A New Cloud Computing Framework Based on Mobile Agents for Web Services Discovery and Selection

Saouli Hamza, Kazar Okba, Benharkat Aïcha-Nabila

# Computer Science Department, University of Biskra, 07000
Biskra, Algeria

1hamza_saouli@yahoo.fr
2okbakazar@yahoo.fr

*Laboratoire d'InfoRmatique en Image et Systèmes d'information, Lyon, France
3nabila.benharkat@insa-lyon.fr

Abstract: The increasing use of Web services as well as the growing interest lent to Cloud computing technology and its capacity of calculation, storage and Cloud services, impose today that the traditional mechanisms of discovery must be re-examined. This specificity of discovery over the cloud led us to propose a new framework based on mobile agents dedicated to web services discovery. So the problematic, consist of making this framework, benefiting from the characteristics of Cloud computing and mobile agents technologies. More accurately, we propose a new algorithm of comparison between the request of the client and the description of the Web services. The proposed system is composed of two areas of Cloud. The first deals with Key words based research and second supports the filtering of the found web services. This discovery is performed by an algorithm based on the calculation of similarities between the request and the description of the web services. The results show that the quality of the service (response time) has been enhanced due to the proposed virtualization policy, which consist of decomposing the discovery operation to six main tasks then creating, for each task, its own virtual machine (VM), each time we create a new VM the response time is reduced.

Keywords: Cloud Computing, Mobile Agent, MABOCCF, Discovery of Services, WSDL, Similarity measures.

1. Introduction

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.

Web services discovery is one of the most important applications on Internet. It aims to create a mechanism which makes it possible to choose and to access the web services that best corresponds to the clients requests.

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 for web services selection to obtain more precision with the discovered web services.

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

2. OCCF and MABOCCF [1]

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.

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 experiment results manifests that, the using 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.

3. Related Works and Problems

In [2] Chen et al. proposed a model “Service Registry on Cloud (SRC)” which is an extension of a service model based on keywords, this application is deployed like a Cloud application. This model aims to discover 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 services) of Web services as GFS (Google file system) files in the Cloud, and uses the mechanism of MapReduce to process files. This work suffers from two problems; the first problem is the absence of one tool (a tool like mobile agents): autonomous, auto-adaptable, and especially adapted to the nature of web services that are distributed on the Internet, for the research of Web services, represents a persistent problem. The second problem is the absence of any indication for eventual relations between the different SRC instances that can publish the WSs, the mobile agents technology also, can be a good solution for making those relations.

In [3] the authors proposed an architecture based mobile agents for the management of mobile services in Grid computing and described in details 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 this architecture is that the economic model of Grid computing suffer from the problems of instantaneous 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 the paper [4] 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, on 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 service compared to the request and the repeated words in the request.

In [5] the authors proposed an approach based on WordNet for vector extracting (vector space model VSM), which allows, not only to add semantic informations 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 e-learning, are used to estimate the similarity between web services. The classic VSM representation suffers from three main problems. First, the VSM vector representation does not take into account sub strings. Second, the spamming keywords are ignored in the classical vector representation. Third, the VSM vector representation does not take into consideration the arrangement off words in the request. In addition, 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.

4. The Proposed Framework

The following figure shows the general architecture of the proposed discovery system:

Figure 1. General architecture.

5.1 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 the discovery system
2. Login to his account (in case of existing account)
3. Introduce his request (in textual form);
4. Display the results and billing.

5.2 Cloud Computing Part

5.2.1 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 agent system, which has for essential goal, the keywords based discovery of web services, by creating the researcher mobile agents which will deals 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.

The datacenters that contain the different Virtual machines of regions A are always active, which guarantee the Instantaneity (instantiate access to resources) and the availability of the resources, to the clients.

5.2.2 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 region is to give more precision and to avoid the disadvantages of the
syntactic research; when the selection of the web services is finish, a transport agent is created to transport the selected web services to the region A.

As the region A, the datacenters that contain the different Virtual machines of regions B are always active.

Moreover, each region contains a Task Manager:

5.2.3 Task Manager of Region A:

The Task Manager of the region A, is responsible for the reception of the request, and associates 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 Task Manager of the region A is composed of three modules:

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 WSDL indexes that describe the web services, divided in a set of groups (each group contain a set of WSDL indexes that are in the same Internet region). The second directory, contain the addresses of other Cloud computing regions that provides the same services as the region A, which guarantee the massive scalability to the proposed 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, toward another region which offers the same services as the region A or more resources to this region).

2. Treatment module (TM): the TM 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 WSDL indexes 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: the proposed system flow the Cloud computing way to calculate costs, it mean that the client pay only what he use.
   - Updating of the software of request treatment (software as services): the client doesn’t need to pay, each time, for updating the new versions of the software using in the discovery operation, it’s to the Task 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. The task manager 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.

