Web Services Discovery, Selection and Ranking Based Multi-Agent system in Cloud Computing Environment

Saouli Hamza[1], Kazar Okba[1], Benharkat Aïcha-Nabila[2], Amghar Youssef [2]
[1]Computer Science Department
University Mohamed Khider
07000, Biskra, Algeria
[2]Laboratoire d'InfoRmatique en Image et Systèmes d'information
Lyon, France
hamza_saouli, okbakazar@yahoo.fr, nabila.benharkat, youssef.amghar@insa-lyon.fr

ABSTRACT: *One of the most important operation in Internet, is the discovery of web services, because of the increasing use of this technology in the last years, as well as, Cloud computing technology is becoming the new way to conceive applications in the network, due to the capacity of treatment and storage offer by the Cloud based systems. In this work we propose, new web services discovery architecture based agents in open cloud computing federation. More accurately, we propose a new algorithm of comparison between the request of the client and the description of the Web services, this algorithm aim to select and rank the discovered web services to give more precision to the web services that the system return to the client. The proposed system is composed of two areas of Cloud. The first deals with Key words based research and the second supports the filtering of the found web services based on a new matching algorithm for web services selection. The results show the efficacy of the proposed architecture in the optimization of the quality of service (QoS) (the response time in our case) due to, on the one hand, our v our virtualization policy, and the use of mobile agents to maximize the exploitation of the Cloud computing resources.*

Keywords: Cloud Computing, Multi-Agent system, Discovery of Web Services, UDDI, Business Service, WordNet

Received: 1 May 2012, Revised 4 June 2012, Accepted 8 June 2012

1. Introduction

The discovery operation is the first and the most important step to use any web service, because the discovered web services can be used directly by the client or in composition with other web services to accomplish complex tasks. So a good choice of web services, conduct to a good results for the client, or a good composition between different web services.

Among emergent technologies currently Cloud computing is really booming these years, and IT actors are convinced that this technology is essential for distributed computing and will change the way of conceiving and creating of web applications.

Recently some researchers were interested in the development of approaches based on mobile agents for WS discovery. These approaches seem to be well adapted to shared resources on the Net, and appear to us as a good way for service discovery over the Cloud.

In this paper we propose a new service of web services discovery based on: "*Open Cloud Computing Federation Based on*

Mobile Agents" [1], this mean that we will propose:

• A New Cloud computing architecture associate to a New mobile agent system for discovery of web services, which benefit from the cloud computing and mobile agents characteristics,

• A New matching algorithm (algorithm of comparison and filtering) for web services selection and ranking, to obtain more precision with the discovered web services.

This paper is organized as follows: section 2 presents some related concepts with our system, section 3 describe certain related works to our system, section 4 explain the general discovery architecture of our system, section 5 present the algorithm of comparison and filtering, section 6 present the implementation of our system, the section 7 show the simulation results, section 8 is the conclusion and prospects of this work.

2. Background

2.1 Cloud Computing

Cloud computing is a new technology, which makes it possible to users (people or companies) to reach shared resources and infrastructures [2] on Internet, Cloud computing is also a business model which makes it possible to the users to benefit from a capacity of treatment and unlimited storage. The users can allocate the resources according to their needs [3].

2.2 Mobile Agents

A mobile agent is an entity which can migrate from a platform to another in order to carry out one or more preset tasks. Asynchronism, autonomy, auto-adaptability, as well as the reduction of communication costs, and faults tolerance [4], characterize the mobile agent technology, place it at the chief candidate of the most effective technologies for the deployment and the exploitation of the distributed applications, in particular on Internet.

2.3 OCCF and MABOCCF

Open Cloud Computing Federation (OCCF) is a concept proposed by several researchers, which consists to incorporate and to use several CCSPs (Cloud Computing Service Providers), to provide a uniform resource interface for the clients; the OCCF is based on some notions that can be a good base, to solve the problems of portability and interoperability between the CCSPs. The OCCF is advantageous compared to the other systems, in the following points: Unlimited Scalability, Availability of resources, and Democratization of the Cloud Computing market, Deploying application on multiple CCSPs, and Reduced the cost to the clients [1].

Mobile Agents based on Open Cloud Computing Federation (MABOCCF), is a new mechanism that allows the realizing of portability and interoperability, allowing an easy and inexpensive implementation of the OCCF.

The MABOCCF is composed of several Cloud computing regions - which can be a simple Cloud regions or CCSPs... etc. each region has its own resources, and it is composed of a set of real machines, which contain one or more virtual machines, each virtual machine must contain one or more locations of mobile agents, in addition each region must contain one or more Tasks managers, which represents the only access point to the Cloud computing region. The tasks manager is responsible of, disaster recovery, indexing resources, authentication, security, and fault tolerance.

MABOCCF allowed the combination between Cloud and Mobile Agents technologies, which make it possible to benefit from the advantages of these two last, to perform various tasks on the network such as: the composition of web services, security systems... Etc. and in our case it is the discovery and selection of web services.

The experiments results manifests that, the use of MABOCCF optimize the access to the resources over Internet by 50.35% compared with normal systems that don't support the portability between different CCSPs.

2.4 Web Services Discovery and UDDI

One of the most important technologies, used to describe web services is *UDDI* [5]. The Universal Description Discovery and Integration (UDDI) is an XML language that allows web services providers to give a dynamic description of their web services. In this work we are interested to Business services tag, which is used to give a textual description (human readable description) of web services; this tag is used by our matching algorithm to compare its contained to the client request.

The goal of the discovery and selection is to make the connection (or the mapping) between the request (which can be very individual and very specific) and the descriptions of web services (which are more objective and more precise). These discrepancies make more difficult the discovery operation [6].

3. Related works and problems

We have chosen some works on the discovery and selection of web services that have been published during the last years, which can illustrate the problems that we will address in our work.

In the paper [7] the authors proposed an architecture based on a set of situated and mobile agents; each situate agent has one of the following tasks: system management, request analyzing, update of the mobile agents, expeditions of the mobile agents, results analyzing. The mobile agents are dispatched to do a syntactic search of web services. The problem with this approach is that it misses, in the one hand, a powerful economic model binding to the discovery web services system, and on the other hand, the research based keywords (that the authors have mentioned in this work) does not take into account the disposition of the words in the description of the web services compared to the request and the repeated words in the request.

In [8] the authors proposed an approach based on WordNet for vector extracting (vector space model VSM), which allows, not only to add semantic information to this representation, but also reduces the size and space vectors. In addition the authors have presented a set of kernel-based methods to calculate kernel based similarity, these methods, very answered and very known in the field of the e-learning, are used to estimate the similarity between web services. This work suffers from three main problems. First, the VSM vector representation does not take into account sub strings. Second, the VSM vector representation does not take into consideration the arrangement off words in the request. Third, the application of the kernel-based methods requires, developer intervention, to determine some parameters, but nothing guarantees that this intervention will not negatively influence the results.

In [9] Aversa et al. proposed a grid architecture based on mobile agents associates with Web services technology in order to discover and access, in an optimal way, to mobile services and resources in the grid. The paradigm of Web services, SIP (Session Initiation Protocol) and UDDI technology used to implement a resource discovery service that allows users and mobile agents, to search and access to resources and applications distributed across heterogeneous terminals, by a dynamic configuration of the session of interactions and the functionalities of the service based on the characteristics of the terminal and the Quality of the interconnection. In [10] the authors proposed an architecture based mobile agents for the management of mobile services in Grid computing and described in detail the discovery service. The architecture is based on a hierarchical model of identical agents. Each agent is able to provide and request services in the grid and when it moves, certain processes of registration and deregistration are called, which makes the discovery faster and more effective. The problem with these tow last architectures is that the economic model of Grid computing suffers from the problems of instantaneity and availability of resources in the Grid, and especially from time of use, i.e. that a resource in the grid is not necessarily available when it is required but one must wait until it is free then one uses it for a determined time as a preliminary.

In [11] Wagener et al. proposed an approach Cloud computing based on "*open standard Extensible Messaging and Presence Protocol* (*XMPP*)" for bioinformatics, including the discovery, the asynchronous invocation, and the definition of the types of data in the service. XMPP Cloud services are discoverable without the need for an external register, the asynchronous invocation eliminates the need for ad hoc solutions, and input/output defined in the services allows the generation of customers on the fly without needing an external semantic description. The problem with this work is, in one hand, the absence of one tool (a tool like mobile agents): autonomous, auto-adaptable, and intelligent, that can replace this protocol, because it is more adapted to the nature of web services that are distributed on the Internet, for the research of the Web services, and in the other hand, the use of the protocols in the communication between the Cloud resources increase the network load, and makes it harder to upgrade the efficiency and security of the application.

