Chaque projet aboutira à la création d'un composant .NET (tel qu'une DLL ou un EXE fichier) [47].

4.2.2.2. Matlab

MATLAB est un environnement intégré pour le calcul scientifique et la visualisation. Il est écrit principalement en langage C et C++. MATLAB est distribué par la société The MathWorks (voir le site www.mathworks.com). Son nom vient de MATrix LABoratory, car il a été initialement développé pour le calcul matriciel [48].

MATLAB est considéré comme un outil puissant dans la modélisation et la résolution des problèmes linéaires en nombres entiers. Parce qu'il dispose de plusieurs méthodes et fonctions prédéveloppées pour traiter ce genre de problème, comme la fonction *lingprog* qui résoud un programme linéaire [49].

4.2.3 Résultats de la contribution

Nous avons utilisé le langage C# pour implémenter l'algorithme IWD pour la sélection d'un cloud combinaison, et on a utilisé Matlab pour implémenter la solution basée sur la programmation linéaire. Pour les données expérimentales, nous avons choisi d'utiliser le test de services web par défaut qui est défini dans le package OWL-S xPlan [50], parce qu'il a été utilisé dans des travaux antérieurs et particulier dans [32] et [33]. Essentiellement il contient cinq services que nous les nommes symboliquement $(S_1, S_2, S_3, S_4, S_5)$. Ces services sont distribués sur quatre cloud bases, pour former une configuration de test. L'ensemble des configurations est indiqué dans le tableau 4, et concernent les services requis par l'utilisateur. La requête liste est la même pour toutes les configurations, telle que $L_{req} = (S_1, S_2, S_3, S_4)$.

Tableau 4 le test de services web par défaut défini dans le package OWL-S xPlan

Configuration	CB1	CB2	CB3	CB4
Configuration 1	(S_1, S_2, S_3)	(S_4, S_5)	(S_3, S_4)	(S_1, S_2, S_3, S_5)
Configuration 2	(S_1, S_2)	(S_3)	(S_2, S_5)	(S_1, S_4, S_5)
Configuration 3	(S_1, S_3, S_5)	(S_5)	(S_1, S_2)	(S_3, S_4)
Configuration 4	(S_2, S_3, S_5)	(S_3, S_4)	(S_1, S_2, S_3)	(S_4, S_5)
Configuration 5	(S_1, S_2)	(S_2, S_3)	(S_3)	(S_1, S_4, S_5)

Après une série d'expériences, l'IWD algorithme proposé montre qu'il est capable de trouver la meilleure solution après un certain nombre d'itérations. Ainsi que la technique branche and bound a trouvé la meilleure solution dans un délai raisonnable. Le tableau 5 montre les résultats détaillés pour les deux techniques. Nous avons choisi de présenter en parallèle les résultats des anciens travaux pour les comparer avec les nôtres :

Tableau 5 les résultats d'exécution en terme de cloud base sélectionné

Configuration	COM2 [32]	ACO-WSC [33]	IWD algorithm	Branch & Bound
Configuration 1	CB_4, CB_2	CB_1, CB_2	CB_1, CB_3	CB_1, CB_2
Configuration 2	CB_4, CB_2, CB_3	CB_1, CB_2, CB_4	CB_4, CB_2, CB_3	CB_4, CB_2, CB_3
Configuration 3	CB_1, CB_4, CB_3	CB_4, CB_3	CB_4, CB_3	CB_4, CB_3
Configuration 4	CB_1, CB_3, CB_2	CB_3, CB_4	CB_3, CB_2	CB_3, CB_4
Configuration 5	CB_2, CB_4	CB_2, CB_4	CB_2, CB_4	CB_2, CB_4

Malgré que les combinaisons de clouds trouvées par les deux techniques sont différentes, elles sont toutes optimales, la table 6 résume le nombre de cloud bases dans chaque combinaison de clouds trouvés par chaque algorithme :

Tableau 6 les résultats d'exécution en terme de nombre de cloud bases

Configuration	COM2 [32]	ACO-WSC [33]	IWD algorithm	Branch & Bound
Configuration 1	2	2	2	2
Configuration 2	3	3	3	3
Configuration 3	3	2	2	2
Configuration 4	3	2	2	2
Configuration 5	2	2	2	2

Afin de comparer les deux techniques de composition de services, on a mesuré le temps d'exécution en milliseconde. Les données du tableau 7 montre que le coût est acceptable pour les deux techniques avec une préférence pour la technique *branch and bound*. Il est important

de mentionner que dans la mise en œuvre de la technique de *branch and bound,* nous avons utilisé la fonction Matlab *linprog* pour calculer la relaxation du problème :