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

5.2.4 Task Manager of Region B:

The Task Manager of the region B has as a function, the reception of the request, and the Web services descriptions found by the researcher’s agents. It transmits them to the software of comparison and filtering, with the lemmas and derivation forms of each words of the request and descriptions of the discovered Web services. The Task Manager of the region B is composed of three modules:

1. Knowledge base (KB): which contain a dictionary of synonyms, derivations forms and lemmas that will be used in, the execution of the filtering and comparison algorithm. the KB contain also, a directory, that contain the addresses of other Cloud computing regions that provides the same services as the region B, which guarantee the massive scalability to the system, for this region.

2. Treatment module (TM): the TM is responsible of:
   - Creation of a transport agent which deal 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 filtering and comparison.
   - Creation of virtual machine: The task manager is responsible of the creation of three Virtual machines that treats the three main task of this region, namely: Storage of web services descriptions, Matching Algorithm execution, Creation of transport agents.

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

![Figure 2. Region A Task manager.](image1)
![Figure 3. Region B Task manager.](image2)
5.3 Mobile Agents Part

5.3.1 Interface Agent (Ia): the Interface agent is used to transmit the client request to the region A and display selected web services to client via the web interface.

5.3.2 Transport Agent (Ta): the transport agent can be used, in one hand, to transport the client request from the region A to the region B, and on the other hand, to transport the Web services detected by the researcher agents to the region B, and then have to transport each Web service selected by the region B to region A.

5.3.3 Researcher Agent (Ra): Researcher agent is useful to do a syntactic research based on keywords (from the request of client).

- The scenario of use of the proposed system is described by the following steps:
  1. Send the client request via the web interface.
  2. An Interface agent is created at the region A to bring the request to this region, via the task manager.
  3. The task manager 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 WSDL indexes 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 researchers 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 the description of the web service (and the identifier of the corresponding request) to the region B via the task manager, to be stored in the zone corresponding to the identifier transmitted with the web service description.
  7. After a predetermined period of time, the researchers agents stop their Internet search.
  8. The task manager of the region B, launch the filtering algorithm to select the best web services that correspond to the client request (For the rejected web services, the task manager will delete them from the region B).
  9. The task manager of the region B calculate the costs of the resources used in the filtering 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 task managers of the region A calculate the Global cost.

5.4 Why combining Mobile agents with Cloud computing technology?

- Unlimited scalability: 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.
- More intelligence: the characters of autonomy and auto-adaptability of mobile agents give to the system more intelligence in the discovery operation.
- 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.
- 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.
- 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.

5.5 Operating Mode

The scenario of use of the proposed system is described by the following steps:
5. Algorithm of Comparison and Filtering

The proposed 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 <documentation>, the idea behind the choice of the <documentation> 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 <documentation>.

The proposed algorithm is applied to all <documentation> tags contained in the WSDL file that describe the matched web service, then it select the best result for taking decision in the last step.

The algorithm is based on 4 steps:

5.1 The pre-treatment step:
This step is applied to the request in the same way as for the descriptions of the Web services:

1. Substitution: It consists of substitutes the words of the description, by their synonyms in the request, using the dictionary in the knowledge base of the region B. This facilitates the calculation of similarity between the request and the description of the Web services.

2. Normalization: normalization is essential for good performance of the proposed algorithm, it simplifies the form in which are written the request and the descriptions; we distinguish two operations:
   • Tokenization: the goal of tokenization is to find the basic units of *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.
   • 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 the rapprochement between, the words of the request, and the words of the descriptions.

5.2 Exact comparison step:
It consist of comparison, between the request and the part which correspond it (word by word) in the description of the Web service (*and each time one shift the request of one alone word, compared to the description*) without carried any change (on the request and the description), 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:

1. Measurement of partial similarity (Partial_Sim): first we calculate the Vector Model based on Normalized Frequencies (VMNF) representation, of the request and their corresponding part in the description. Then we apply the cosine similarity measurement [6], on the two representations, to calculate the degree of similarity between the request and its corresponding part in the description.

2. Measurement of similarity between words (Sim_Word): using the similarity measurement of Jaro-Winkler [7] and the dictionary of Task manager B, we calculate the degree of similarity between each word of the request and the word which correspond it in the description (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.

3. Measurement of index of shift (I_S): the index of shift count the number of the shifted words in the description compared to the words of the request, which will make it possible to have the percentage of the shifted words in the description.

4. Measurement of 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.
5.3 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 consist 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 the description, lemmatized, because the lemmatization influence largely the index of shift and the partial similarity (we eliminate the similarity measurement between words because: the words are lemmatized and substituted, making the calculation of this measurement unnecessary).

5.4 Decision step:
The decision step consists of interpreting the results obtained by calculating the similarity measures and the index of shift:

After the end of the 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 only one of the measurements equalizes or exceeds 50% then one passes to the approximate comparison step.
- If none measurements equalizes or exceeds 50% then the web service is: refused.