In [12] Chen et al. proposed a model "S*ervice Registry on Cloud* (*SRC*)" which is an extension of the service model based on keywords, this application is deployed like a Cloud application. This model aims to discover and select the most appropriate service to functional and non functional requirements expressed in the client's request. SRC stores the semantic descriptors of web services and dynamic state feedback of the QoS (Quality of service) of web services as GFS (Google file system) files in the Cloud, and uses the mechanism of MapReduce to process files. Moreover, the SRC instances stores the ontologies, to calculate

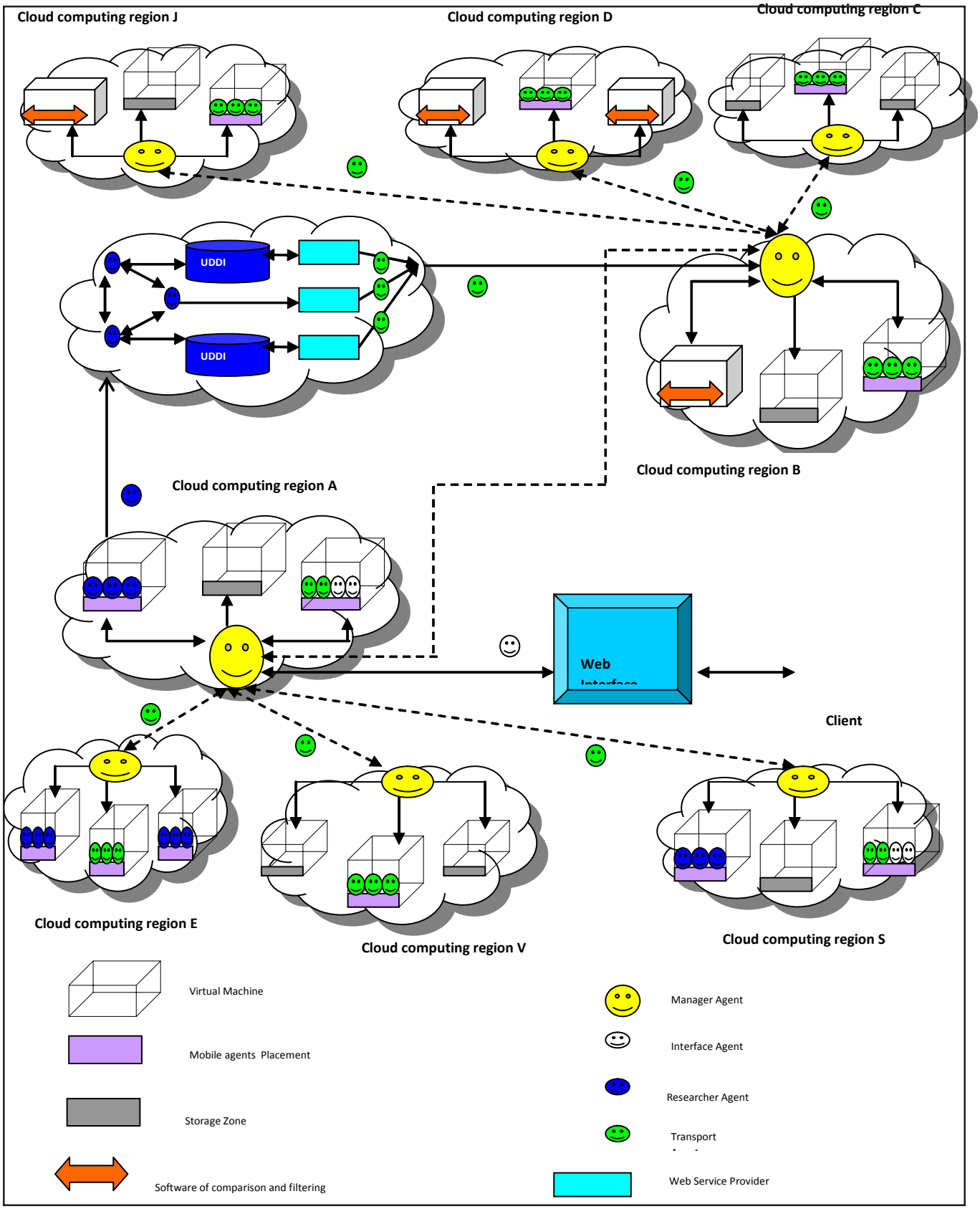


Figure 1. The general architecture

the similarity of the inputs/outputs of two services, and QoS Ontology which is used to describe the features of each QoS attribute. This work suffers from two main problems. First, the construction and maintenance of ontology's is very expensive, the lack of standards which make it possible to integrate and to re-use already existing ontology's and the lack of ontological descriptions, especially for Web services, makes the exploitation of ontology's more difficult. Second, there is no indication for eventual relations between the different SRC instances that can publish the WSs, to maximize the benefit from the Cloud resources in each SRC instance, the mobile agents technology can be a good solution for making these relations.

4. Web services discovery architecture

Our System is composed of four parts: Interface part, Cloud computing part, situated agents part and mobile agents part, all of those parts are integrated in a way to benefit from the capacities, of portability and interoperability (consequently it benefit from the advantages of the OCCF) offers by the OCCF and the implementation facilities offers by MABOCCF.

The figure 1 shows the general architecture of our discovery system.

4.1 Web interface part

The client access to the region A of Cloud via a web interface that allows it to:

1. Create an account to use our discovery and selection system;
2. Login to his account (in case of existing account);
3. Introduce his request;
4. Show results and billing.

4.2 Cloud Computing Part

4.2.1 Region A (RA)

Under the models Software-as-a-Service (SaaS) and Storage-as-a-service (SAaS), this region offers, in the one hand, the services of a software which allows creating and managing a mobile agents system (SaaS is integrate with the manager agent of the region A), which has for essential goal, the keywords based discovery of web services, by creating the researcher mobile agents which will deal with this task, moreover, this region deals with the creation of the Interface agents and the Transport agents in order to communicate with the clients and the region B of Cloud, and in the other hand, the storage of the request and its identifier.

Moreover, the datacenter(s) that contains the different Virtual machines of region A are always active, which guarantee the Instantaneity (instantiate access to resources) and the availability of the resources to the clients.

4.2.2 Region B (RB)

Under the same models of the region A, Software-as-a-Service (SaaS) and Storage-As-a-Service (SAaS), this region store the request and the web services descriptions found by researcher agents, then, its to the software offer as a service to select or reject the web services found by researcher mobile agents, the goal of the creation of this regions is to give more precision and to avoid the disadvantages of the syntactic research; when the selection of the web services is finish, the manager agent of this region create a transport agent (using the SaaS, which is integrate in the manager agent of the region B), to transport the selected web services to the region A.

Like the region A, the datacenter(s) that contain the different Virtual machines of regions B are always active, which guarantee the Instantaneity (instantiate access to resources) and the availability of the resources to the clients.

The reason to introduce two Cloud computing regions is to distribute calculates, as maximum as possible, in order to treat the requests of clients as soon as possible.

4.2.3 Regions C, D, F…:

The role of the other cloud computing regions (C, D, W.. etc) is to guaranty the massive scalability to our Cloud computing system; by using the transport mobile agents, in the case where regions A or B have the need for new resources, those two regions, can benefit from the resources of the other regions (regions C, D, F …etc), because this type of agents is responsible

for transporting the Client request or the discovered web services to be treated (stored or filtered), in the other Cloud computing regions, that offer the same services as regions A and B (it's the case of regions: S and J) or a part of services offers by regions A and B (the case of the regions: V, E, D and C).

4.3 Situated Agents part

4.3.1 Situated Manager Agent of region A (SMARA)

The situated manager agent of region A is responsible for the reception of the request, and the association of each request to an identifier (in the form of a code) that can identify each client (since clients can not be identified by their requests only). It extracts the key words from the request (nouns, verbs… etc) and create a set of researcher mobile agents to do the keywords based research of the web services. The Manager agent of the region A is composed of three components:

1. Knowledge base (KB): which allows the Updating of researcher agents knowledge, it also contains a dictionary, which is used to provide the synonyms and the derivations forms of each word of the request, to the researchers agents, the KB contain also, two directories, the first directory contain the addresses of UDDI registries that describe the web services, divided in a set of groups (each group contain a set of UDDI registries that are in the same Internet region). The second directory, contain the addresses of the other Cloud computing regions that provides the same (or a part) services as the region A, which guarantee the massive scalability to our system for this region (in the case of stop of the operations in the region A or needing more resources to accomplish the discovery operation, a transport agent is created to take the request of the client towards another region which offers the same services as the region A or more resources to this region).

2. Software-as-a-Service (SaaS): the SaaS is responsible of:

• Updating of the KB (Updating of the dictionary and the directories).

• Creation, updating and affecting the researcher mobile agents (the number of researcher mobile agents is equal to the number of the UDDI registries groups in the directory).