Tableau 7 comparaison entre les deux algorithmes pour le temps d'exécution

Configuration	Temps d'execution	
	IWD algorithme	branch and bound
Configuration 1	25.12	24.11
Configuration 2	28.09	27.00
Configuration 3	26.43	25.01
Configuration 4	25.67	25.07
Configuration 5	24.87	24.60

Pour prouver l'efficacité de l'algorithme IWD proposé, nous avons effectué une série de tests (dix tests), où le nombre de bases de cloud augmente régulièrement dans chaque test. Les travaux antérieurs présentés par Kurdi et al. (2015) et Yu et al. (2015) n'ont pas expliqué le comportement de composition de services avec une grande taille de données. Par conséquent, nous avons choisi de comparer l'algorithme utilisé dans notre travaille avec un autre algorithme, appelé Smart Cloud, présenté par Zou [51]. Dans l'algorithme Smart Cloud, l'ensemble des bases de cloud est modélisé comme un arbre, où son nœud racine représente un point de départ pour l'algorithme. Ce nœud racine a comme sous-noeuds pour chaque base de clouds dans l'ensemble des bases de cloud. Chaque nœud de base de clouds a un ensemble de sous-nœuds pour l'ensemble des services disponibles. Ensuite, l'algorithme explore l'arbre tout en essayant de trouver un quasi-optimal ensemble de bases de cloud pour satisfaire la demande de l'utilisateur.

La figure 31 montre les résultats de ce test. L'axe des x montre les dix tests de notre expérience, alors que le nombre de bases de cloud apparaît sur l'axe des y. Le graphique contient quatre courbes. La première avec des astérisques représente le nombre total de bases de cloud, la deuxième avec des marques plus représente le nombre de bases de cloud divisé par deux, la troisième avec des marques carrées représente le résultat de l'algorithme Smart Cloud, et la dernière avec moins continu représente le nombre de bases de cloud sélectionnées par l'algorithme IWD dans chaque teste.

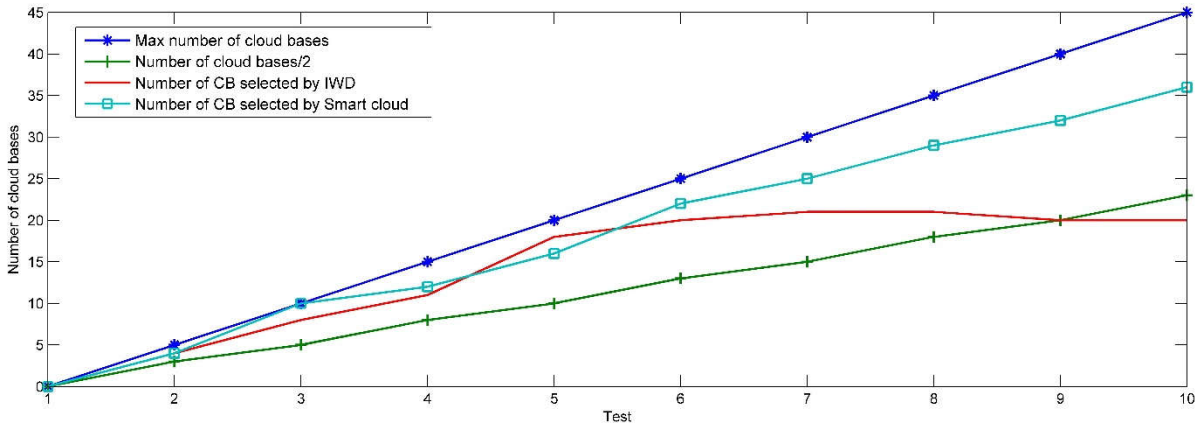

Figure 31 Performance de la algorithme proposé dans les grandes données

Le graphique représenté la figure 31 peut être divisé en deux parties, dans la première partie du test 1 à Test 5, les résultats présentés ici sont très proches, et nous ne pouvons pas voire beaucoup de différences entre la performance des deux algorithmes (IWD et Smart Cloud). Nous pouvons voir cela dans Test 3, l'algorithme IWD est meilleur que le Smart Cloud, et vice versa dans le test 5. Nous ne pouvons pas juger correctement les deux algorithmes car la taille des données n'est pas très grande. Cependant, la première partie montre que les résultats des deux algorithmes sont entre le nombre total et le la moitié du nombre de clouds.