After the end of the approximate comparison step:
- If two measurements equalizes or exceeds 50% then the Web service is: weakly accepted.
- If only one of the measurements is not equal or do not exceed 50% then the Web service is: refused.

After the attribution of the degree of acceptability to each web service, the task manager of the region A select a reasonable number (between 10 and 15 [8]), of Web services which have the greatest degree of acceptability. If the number of the selected web services exceeds 15, then one uses the global similarity to rank the Web services in order to extract the 15th best web services.

Algorithm of comparison and filtering

Input: the Request of client and Description of web service
Output: degree of acceptability and degree of similarity

Pre-treatment
1: Tokenisation
2: Request = Tokenize (Request);
3: Description= Tokenize (Description);
4: Description= Substitution (Description);
/*the substitution use the dictionary in the task manager of the region B, to find the synonyms of the request words in the description of the web services to replaces it in this last*/

Exact comparison
5: VRequest = VMNF (Request);
6: 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 */
   7: VDescription = VMNF (Description[i]);
   8: Partial_Sim=Similarity_Cosines(VRequest, VDescription);
   9: Sim_Word=Similarity_Words(Request, Description[i]);
   10: I_S=Index_Shift (Request, Description[i]);
   11: Tab_Sim[i][1]=Sim_Word;
   12: Tab_Sim[i][2]=Part_Sim;
   13: Tab_Sim[i][3]=I_S;
   14: End for

15: 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*/
16: Glob_Sim=Percentl_Similarity (Tab_Sim);

Decision
17: If (Cont==3) then
18: Return the Web service is strongly accepted;  
19: If (Cont==2) then
20: Return the Web service is medium accepted;  
21: If (Cont==0) then
22: Return The Web service is refused;  
23: If (Cont==1) then

Pre-treatment
24: Lemmatisation
25: Request= Lemmatise (Request);
26: Description= Lemmatise (Description);
27: VRequest = VMNF (Request);

Approximate comparison
28: For (i=1; i= M-N+1; i++) do
29: VDescription = VMNF (Description[i]);
30: Partial_Sim=Similarity_Cosines(VRequest, VDescription[i]);
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);

Decision
36: If (Cont==2) then
37: Return The Web service is weakly accepted;  
38: If (Cont<2) then
39: Return The Web service is refused;  
40: End if
41: End

Figure 8. Comparison and filtering algorithm

6. Simulation and Results
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 tests. Because if we do tests 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. [9].

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

6.2.1 Aglets platform:
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 [10].

6.2 Performances and results
6.2.1 Simulation configuration:
To evaluate the performances of the proposed system we will study the following scenario:

the scenario consist of creation of one Cloud computing service provider (CCSP) that contain three datacentres, the first datacentre represent the region A of Cloud and the rest of datacentres represent the region B (because this region will execute the matching algorithm, which require more resources). Then we create six tasks (representing the six principle tasks, to treat the client request in the proposed discovery system), then we test the impact of number of Virtual machines (VM) on the global time execution (response time) and the Global cost (cost of using the proposed system).

This scenario allows us to evaluation, the internal performances of the proposed system, which mean, the evaluation of using this one in each one cloud computing region.

The table 1 shows the main tasks characteristics:

Table 1. Tasks characteristics.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Length</th>
<th>File Size</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>key words extraction</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Creation of transport and interface agent</td>
<td>80</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Creation of Researcher agents</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Storage of web services descriptions</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Matching algorithm execution</td>
<td>4000</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Creation of transport agents</td>
<td>40</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

6.2.2 Simulated scenario and analysis of results:

We will vary the number of virtual machines from 1 to 6, then we submit (equitably) the tasks (In Cloudsim the tasks are represented by cloudlet class) to the virtual machines, the table 2 show the experiences results:

The experiences show that the internal functioning of the proposed system enhance the response time, due to:

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

Which confirm the efficiency of the proposed policy, which 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 the proposed system benefit largely from the virtualization characteristic of Cloud computing technology).

But the normal consequence to increasing the number of VM is the increasing of the Global cost, due, essentially to using of Ram in each VM.

We note also that, we was bring back, to add, more hosting machine, in the data centers of the last experience to allow the execution of the VMs.

The following figure depicts variation of response time:

Figure 9. Summary of response time results

6.3 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 displayed 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.

7. Conclusion and Future Work

In this paper we have presented a new Cloud computing, framework based on mobile agents for web services discovery and selection, the proposed framework is based on: a new cloud computing architecture that benefit from the characteristics of mobile agents, and a new matching algorithm.

The experiences show the efficiency of the proposed virtualization policy (that consist of maximizing the number of virtual machines to treat one request), in the global time execution,

The proposed system can be applied, directly by, simple clients or by companies, as a cloud service, or in the operation of web services composition, to accomplish complex tasks over internet.

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

- Finally we will test the proposed system, using CloudAnalyst: to evaluate, also, the external performances of the proposed system, between the Cloud computing regions [11], not only for a single user, but on a large user bases.

References