• Creation of the request identifier.

• Auto-provisioning of resources: it represents one of the most important characteristics of Cloud computing, it means that the system is responsible for finding resources when the discovery system needs them.

• Billing: our system flow the Cloud computing way to calculate the costs, it mean that the client pay only what he use.

• Updating of the software of the request treatment: the client doesn't need to pay, each time, for updating the new versions of the software used in the discovery operation, it's to the tasks manager to do this updating every appearance of new version.

• Creation of virtual machine: The virtualization is the most important character that gives to the Cloud computing his power and celebrity. Our Virtualization policy consist of maximizing the number of virtual machines and maximizing the decomposition of the global operation (the discovery operation) to elementary main tasks, then submitting each task to its VM (by this way our system benefit largely from the virtualization characteristic of Cloud computing technology). So, the task manager in region A is responsible of the creation of three Virtual machines that treats the three main tasks of this region, namely: Storage and key words extraction from the request, Creation of Interface and transport agents, Creation of Researcher agents.

• Disaster recovery.

• Security.

• … Etc.

3. Communication module (CM): the CM is responsible of the communication with the transport, interface and researcher mobile agents.

4.3.2 Situated Manager Agent of Region B (SMARB)

The manager agent of the region B has as function, the reception of the request, and the Web services descriptions found by the researcher agents. It transmits them to the software of selection, with the lemmas and derivation forms of each word of the request and descriptions of the discovered Web services. The manager agent of the region B is composed of three components:

1. Knowledge base (KB): it contain The WordNet dictionary (for using of the semantic synonyms, the derivation forms and the lemma, of each word in the request). the KB contain also, a directory, that contain the addresses of the other Cloud computing

regions that provides the same (or a part) services as the region B, which guarantee the massive scalability to our system for this region (in the case of the stop of the operations in the region B or needing more resources to accomplish the selection operation, a transport agent is created to take the request of the client towards another region which offers the same services as the region B or more resources to this region).

2. Software-as-a-Service (SaaS): the SaaS is responsible of:

• Creation of a transport agent which deals with the transport of the selected web services to the region A.

• Updating the dictionary and the directory of the KB.

• Creating a storage zone for storing each request and their corresponding discovered web services.

• Auto-provisioning of resources

• Billing: the client pay only what he use.

• Updating of the software of selection.

• Creation of virtual machine: The task manager is responsible of the creation of three Virtual machines that treats the three main tasks of this region, namely: Storage of web services descriptions, execution of the selection Algorithm, Creation of transport agents.

• … Etc.

3. Communication module (CM): the CM is responsible of the communication with the transport agents.

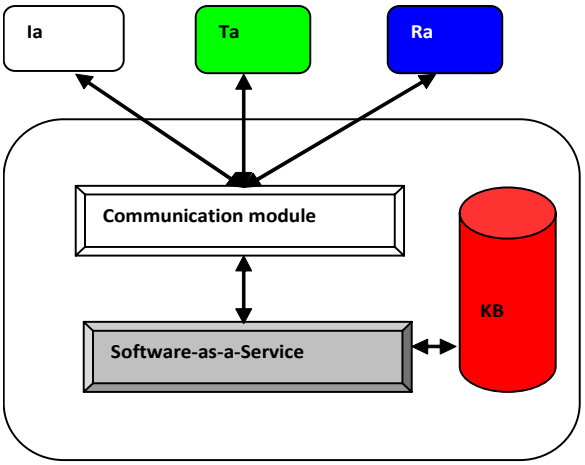


Figure 2. The manager agent of the region A

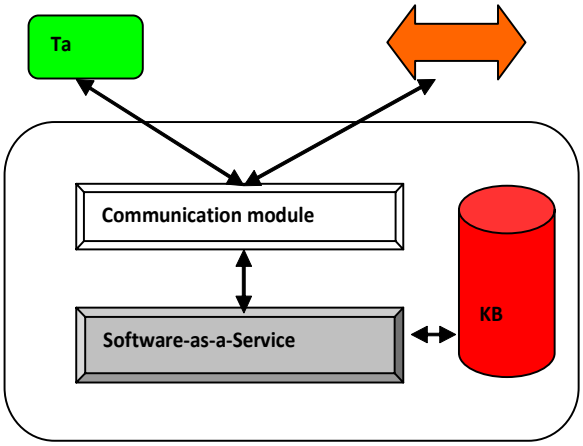


Figure 3. The manager agent of the region B

As conclusion to this part, we can say that our architecture, benefit largely from the cloud computing characteristics, namely: instantaneity, availability, scalability, …etc. but its not all; the virtualization, that our system benefit from, Reduced cost implementations of our system and offer the possibility, of accessing each Cloud resource by several users at the same time, we call this Cloud computing character: Multitenancy [13].

In the other hand, our cloud computing architecture avoids the disadvantages of the work in paper [12] because we relate the different Cloud regions using mobile agents, so we maximize the benefit from the resources of our system. Our architecture, present also a solution for Grid based discovery of web services by resolving problems mentioned in papers [9] and [10] especially for problems of instantaneity and availability of resources. Finally the economic model (which is absent in paper [7]) of our system allow client to pay only what he use.

4.4 Mobile agents part

4.4.1 Interface agent (Ia)

The interface agent is used to: transmit the client request to the region A, and display the billing and the selected web services to client via the web interface. This agent is composed of: (1) communication module with the client, (2) treatment module, (3)

communication module with the Manager agent of the regions A, and finally (4) data base to store the request and the billing.

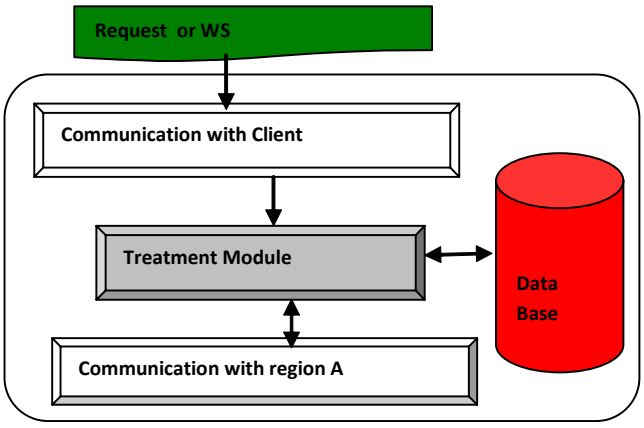


Figure 4. Interface Agent

4.4.2 Researcher agent (Ra)

The researcher agent is used to do the research based on keywords, this agent is composed of: (1) treatment module which is responsible of: the interrogation of UDDI registries, access to the discovered web services after the keywords search, and the creation of transport mobile agents to transport these web services and their descriptions to the region B, (2) inter-agents communication module , (3) communication module with the Manager agent of the region A, (4) data base to store the request and its identifier ,and finally (5) a base for storing web services.

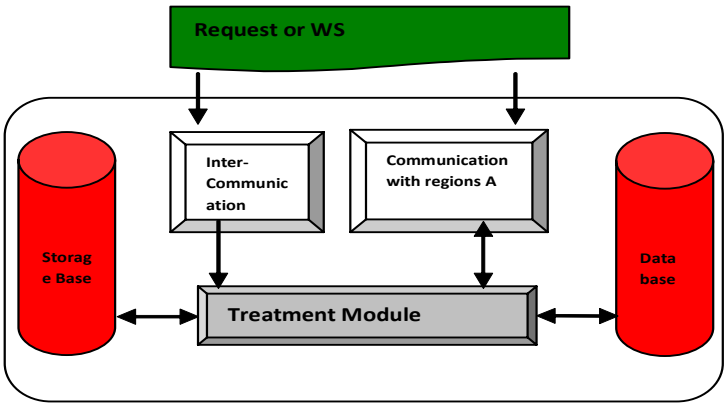


Figure 5. Researcher Agent

4.4.3 Transport Agent (Ta)

The transport agent can be used, on the one hand, to transport the client request and its identifier, from the region A to the region B, and on the other hand, has to transport the Web services discovered by the researcher mobile agents to the region B (with the identifier of the corresponding request), and then has to transport the Web services selected by the region B to region A (also with the identifier of the corresponding request). This type of agents are composed of (1) communication module with the Manager agent of the regions A and B, (2) treatment module, (3) inter-agents communication module, and finally (4) a base for storing web services.

As conclusion to this part we can say that our system has take advantage from the essential characteristics of mobile agents, namely: Autonomy, auto-adaptability and fault tolerance:

In case of panes (fault tolerance) in region A, the mobile agents decide (autonomy and auto-adaptability) to keep the client's request until the fault is set (the case of interface agents). The same, in the case of panes of one of the two regions, A or B, the transport agents, or in some identical case the researcher mobile agents, can keep: the client's request, discovered web services or the selected Web services. (When we say "*keep*" it means that the agent decides what to do with the information that it had,

since it is autonomous and can be adapted to different situations without returning to its owner to decide).