Dans la deuxième partie (Tests 6-10), le nombre total de bases de cloud garde la même rythme d'augmentation, et nous observons clairement que le nombre de clouds sélectionnés par Smart Cloud augmente régulièrement. Cependant, le Smart Cloud donne un résultat acceptable, toujours inférieur au nombre total de bases de clouds. Concernant l'algorithme IWD, le nombre de bases de cloud sélectionnées augmente lentement, et la différence entre le total et les bases de clouds sélectionnés commence à être remarquable. Ce que nous pouvons voir dans les tests 9 et 10, que la méthode proposée montre sa meilleure performance et a éliminé plus de la moitié des bases de clouds disponibles.

4.2.4. Conclusion

En raison du dernier développement technologique, de nombreux concepts développés dans le passé, sont devenus des produits dans le présent. Parmi eux, le cloud computing est une technologie récente basée sur l'offre de services sur Internet. En conséquence, de nombreux services avec la même fonctionnalité deviennent disponibles sur Internet, malgré que ces services offrent la même fonctionnalité. Par conséquent, pour répondre aux demandes des

utilisateurs, les services doivent être composés pour former la composition optimale qui donne un maximum satisfaction aux utilisateurs

Dans ce dernier chapitre nous avons présenté l'ensemble des résultats expérimentaux durant notre recherche. Ces résultats sont comparés avec des travaux antérieurs et aussi avec les objectifs fixés au début de ce travail. Pour minimiser le nombre de cloud bases impliqués dans le processus de composition, nous avons modéliser chaque base de cloud comme un nœud dans un graphe ensuite de lancer l'algorithme IWD. Ce dernier dispose d'un ensemble d'IWDs qui voyagent dans le graphe pour trouver une combinaison optimale.

Conclusion générale

Conclusion général et perspectives

Le cloud computing est un nouveau paradigme informatique où les infrastructures, les plates-formes et les logiciels sont proposés comme des services, qui conduisent à l'augmentation du nombre de services disponibles sur le marché. De nombreux services parmi eux offraient les mêmes fonctionnalités, et comme remarquable résultat est que le choix de l'utilisateur sera multiplié. Dans un autre côté, dans la plupart des cas, la demande de l'utilisateur implique de nombreux services et pas seulement un seul. C'est pour cela ces services doivent être composés pour construire un service global afin de satisfaire la demande de l'utilisateur.

Dans cette thèse nous avons traité le problème de composition de services web dans le cloud computing. La thèse a été organisée en quatre chapitre, tel que on a détaillé les trois axes de recherches, les services web, l'architecture orientée services et le cloud computing dans le premier chapitre. L'objectif était de couvrir tous les concepts qui sont en relation avec la composition de services dans le cloud. Ensuite dans le deuxième chapitre, nous avons donné en détail la description de la composition de services dans le cloud, qui est le problème traité dans cette thèse, avec un état de l'art sur les travaux qui ont abordé le même problème. Nous avons essayé de choisir les travaux les plus récents, et l'objectif de cette étude est de la conclure par plusieurs contributions. Dans le troisième chapitre nous avons expliqué notre contribution pour la composition de services. Notre approche consiste à minimiser le nombre de cloud bases impliqués dans le processus de composition de services dans un environnement multi cloud. Le dernier chapitre dans cette thèse a été consacré pour présenter mes résultats durant notre recherche. Ces résultats sont présentés sous forme de courbes et de tableaux qui permettent de les analysés et de les commenter dans le même chapitre.

Notre contribution dans ce travail est à propos de la composition de services dans un environnement multi cloud. L'objectif est de minimiser le nombre de cloud bases impliqués dans la demande de l'utilisateur. Par conséquent, nous avons utilisé une technique d'optimisation inspirée de la nature, appelée l'intelligent water drops algorithme. Elle définit un ensemble de gouttes d'eau intelligentes qui circule sur un graphe pour représenter la modélisation du problème. Ses nœuds sont les cloud bases, pour trouver enfin une combinaison de clouds. Nous avons utilisé également, la programmation linéaire en nombres entiers pour la modélisation du problème et nous avons mis en œuvre la technique de *branche & bound* pour trouver une solution à ce problème. Malgré que les deux techniques sont prouvés d'être efficaces et peuvent trouver la meilleure solution avec un coût acceptable, notre étude donne une légère préférence pour la technique *branche & bround* pour la composition de services car elle est moins couteuse que l'IWD algorithme.