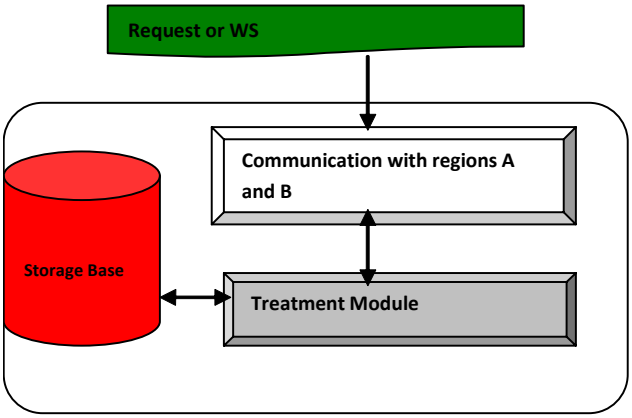


Figure 6. Transport agent

2.1 Contributions of the combination between mobile agents and Cloud computing Technologies

1. Unlimited scalability [14]: with mobile agents the resources in the cloud are used optimally, because each time the client request need more resources, a transport agent is create to search for these resources.

2. More intelligence: on one hand, the use of situated manager agent give more intelligence in the management of the cloud computing regions (in MABOCCF we only use a simple task managers) and in the other hand, the characters of autonomy and auto-adaptability of mobile agents give to our system more intelligence in the discovery operation (by researcher mobile agents) and in the research for more resources (by transport mobile agents).

3. The optimization and the reduction of the Consumption of the band-width: one of the most important problems of Cloud computing technology is the need of great ban-width, which will be enhanced by using transport agents in the communication between the different Cloud computing regions, or by using interface agents in the communication with clients.

4. Independent Computing: With the use of interface and transport mobile agents, the client or the Cloud computing regions can submit their requests or tasks then disconnect, then came back to retrieve the results, which allows, to the client or Cloud resources to be independent toward the other components of the Cloud network.

5. Robust and fault tolerance: the mobile agents have the ability to react dynamically to unfavorable situations and events makes it easier to build robust and fault tolerant distributed system, it's the case of transport agents, if one of the Cloud computing regions is being to shut down, the transport agents will take the tasks, that execute in this region, and continue to execute in another region that offer the same services.

6. Adapted to heterogeneous networks: because mobile agents are based on JAVA, it will be easy to use them in, networks which are based on heterogeneous hardware and software systems, its only question of installing (automatically) JAVA Virtual Machine (JVM) in each Cloud computing region, which enable mobile agents to transport and execute their tasks in these heterogeneous Cloud computing regions.

7. Encapsulation of protocols: the transport mobile agents, establish channels based on proprietary of the communication protocol between the Cloud computing regions, making easy to different Cloud computing services providers (that own the Cloud regions) to change or enhance (to get more efficiency or security) the communication mechanism without changing the architecture or the code of their local software's.

8. Reducing costs: The particular benefit of using the researcher mobile agents is the costs reduce of using resources for the client. Among the important characteristics of cloud computing is that the client pays only what he use as resources in the Cloud. So if we don't use the researcher agents (which are running on the server hosting the UDDI registries that describes the WSs) the manager agent of region A will itself creates the SOAP messages to query the UDDIs, which will increase the used resources and thus will increase the communication costs, then the client will be obliged to pay Moreover.

4.6 AUML Modeling

4.6.1 AUML for Modeling Multi-agent Systems

The UML is sometimes insufficient for modeling agents and agent-based systems. As response to this issue, an Agent-based Unified Modeling Language (AUML) is being developed. AUML is an extension of UML language, which enables the description of agents and agent-based systems. The goal of AUML is to be consistent with existing specifications of FIPA (Foundation for Intelligent Physical Agents) and OMG (Object Management Group) [15].

4.6.2 General Sequence Diagram

The scenario of use of our system is described by the following steps:

1. Send the client request via the web interface and launch the search.

2. An Interface agent is created at the region A to bring the request to this region, via the manager agent.

3. The manager agent of the region A creates:

- A storage zone, to store the request and its identifier.
- A set of researcher mobile agents, then transmits to them: the keywords of the request and their synonyms and their derivations forms and the request identifier, and the addresses of the UDDI registries from the same Internet region to each researcher agent (one researcher agent per Internet region).
- A transport agent for transporting the client request and its identifier, to the region B of Cloud.

4. The region B creates a storage Zone, to store the request and its identifier.

5. Launching of the researcher mobile agents in the Interne, to make the keywords based research.

6. Each time, a researcher mobile agent find a Web service that correspond to the request, it create a transport mobile agent which deals with transporting of the description of the web service (and the identifier of the corresponding request) to the region B via the manager agent, to be stored in the zone corresponding to the identifier transmitted with the web service description.

7. After a predetermined period of time, the researcher agents stop their Internet search.

8. The manager agent of the region B, launch the selection algorithm to select the best web services that correspond to the client request (For the rejected web services, the manager agent will delete them from the region B).

9. The manager agent of the region B calculate the costs of the resources used in the selection operation, then it create a transport agent which will transport the costs and the selected web services to the region A (associated with the request identifier).

10. The manager agent of the region A calculates the global cost.

11. The manager agent of the region A, create an Interface agent to display the selected web services and the Billing to the client via the Web interface.

5. Algorithm of comparison and filtering

5.1 Comparison and filtering mechanism

We present in this section a new mechanism of selection and ranking of web services based on WordNet, we use JUDDI [16] library to extract the information contained in the Business Service tag of the UDDI registry, and JAWS [15] library to add semantic information from WordNet database, Our matching algorithm is based on five main steps, (1) Pretreatment step (2) Exact comparison step (3) Approximate comparison step (4) Relative comparison step (5) decision step. The following figure show the general steps of our matching algorithm:

5.2 The Steps of the proposed Matching Algorithm

Our matching algorithm is based on the calculation of the degree of similarity between the request and the description of the Web services which one finds in the data structure Business Service, the idea behind the choice of the Business Service data structure, to be compared with the request, is that this last, is under textual format, so it will be more appropriate and more precise, to choose another textual format that describe the web services, to compare it with the request, we find this textual format only, in the data structure Business Service, so we suggest to web services providers to give more importance to this data structure and to also give a clear, textual description, to the UDDI that describe the Web service, because – as we will see with our algorithm, it's a very efficient way, to compare the client requests with web services descriptions.

The algorithm is based on 5 steps:

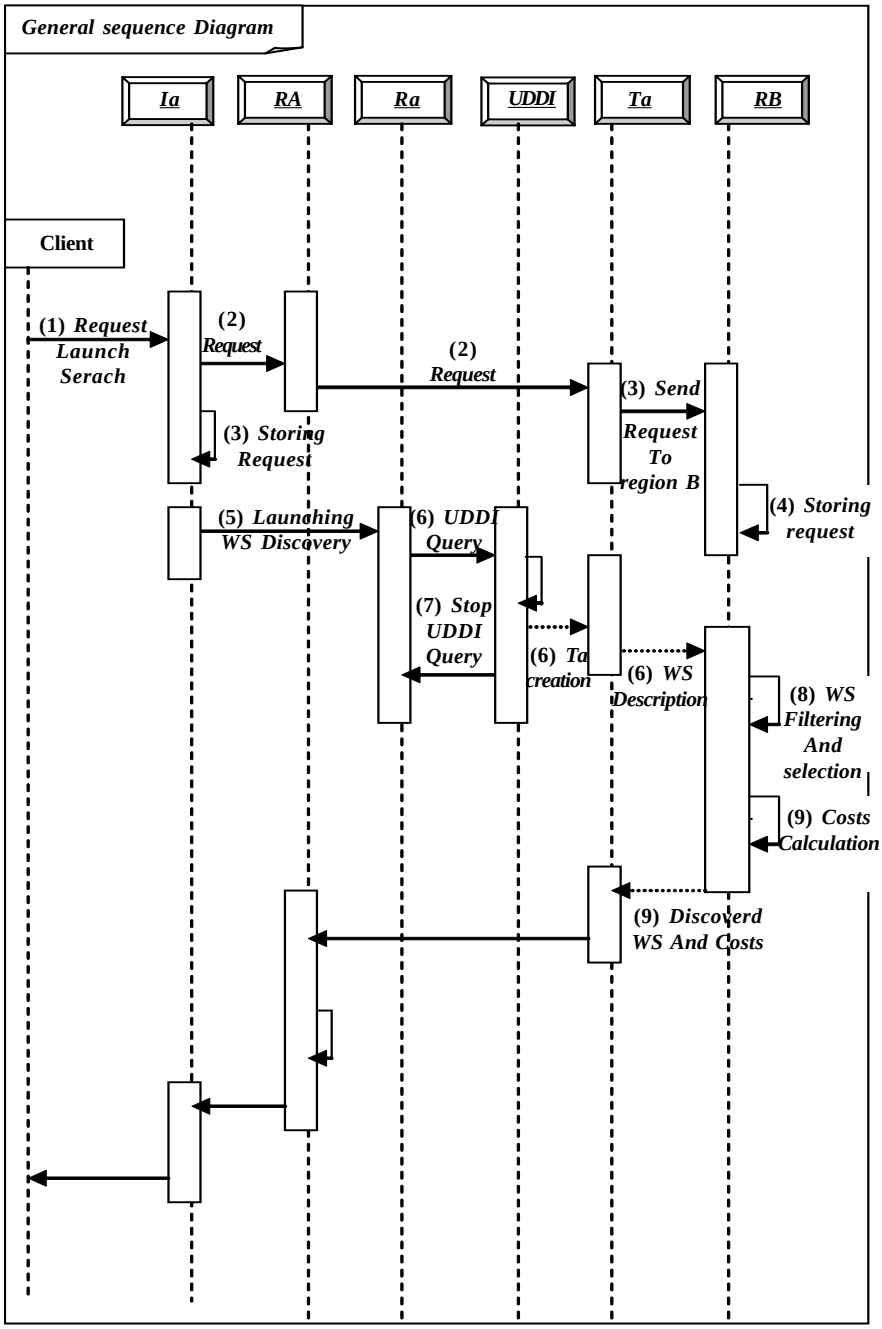


Figure 7. Architectural Components Interaction

5.2.1 The Pre-Treatment Step

This step is applied to the request in the same way that for the description of the Web services:

• Substitution: It consists of substitutes the words of the description, by their synonyms sets (semantic synonyms) in the request, by using WordNet lexical database (in the knowledge base of the region B), we say that two different terms are synonymous if they designate the same sense, or if they are interchangeable in specific contexts. This will facilitates the development of the request and the description by adding semantic information to their representation.

WordNet is a lexical database of English (but there are other versions for other languages such as: WOLF for French language) which is composed by a set of, conceptual-semantic, and lexical relationships between the different words of the English

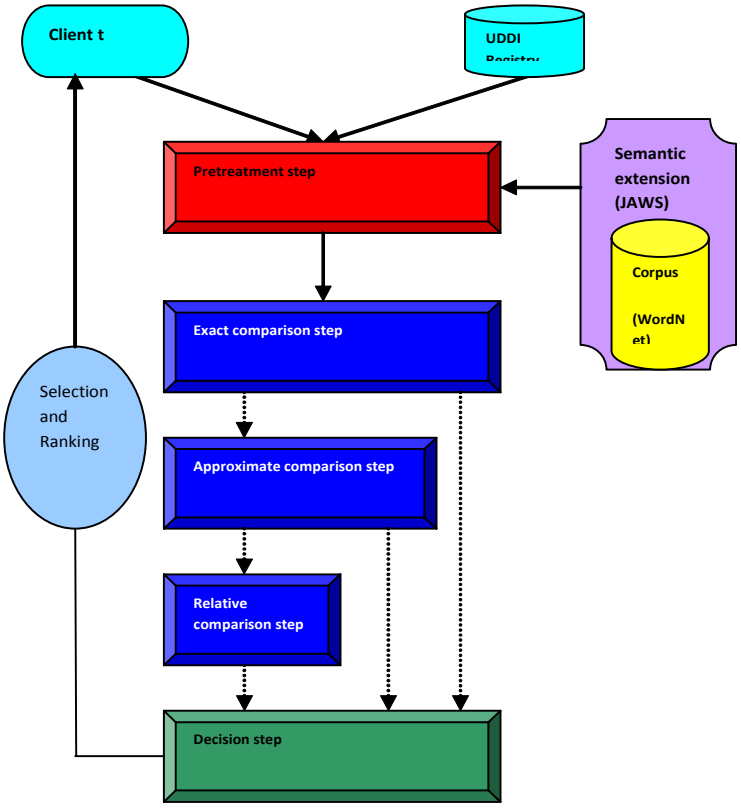


Figure 8. The Filtering and comparison algorithm

vocabulary, these relationships are called cognitive synonyms or Synsets.

In addition to the synonymy, WordNet contains the following concepts:
-1- Hyperonymy: a term that refers to a general sense, compared to a more specific term.
-2- Hyponymy: refers to a specific term, compared to a term that has a more general sense.
-3- Derivationally related form: means the terms that are derived lexically another term and that shares a common sense.

These concepts represent the semantic relations between the words of WordNet lexical database.

WordNet has been used in many domains especially in natural-language processing domain, for information retrieval [17].

• Normalization: normalization is essential for good performance of our algorithm, it simplifies the form in which are written the request and the descriptions of web services, we distinguish two operations:

-1- Tokenization: the goal of tokenization is to found the basic units of the *sense*, and that by applying some operations such as: The segmentation, Treatment of the ambiguous separators (-and ‘) except compound words, translation numbers in words …etc. This allows distinguish, in a clear and easy way, the words which compose the request or the descriptions of the web services.

-2- Lemmatization: the goal of the lemmatization is to transform the flexions (the flexions are the different forms inflected of a word) into their lemma, which will facilitate rapprochement between, the words of the request, and the words of the description of the Web service. We use also WordNet database to found the lemma of each word (we choose lemmatization in place of stemming, because this last don’t take into consideration the semantic and the context of the words in the expression [18]).

• Identification of Keywords: the majority of matching algorithms eliminate the stopwords to extract Keywords from the request and from the description of web services. In our algorithm we don’t eliminate stopwords because we will use them in the calculation of index of shift (see next step) but we will only distingue them from the other words of the request and descriptions of web services, to identify the Keywords (so we don’t eliminate stopwords but we only distingue them to identify the Keywords).

5.2.2 Exact Comparison Step

It consist of comparison, between the request and the part which corresponds it (Keyword by Keyword) in the description of the Web service *(and each time one shifts the request of one alone word compared to the description)* without carried any change (on the request and the description of the Web services), i.e. without lemmatization, because the lemmatization can sometimes make lose the sense of the lemmatized words.

This step is based on four similarities measurements:

5.2.2.1 Measurement of partial similarity (Partial_Sim):

First we calculate the Vector Models based on Normalized Frequencies (VMNF) representation of the request and their corresponding part in the description, according to the following formulas:

• The weight of term *i* in document *j* can be written as:

$$w_{i,j} = f_{i,j} * \log\left[\frac{D}{df_i}\right] \quad (1)$$

df_i : document frequency or number of documents containing term *i*
D: number of documents in a database.

• The weight of term *i* in Request *R* can be written as:

$$w_{R,i} = f_{R,i} * \log\left[\frac{D}{df_i}\right] \quad (2)$$

Then we applied the similarity measure of cosine [19], on the two representations, to calculate the degree of similarity between the request and its corresponding part in the description, according to the following formula:

$$\cos ine\,(R, D_i) = \frac{\sum_i w_{R,j} w_{i,j}}{\sqrt{\sum_i w_{R,j}^{\ 2}} \sqrt{\sum_i w_{i,j}^{\ 2}}} \quad (3)$$

Our choice is justified by the fact that this similarity measurement takes into account the frequency of the words in the text (against the method of JACCARD [19]).

5.2.2.2 Measurement of similarity between words (Sim_Word)

Using the similarity measurement of Jaro-Winkler [20] and wordNet database in knowledge base of the tasks manager B, we calculate the degree of similarity between each word of the request and the word which correspond it in the description (we use WordNet database to verify if the word in the description belong to the derivation forms of the word in the request), and then we calculate the mean of these measures. Our choice is justified by the fact that this similarity measurement is adequate to comparison between short strings (words):

• Jaro distance between two strings S_1 and S_2 is defined by:

$$d_j = \left[\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right] \quad (4)$$

$|S_i|$: is the length of the string S_i
m: is the numb re of the corresponding characters
t: is the number of transpositions

• The similarity measurement of Jaro-Winkler is done by :

$$S_{j-w} = d_j + (lp\,(1-d_j)) \quad (5)$$

l: is the length of the common prefix (maximum 4 characters).
p: is a coefficient which makes it possible to support the chains with a common prefix. Winkler proposes for value $p = 0.1$.

5.2.2.3 Measurement of index of shift (I_S)

The index of shift counts the number of the shifted Keywords of the descriptions compared to the Keywords of the request,

which will make it possible to have the percentage of the shifted words. It's the raison that drive us to leave stopwords in their position in the request and in the descriptions, because we use their positions to calculate the index of the shifted Keywords (but we don't use their senses of course)

5.2.2.4 Measurement of the Global similarity (Glob_Sim)

The global similarity is the average between the three similarity measurements seen above; it allows the selection of the best web services from the web services that have the same degree of acceptability:

$$(Glob\_Sim) = \frac{Partial\_Sim + Sim\_Word + I\_S}{3} \quad (6)$$

5.2.3 Approximate Comparison Step

We must admit that the exact comparison can be very draconian, which can lead to refuse, acceptable web services, from which comes the approximate comparison which consists of: Measurement of partial similarity, the index of shift, and the Global similarity, in the same way that in the exact comparison.

We apply this step, if at least two measures are less than 50% of similarity (calculate after the exact comparison step), but this time with the request and descriptions lemmatized, because the lemmatization influence largely the index of shift and the partial similarity (we eliminate the similarity between words because the words are lemmatized and substituted, making this operation unnecessary).

5.2.4 Relative comparison Step

The index of shift allows the calculation of the percentage of the shifted words of the request compared to the description, but this index don't take into consideration the distance between the key-words in the request relatively to the *same* Key-words in the description, if the distances between the Key-words don't change the meaning of the description (the description still correspond to the request).

5.2.5.1 Measurement of the index of Relativity (I_R)

The index of relatiity calculte de distance between each Key-word of the request and the same Key-word in the discription, then it addition between those disctances then it divide the result on the number of the request Key-words. Then the result will be transformed on persentage format to be used in the decision step.

We apply this step: if the similarity of Index of shift is less than 50% of similarity (calculate after the approximate comparison step), with the request and the description lemmatized (the cosine similarity must be greeter than 50% of similarity after the approximate comparison step).

5.2.5 Decision Step

The decision step consists of interpreting the results obtained after the calculation of the similarity measures and the index of shift:

After the end of exact comparison step:

- If all measurements equalizes or exceeds 50% then the Web service is: *strongly accepted.*
- If two measurements equalizes or exceeds 50% then the Web service is: *medium accepted.*
- If none measurements equalizes or exceeds 50% then the web service is: *refused.*
- If only one of measurements equalizes or exceeds 50% then one passes to the approximate comparison step.

After the end of the approximate comparison step:

- If two measurements equalizes or exceeds 50% then the Web service is: *weakly accepted.*
- If the two measurements are not equal or do not exceed 50% then the Web service is: *refused,*
- If only the index of shift measurement is less then 50% then one passé to the relative comparison step.

After the end of the Relative comparison step:

- If the measurement of the index of relativity equalizes or exceeds 50% then the Web service is: *Very weakly accepted.*

• If not, then the web service is: *refused.*

After the attribution of the degree of acceptability to each Web service, the manager agent of the region B select a reasonable number (between 10 and 15 [21]), of Web services which have greatest degree of acceptability. If the number of selected Web services exceeds 15 (because they have the same degree of acceptability), then one uses the global similarity to rank the Web services in order to extract the 15th best web services.

Finally we can say that our algorithm avoids the disadvantages of the research based keywords, the VSM vector representation, the cosine similarity, and kernel based methods at the same time. So as responses to the problems seen in section 3:

• To avoid having few web services (one of the great problems of key words based research): we use a large number of researcher mobile agents (one mobile agent fore each Internet region), which increase our chances of avoiding having few web services; but it may be that we obtain a great number of Web services, so;

• To avoid having a large number of web services return to the client, we use the selection algorithm, that allows, for each web service, has attributed a degree of acceptance or refusal, then the manager agent of the region B choose a reasonable number of web services that have the greatest degree of acceptance.

• The filtering algorithm takes into account the semantic side of the request and web services descriptions, thanks to the use of WordNet lexical database.

• The filtering algorithm take into account the sub strings, thanks to the use of Jaro-Winkler measurement and the derivation forms of each word.

• The spamming keywords are taking in consideration, thanks to the use of the Normalized Frequencies (VMNF) representation.

• The filtering algorithm takes into account the repeated words thanks to the use of the cosine measure.

• The filtering algorithm take into account the arrangement of words in the description of web services compared to client's requests thanks to the use of the index of shift.

• The filtering algorithm takes into account the distances between the key-words of the request and the key-words of descriptions, thanks to the use of the index of relativity.

• The filtering algorithm is an automatic one; it means that our algorithm doesn't need the intervention of developers to determine any parameter, contrary to kernel based methods seen in section 3.

The following figure illustrates our filtering and comparison algorithm:

```
Input: the Request of client and Description of web service
Output: degree of acceptability and percentage of similarity
Pre-treatment
Substitution
1:Description=Substitution(Description);
/*the substitution use the WordNet in the task manager of the region B, to find the
semantic synonyms of the words of the request in the description of the web service to
replaces them in this last*/
Tokenization
2: Request= Tokenise (Request);
3: Description= Tokenise (Description);
Identification of Keywords
4: Request= KW_Idf(Request);
5: Description= KW_Idf(Description);
Exact comparison
6: VRequest = VMNF (Request);
/*VMNF: is the function that calculate the Vector Models based on Normalized Frequencies
representation*/
7: For (i=1; i= M-N+1; i++) do
// M: Number of the words of description
```

```
// N: Number of the words of the request
/*M-N+1: Number of the traversed sections of the description which corresponds to the
request, word by words */
Measurement of partial similarity
8: VDescription = VMNF (Description[i]);
9:Partial_Sim=Similarity_Cosines(VRequest, VDescription);
Measuremen of similarity between words
10: Sim_Word=Similarity_Words(Request, Description[i]);
Measurement of index of shift
11: I_S=Index_Shift (Request, Description[i]);
12: Tab_Sim [i][1]=Sim_Word;
13: Tab_Sim[i][2]=Part_Sim;
14:Tab_Sim[i][3]=I_S;
15: End for
16: Cont= Optimal_Values (Tab_Sim);
/* The optimal values are the combination between the greatest number of the best
three values (Partial_Sim, Sim_Word,  I_S) in the same segment of the description of
Web service which corresponds to the request*/
Measurement of global similarity
17: Glob_Sim =Percentage_Similarity (Tab_Sim);
/* The similarity percentage is calculated, between the optimal values of the  :
Partial_Sim, Sim_Word and  I_S */
/* Cont: is the number of measurements which exceed 50% of the best section (of Web
service description) which correspond to the request */
Decision
18: If (Cont==3) then
19: Return: the Web service is strongly accepted;
20: If (Cont==2) then
21: Return: the Web service is medium accepted;
22: If (Cont==0) then
23: Return: The Web service is refused;
24: If (Cont==1) then
Pre-treatment
Lemmatisation
25: Request= Lemmatize(Request);
26: Description= Lemmatize(Description);
Approximate comparison
27: VRequest = VMNF (Request);
28: For (i=1; i= M-N+1; i++) do
29: VDescription = VMNF (Description);
Measurement of partial similarity
30:Partial_Sim=Similarity_Cosines(VRequest,VDescription[i]);
Measurement of index of shift
31:I_S=Index_Shift(Request, Description[i]);
32: Tab_Sim[i][2]=Part_Sim;
33:Tab_Sim[i][3]=I_S;
34: End for
35: Cont= Optimal_ Values (Tab_Sim);
Measurement of global similarity
36: Glob_Sim =Percentage_Similarity (Tab_Sim);
Decision
37: If (Cont==2) then
38: Return: The Web service is weakly accepted;
39: If (Cont==0) then
```

```
40: Return: The Web service is refused;
41: If ((Tab_Sim[i][2]>=50%) && Tab_Sim[i][3]<50%) then
Measurement of the index of relativity
42: I_R=Index_Relativity (Request, Description[i]);
Decision
43: If (I_R>=50%) then
44: Return: The Web service is very weakly accepted;
45: else
46: Return: The Web service is refused;
47: End if
48: End
```

Figure 9. Selection and ranking algorithm

6. Implementation

6.1 Development platforms

6.1.1 Cloudsim platform

One of the great problems, to develop Cloud computing applications is the environment where we can reproduce testes. Because, if we do testes in real cloud computing environments like Amazon EC2, the experiments will be, expensive and very limited. But with simulation, developers can test and optimize their Cloud applications without paying any thing, and then the clients can test, the performances of their services in repeatable and controllable environment free of cost, before putting those applications in real Cloud computing environment [22].

Cloudsim is a powerful tool for modeling, simulation and experimentation of Cloud infrastructure, which allows developers to test their Cloud application services, without getting concerned about the low level details related to Cloud-based infrastructures and services.

The principle functionalities of Cloudsim are [23]:

- Support for modeling and simulation of large scale Cloud computing datacenters.
- Support for modeling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines.
- Support for modeling and simulation of energy-aware computational resources
- Support for modeling and simulation of datacenters network topologies and message-passing applications.
- Support for modeling and simulation of federated clouds.
- Support for dynamic insertion of simulation elements, stop and resume of simulation.
- Support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

6.1.2 Aglet platform [24]

Aglet is a project developed by the IBM Tokyo research laboratory in 1995, it aim to create a uniform platform for the execution of mobile agents, in heterogeneous environments like Internet.