Perspectives :

Comme perspectives, nous avons l'intention de traiter quelques problèmes en rapport avec le sujet de cette thèse :

- Le premier problème c'est la composition de services web basée sur la QoS, L'objectif est la satisfaction de la demande de l'utilisateur pour mesurer en terme de propriétés de QoS.
- Le deuxième problème c'est la composition de services Web basée sur la qualité du service dans un environnement multi cloud. Les services sont caractérisés par un ensemble de qualités, et ils sont situés sur plusieurs cloud bases. Le but est de les composer de telle façon d'avoir la qualité de service maximale et un nombre minimal de cloud bases impliqués.
- Le troisième problème c'est la sémantique dans la composition de services web [52], où des informations sémantiques sont pris en considération au cours du processus de composition.

Le travail réalisé dans cette thèse est couronné par deux publications. Une communication orale dans *the International Symposium on Informatics and its Applications* (ISIA 2016) à l'université de M'sila, cette communication est intitulée *La sélection d'une cloud combinaison dans le processus de composition de service*. Une deuxième publication intitulé *Service composition in the multi cloud environment* dans le journal International Journal of Web

Information Systems de la maison d'édition Emerald (2017, Volume 13 Issue 4pp. 471 - 484)).

Bibliographie

[1] M. Bernichi, “Surveillance logicielle à base d’une communauté d’agents mobiles,” 2009.

[2] D. Booth *et al.*, “Web Services Architecture,” *W3C Note*, vol. 22, no. February, pp. 1--98, 2004.

[3] S. Dustdar and W. Schreiner, “A Survey on Web Services Composition,” *Int. J. Web Grid Serv.*, vol. 1, no. 1, pp. 1–30, 2004.

[4] “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.” [Online]. Available: https://www.w3.org/TR/2007/REC-wsdl20-20070626/. [Accessed: 24-Dec-2017].

[5] R. Nagappan, R. Skoczylas, and R. P. Sriganesh, *Developing Java Web Services*. 2003.

[6] J. Snell, D. Tidwell, and P. Kulchenko, “Programming Web Services with SOAP,” *Language*. p. xiii + 244, 2002.

[7] E. Newcomer, “Understanding Web Services: XML, WSDL, SOAP, and UDDI,” p. 332, 2004.

[8] “UDDI from Developer Tools.” [Online]. Available: https://docs.oracle.com/cd/E13173_01/alsr/docs20/help/dev_guide/pr.uddi.dev.html. [Accessed: 25-Dec-2017].

[9] P. O. Reilly, *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*. .

[10] M. Papazoglou, “Web Services: Principles and Technology,” *Technology*, p. 784, 2007.

[11] M. Anbazhagan and N. Arun, “Understanding quality of service for Web services,” *IBM*, 2002. [Online]. Available: https://www.ibm.com/developerworks/library/ws-quality/index.html. [Accessed: 30-Oct-2017].

[12] L. Maesano, C. Bernard, and X. Le Galles, *Services Web en J2EE et. NET: conception et implémentations*, Eyrolles. Eyrolles, 2003.

[13] I. Toma and D. Foxvog, “Non-functional properties in Web services,” *WSMO Deliv.*,

pp. 1–57, 2006.

[14] C. Grosan, A. Abraham, F. Biennier, and Y. Badr, “Enhancing web service selection by user preferences of non-Unctional features,” *Proc. - Int. Conf. Next Gener. Web Serv. Pract. NWeSP 2008*, pp. 60–65, 2008.

[15] D. S. Linthicum, *Praise for Cloud Computing and SOA Convergence in Your Enterprise*. 2010.

[16] F.-M. Xavier, G. Pascal, P. Guillaume, and R. Cyril, “SOA Le guide de l’architecte d’un SI agile,” *Paris, Dunod*, 2011.

[17] A. Arsanjani, “Service-oriented modeling and architecture,” *IBM Dev. Work.*, no. January, pp. 1–15, 2004.

[18] A. Karande, M. Karande, and B B Meshram, “Choreography and Orchestration using Business Process Execution Language for SOA with Web Services,” *Int. J. Comput. Sci. Issues IJCSI´11*, vol. 8, no. 2, pp. 224–232, 2011.

[19] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, p. 50, 2008.

[20] P. Mell and T. Grance, “The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology,” 2011.

[21] B. Furht and A. Escalante, *Handbook of Coud Computing*. 2010.

[22] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. 2011.

[23] R. Hill, L. Hirsch, P. Lake, and S. Moshiri, *Guide to cloud computing: principles and practice*. Springer Science & Business Media, 2012.

[24] The Open Group, “Cloud Cube Model - The Open Group,” *Jericho Forum*, no. April, pp. 1–7, 2009.

[25] “Web Services and Cloud Computing.” [Online]. Available: https://www.service-architecture.com/articles/cloud-computing/web_services_and_cloud_computing.html. [Accessed: 26-Dec-2017].