An Aglet is characterized by:

- High level of security.
- Clarity of structure.
- Communication based on sockets: RMI, HTTP, ATP.
- Very good documentation.
- Famous product.

6.1.3 CloudAnalyst

With Cloudsim, it's very difficult to quantify some important parameters such as geographic location of datacenters to calculate

distance between them and users, impact of number of simultaneous users, and Network in implication.

CloudAnalyst is a simulation tool that allows developers to execute and to test different simulations in repeatable way, taken into account the different parameters seen above.

	architecture	Number of host	MIPS	Ram (MB)	Time zone	Cost by using Datacenter ($/CPU Tim)	Cost per Memory ($/S)	Cost per Bw ($)	Cost per storage ($/S)
Characteristics	X86	1	10000	4000	GMT +10	0.3	0.05	0.001	0.05

Table 1. Datacenters Characteristics

Because CloudAnalyst is equipped with an easy to use graphical user interface (GUI) (see Figure 12), it allow developers to set up experiments quickly and easily, which mean that CloudAnalyst allows to separate between the simulation experimentation exercise from a programming exercise, so a modeler can focus on the simulation complexities without spending too much time on the technicalities of programming using a simulation toolkit [25].

Figure 10. CloudAnalyst interface

6.2 Simulation configuration

To evaluate the performances of our system we will do two case studies:

The First case study consists of creation of one Cloud computing service provider (CCSP) that contains three datacenters: the first datacenter represent the region A of Cloud and the rest of datacenters represents the region B (because this region will execute the matching algorithm, which requires more resources). Then we create six tasks (representing the six principle tasks to treat the client request in our discovery system), then we test, with one client request, the impact of number of Virtual machines (VM) on the global time execution (response time). We use Cloudsim to implement the different executed scenarios, Aglets to implement the multi-agent system and JAVA to achieve the implementation of the filtering algorithm.

This case study allows us to evaluate, the internal performances of our system, which mean, the evaluation of our Virtualization policy in each Datacenter of the Cloud computing service provider (CCSP).

The second case study (we use CloudAnalyst) consist of creation of 6 user bases, representing the 6 main regions of the world, which allows us to test the performances of our system from different geographic location (because the distance between datacenters and client have a significant impact on the quality of the Cloud services).

We will test the performance of our system for 3 hours, with supposing that the most of clients use the application in the night after work.

We suppose also that 50% of registered clients are online during the peak time simultaneously and 10% of registered clients are online during the off-peak time hours.

We used also a time-shared policy to schedule resources to VMs. The number of simultaneous clients from a single user base is 10000 and the number of simultaneous requests, a single application server instance can support is 1000.

The second case study allows us to evaluate, the external performances of our system, which mean, the evaluation of the impact of the number of Datacenters (each data center can represent a New CCSP) on the response time.

User Base	Region	Peak Hour	Online Clients During Peak Hours	Online Clients During Off-peak Hours
UB1	0- N America	GMT: 14h-17h Local time:20h-23h	1000	200
UB2	1- S America	GMT: 16h-19hLocal time:20h-23h	500	100
UB3	2- Europe	GMT: 21h-00hLocal time:20h-23h	300	60
UB4	3- Asia	GMT: 02h-5hLocal time:20h-23h	200	40
UB5	4- Africa	GMT: 22h-01hLocal time:20h-23h	100	20
UB6	5- Ocenia	GMT: 06h-09hLocal time:20h-23h	70	14

Table 2. User Bases Characteristics

The table 1 and 2 shows, respectively, the datacenter characteristics and the user base, of the first and the second case study.

6.3 The First case study

Our case study is based on the following steps (but each scenario has its specifications):

6.3.1 Datacenter creation

Using "*Datacenter*" class we create our datacenters that represent the Cloud computing service provider of our system, using this class we define the hardware and the software characteristics of our system as shown in table 1.

```
// Datacenter Creation
Datacenter datacenter0 =
createDatacenter("Datacenter_0");
```

6.3.2 Creation of datacenter Broker

Using "*DatacenterBroker*" classs, we create the Broker wich is responsible for allocation of cloud resources and submitting of different tasks to the virtual machine(s), and it's also, to the broker, to choose another Datacenter(s) (of another CCSP) in case of virtual machines creation failed in this datacenter for example.

```
//Create Broker
DatacenterBroker broker
=createBroker();
// Broker Identification
int brokerId = broker.getId();
```

6.3.3 Creation of Virtual machine(s) (VM)

When the client submits his request to our discovery system, a virtual machine(s) is created to treat this request.
The table 3 shows the virtual machines characteristics:

```
// Virtual machin Creation
Vm vm1 = new Vm(vmid, brokerId,
mips, pesNumber,  ram, bw, size,
vmm, new
ClouletSchedulerTimeShared());
```

6.3.4 Creation of six Tasks

Using "*Cloudlet*" class we create six Cloudlets, each cloudlet represent one of the six main tasks for treatment of the client request.

As shown in Table 4, each Cloudlet has its own characteristics, because each one represent, one of the six different tasks.

```
// Creation of cloudlet
Cloudlet cloudlet1 = new
Cloudlet(id, length, pesNumber,
fileSize, outputSize, utilizationMod
el, utilizationModel, utilizationMod
el);
```

	MIPS	RAM (MB)	BW (MB)	CPU
Virtual machine	250	512	100	1

Table 3. Virtual machines characteristics

Tasks	Cloudlet Number	Length	File Size	Out put Size
Storage and key words extraction from the request	1	10	5	5
Creation of transport and interface agent	2	80	40	40
Creation of Researcher agents	3	200	100	100
Storage of web services descriptions	4	200	100	100
Matching algorithm execution	5	4000	300	300
Creation of transport agents.	6	40	20	20

Note: *The values of cloudlets characteristics were chosen after making an estimation of the sizes of the six tasks.*

Table 4. Cloudlets characteristics

6.4 The second case study

The second case study contains six scenarios. The basic one, consist of using two Datacenters, the first datacenter represent the region A of Cloud and contain 50 VMs, and the second datacenter represent the region B of Cloud with 100 VMs because this region will execute the selection algorithm.

The second scenario consist of adding new datacenter, representing region C of the Cloud, this region offer Software of keywords extraction (from the request) as a service. The datacenter contain 15 VMs.

The third scenario consist of adding a fourth datacenter that represent the region D of Cloud, this region offer a software of creation and managing of mobile agent system (managing of researcher and transport agents), as a service. This datacenter contain 30 VMs.

The fourth scenario consist of adding a fifth datacenter that contain 20VMs, representing the region E of Cloud, which offer a Storage-As-a-Service, to store the descriptions of the discovered web services.

The fifth scenario consists of adding a new datacenter with 50 VMs, representing the region F of Cloud, which offer our Software of selection as a service.

Finally, in the last scenario we add a seventh datacenter with 100 VMs, representing the region G, that offer the same services as the region B of Cloud.

Each time, we add a new datacenter; we put them in a new world region, to test the impact of the geographic location on the response time.

7. Results and discussion

7.1 First case study

We will vary the number of virtual machines from 1 to 6, then we submit (equitably) the cloudlets to the virtual machines, the table 5 show the experiences results:

Scenarios	Number of Datacenters	Distribution of VMs	Overall average response time (milliseconds)	Global costbasedon the quantity of used memory
1	2	50VM/100VM	237.80	$ 136.06
2	3	50/VM/100VM15VM	208.68	$ 149.57
3	4	50VM/100VM15VM/30VM	141.31	$ 176.58
4	5	50VM/100VM15VM/30VM20VM	101.00	$ 182.59
5	6	50VM/100VM15VM/30VM20VM/50VM	92.62	$ 239.61
6	7	50VM/100VM15VM/30VM20VM/50VM100VM	67.62	$ 329.66

Table 6. Simulated scenarios and Results of the first case study

Our case study is composed of six scenarios:

The first scenario consists of creation of one virtual machine and submitting all of cloudlets to this VM.

The second scenario consist of creation of two VM, then, dividing the Cloudlets on two groups, each group contain three Cloudlets, then we submit each group two one VM.

The third scenario is similar to the second, but with three VM and three Cloudlets groups, each group contain two Cloudlets.

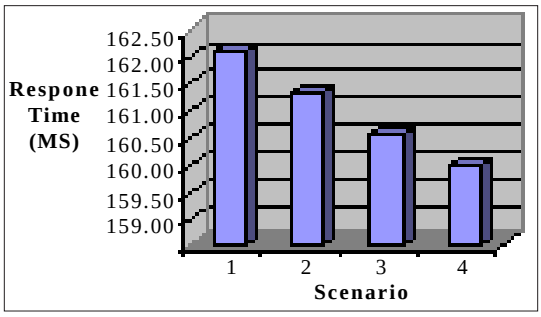
The fourth scenario consist of creation of six VM, each VM is used to execute one Cloudlet.