[26] A. Jula, E. Sundararajan, and Z. Othman, “Cloud computing service composition: A

systematic literature review," in *Expert Systems with Applications*, vol. 41, no. 8, Elsevier, 2014, pp. 3809–3824.

[27] E. M. D. Ingenieurs, *Vers une composition dynamique des Services Web : une approche de composabilité offline*. 2014.

[28] 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.

[29] A. Bennajeh and H. Hachicha, "Web Service Composition Based on a Multi-agent System," in *Software Engineering in Intelligent Systems*, Springer, 2015, pp. 295–305.

[30] J. O. Gutierrez-Garcia and K.-M. Sim, "Agent-based service composition in cloud computing," in *Grid and distributed computing, control and automation*, Springer, 2010, pp. 1–10.

[31] J. O. Gutierrez-Garcia and K. M. Sim, "Self-organizing agents for service composition in cloud computing," in *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010*, 2010, pp. 59–66.

[32] H. Kurdi, A. Al-Anazi, C. Campbell, and A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition," in *Computers & Electrical Engineering*, vol. 42, Elsevier, 2015, pp. 107–113.

[33] Q. Yu, L. Chen, and B. Li, "Ant colony optimization applied to web service compositions in cloud computing," in *Computers & Electrical Engineering*, vol. 41, Elsevier, 2015, pp. 18–27.

[34] G. Zou, Y. Chen, Y. Xiang, R. Huang, and Y. Xu, "AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing," *Proc. Int. Conf. Cloud Comput. Virtualization 2010 CCV 2010*, pp. 28–35, 2010.

[35] Z. Yang, C. Shang, Q. Liu, and C. Zhao, "A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm," *J. Comput. Inf. Syst.*, vol. 8, pp. 2617–2622, 2010.

[36] Y. Yang, M. Hui, and Z. Mengjie, "A Genetic Programming approach to distributed QoS-aware web service composition," *Evol. Comput. (CEC), 2014 IEEE Congr.*, pp.

1840–1846, 2014.

[37] 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.

[38] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[39] H. Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," in *International Journal of Bio-Inspired Computation*, vol. 1, no. 1–2, Inderscience Publishers, 2009, pp. 71–79.

[40] H. Shah-hosseini, "Intelligent water drops algorithm A new optimization method for solving," *Int. J.*, vol. 1, no. 2, pp. 193–212, 2008.

[41] V. Gabrel, M. Manouvrier, and C. Murat, "Web services composition: complexity and models," in *Discrete Applied Mathematics*, vol. 196, Elsevier, 2015, pp. 100–114.

[42] O. M. Group, "OMG Unified Modeling Language TM (OMG UML), Superstructure v.2.3," *InformatikSpektrum*, vol. 21, no. May, p. 758, 2010.

[43] Z. Z. Liu, Z. P. Jia, X. Xue, and J. Y. An, "Reliable Web service composition based on QoS dynamic prediction," *Soft Comput.*, vol. 19, no. 5, pp. 1409–1425, 2015.

[44] 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.

[45] B. Abdelbasset and K. Okba, "La sélection d'une cloud combinaison dans le processus de composition de services," in *the International Symposium on Informatics and its Applications ISIA 2016*, 2016, pp. 167–176.

[46] "Learn to Develop with Microsoft Developer Network | MSDN." [Online]. Available: https://msdn.microsoft.com/en-us/dn308572.aspx. [Accessed: 24-Dec-2017].

[47] J. Plenderleith, *Microsoft Visual Studio 2008 Programming*. McGraw-Hill, Inc, 2008.

[48] P. G. A Quarteroni, F Saleri, *Calcul Scientifique Cours, exercices corrigés et illustrations enMATLAB et Octave*. Springer, 2010.

[49] "MathWorks - Makers of MATLAB and Simulink." [Online]. Available: https://www.mathworks.com/?s_tid=gn_logo. [Accessed: 24-Dec-2017].

[50] M. Klusch and A. Gerber, "Fast composition planning of owl-s services and application," in *Web Services, 2006. ECOWS'06. 4th European Conference on*, 2006, pp. 181–190.

[51] G. Zou, Y. Chen, Y. Yang, R. Huang, and Y. Xu, "AI planning and combinatorial optimization for web service composition in cloud computing," in *Proc international conference on cloud computing and virtualization*, 2010, pp. 1–8.

[52] S. Bansal, A. Bansal, G. Gupta, and M. B. Blake, "Generalized semantic Web service composition," in *Service Oriented Computing and Applications*, Springer, 2014, pp. 1–23.