The raison why we do not create four or five VMs, is that we can not divide equitably six Cloudlets out of four or five VMs.

The scenarios show that the internal functioning of our system optimizes the global response time, due to:

- The decomposition of the discovery operation, to the main elementary tasks (the six main tasks),
- Creation, for each task, its own virtual machine, which decrease (each time we add new VM) the Global time execution.

Which confirm the efficiency of our policy, which consist of benefiting, as maximum as, possible from the virtualization character of Cloud computing technology. The following figure depicts variation of response time:



Note: *We note also that, we was bring back, to add, more hosting machines, in the datacenters of the last scenario to allow the creation of the six VMs, which conduct us to the second case study.*

Figure 11. Summary of response time results

Scenarios	Number of Virtual machines	submitting of cloudlets	Response time (milliseconds)	Global cost
1	1	Six cloudlets to one VM	162.1	$0.81
2	2	Three cloudlets per VM	161.33	$0.63
3	3	Two cloudlets per VM	160.53	$0,93
4	6	One cloudlet per VM	160	$1.81

Table 5. Simulated scenarios and Results of the second case study

The results show that bringing, the datacenters, that represent regions C and D, closer to region A improve the response time, and consequently it reduce the global cost (quality of Cloud service). The same for the region B, bringing regions E, F and G to this region improve the quality of Cloud service (QoCS). So we can say that each time the distances between regions C, D, E...Etc and regions A and B are short, and the distance between the client and region A is short, the quality of Cloud service is more enhanced.

The results also show, that we can enhance the quality of our Cloud service, if we create a new Datacenters (representing each one a new CCSP) that are intended to a precise tasks (Keyword extraction, managing of mobile agent system, selection of web services).

The third note that the results show, is that we can optimize the quality of our Cloud service, by dynamically increasing and decreasing the number of VMs in the datacenters (CCSPs) which are near from the geographical zones that are in peak hour period.

Finally, this case study show that our Cloud system optimizes the response time, by using mobile agent (specially transport mobile agent) to relate the different CCSPs to maximize the benefit from the different datacenters that compose our system.

The following figure depicts variation in the response time:

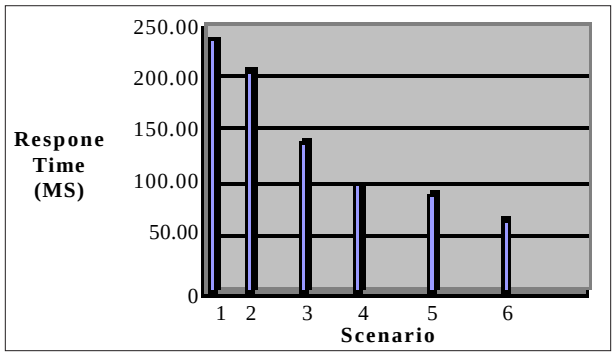


Figure 12. Summary of response time results

7.3 System Interfaces

The first interface is the login interface. It allows the client to access his account or to launch the creation of new account. The second interface is the creation account interface; it allows client to create a new account by introducing his personnel information's like: Name, Address, Phone number …Etc. The third interface is the information payment interface; it allows client to introduce: some information's about his credit card, and the billing address. The fourth interface is the search interface; it allows client to introduce his request in textual format, then the client launch a search and waits for the results (the application display an interface of waiting, this last show the message "*please wait…*"): The last interface is the results interface: it show, in one hand, the selected web services with their degree of acceptance and their global similarity percentage, and in the other hand, the billing details, like: Datacentres costs, storage costs, global cost.. ..Etc.

Figure 13. Login Interface

Figure 14. Creation new account Interface

Figure 15. Information payment Interface

Figure 17. Results Interface

8. Conclusion

In this paper we have presented a new approach of web services discovery based on mobile agents in Cloud computing environment, this approach benefit from the "*Business Model*" of Cloud computing, which allows, at the same time, an instantaneous access to the Cloud services, and which guaranteed the availability of the resources requested by the client.

We present also a new algorithm of selection and ranking that provides solutions for problems encountered in the search based key-words, VSM representation, and kernel based methods.

With regard to the prospects for our work, we plan to approach the following points:

• Improve the architecture of researcher's mobile agents to perform a search more relevant, rather than the key words based research.

• Integrating an algorithm that takes into account non-functional aspects of web services such as: quality of service, service reputation, semantics interfaces, and cost.

References

[1] Zhang, Z., Zhang, X. (2009). Realization of Open Cloud Computing Federation Based on Mobile Agent, IEEE.

[2] Raov, S., Raonk, N., Kumari, E. (2009). Cloud ComputingG: An Overview, *Journal of Theoretical and Applied Information Technology*.

[3] Zhang, S., Zhang, S., Chen, X., Huo, X. (2010). Cloud Computing Research and Development Trend, Second International Conference on Future Networks, IEEE Computer society.

[4] Baumann, J. (2000). Mobile Agents: Control Algorithms, Lecture Notes in Computer Science, Edited by Goos, G., Hartmanis, J., van Leeuwen, J. p. 7-8.

[5] http://uddi.org/pubs/uddi-v3.00-published-20020719.htm

[6] Fensel, D., Keller, U., Lausen, H., Polleres, A., Toma, I. (2005). **WWW** OR **W**HAT IS **W**RONG WITH **W**EB SERVICE DISCOVERY. Position Paper for the Workshop on Frameworks for Semantics in Web Services, Innsbruck, Austria.

[7] Jingliang, C., Zhe, M. (2010). Research on Web Service Discovery Based on Mobile Agents. International Conference On Computer Design And Appliations.

[8] Chen, L., Yang, G., Wang, D., Zhang, Y. (2010). WordNet-powered Web Services Discovery Using Kernel-based Similarity Matching Mechanism, Fifth IEEE International Symposium on Service Oriented System Engineering.

[9] Aversa, R., Di Martino, B., Mazzocca, N. (2004). Terminal-aware Grid resource and service discovery and access based on Mobile Agents technology. *In*: Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE Computer society.

[10] He, Y. W., Wen, H. J., Liu, H. (2005). Agent-based Mobile Service Discovery in Grid Computing. *In*: Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology, IEEE Computer society.

[11] Wagener, J., Spjuth, O., Willighagen, L. E., Wikberg, ES. J. (2009). XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services. BMC Bioinformatics, BioMed Central.

[12] Chen, H., Li, S. (2010). SRC A Service Registry on Cloud Providing Behavior-aware and QoS-aware Service Discovery. Service- Oriented Computing and Applications (SOCA), IEEE International Conference.

[13] Mather, T., Kumaraswamy, S., Latif, S. (2009). Cloud Security and Privacy, O'Reilly, p. 7-9.

[14] http://searchdatacenter.techtarget.com/definition/scalability

[15] http://www.auml.org/

[16] https://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm

[17] http://wordnet.princeton.edu/wordnet/

[18] Joël Plisson, Nada Lavrac, Dunja Mladenic. (2004). A Rule based Approach to Word Lemmatization, *SiKDD multiconference*, 12-15 October, Ljubljana, Slovenia.

[19] Rosset, S., Jardino, M. (2008). Comparaison de documents: mesures de similarité et mesures de distance, june 10.

[20] Cohen, W. W., Ravikumar, P., Fienberg, S. E. (2003). A Comparaison of String Distance Metrics for Name-Matching Tasks, American Association for Artificial Intelligence (www.aaai.org).

[21] Denayer L. M. (2004). D´ecouverte de Service : ´Etude d'Approche Syntaxique et S´emantique, http://www.info.fundp.ac.be/~ven/CISma/FILES/40b0fc1cbcbd42004_denayer_marie_laetitia.pdf

[22] Calheiros, N. R., Ranjan, R. (2011). Beloglazov. A, De Rose, A. C. F., Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE).

[23] http://www.cloudbus.org/cloudsim/

[24] The Aglets User's 2.0.2 manual, Aglets Development Group,(2009).

Wickremasinghe, B., Calheiros, N. R., Buyya, R. (2010). CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications, http://www.gridbus.org/reports/CloudAnalyst 2009. pdf.

[25] Rittinghouse, J. W., Ransome, J. F. (2010). Cloud Computing Implementation, Management, and Security, CRC Press, Taylor & Francis Group, pp xxxii-xxxvi.

[26] Palathingal, P., Chandra, S. (2004). Agent Approach for Service Discovery and Utilization, *In*: Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[27] Newcomer, E. (2000). Understanding Web Services- XML, WSDL, SOAP and UDDI, David Chapell Series Editor, p. 110-154.