Journal of Theoretical and Applied Information Technology
28th February 2014. Vol. 60 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SKYLINE IN CLOUD COMPUTING

[1] **ABDELLAH IDRISSI,** [2] **MANAR ABOUREZQ**
Research Computer Sciences Laboratory (LRI)
Computer Sciences Department, Faculty of Sciences
University Mohammed V - Agdal, Rabat
E-mail: [1] idriab@gmail.com , [2] manar.abourezq@gmail.com

ABSTRACT

The cloud computing technology is booming. The utility of this technology is no longer to show. In this paper, we investigate the problem of search and selection systems allowing users to search through Cloud services and find the ones that best meet their needs. In this context, we propose a new algorithm to address this problem. This algorithm is based on the principle of the Skyline. One of the main contributions of our work is the construction of a Web Agent using the Skyline method to determine which Cloud services best meet users' requirements. In this work, we expose our algorithm and present some experimental results showing that our approach is very promising.
Keywords: *Cloud Computing, Cloud Services, Skyline, Block-Nested Loops Algorithm.*

1. INTRODUCTION

Cloud computing has emerged as one of the new technologies that will reshape the way enterprises function in the near future [1]. Its goal is to replace the local use of computers with a centralized use where resources such as networks, servers, storage space, applications, and services are stored, used and managed by a third-party in a way that is transparent for end-users. It has rapidly evolved with big IT companies developing their own solutions, such as Amazon's Elastic Compute Cloud [2], Google's App Engine [3], IBM's Blue Cloud [4]…

The concept of Cloud Computing is not new. In 1960, John McCarthy predicted that « *Computing may someday be organized as a public utility just as the telephone system is a public utility* » [5]. In the 90s, the term « Grid » was coined to refer to technologies that allow on-demand use of computing resources. However, the use has evolved since the needs have shifted from treatment power to on-demand services, which are offered by Cloud Computing. Thus, Cloud Computing can be seen as an evolution of Grid Computing [6].

Cloud Computing can be defined as being a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources [7]. These resources can be provisioned and released in a rapid and simple way.

Every Cloud system has the following essential characteristics:

– It is a shared system that uses virtualization to offer a set of physical and virtual resources such as networks, servers, storage space, bandwidth, applications…;
– It is a system that is dynamically configurable, which makes it easy to expand or decrease depending on the user's needs, without affecting the level of reliability and security;
– It is a system that is accessible via a network, usually the Internet, from various machines (computers, smart phones, tablets, PDAs…) using standard APIs;
– It is a system that uses specific measure systems to control and optimize the use of resources and to offer a billing based on what was consumed, without surplus or need of managing the underlying infrastructure.

The services reachable via Cloud may be divided into three categories [8]: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Each one of these categories has specific characteristics that make it more adapted to certain use cases.

SaaS [9] allows users to remotely access applications that run in the Cloud's infrastructure by using thin or thick clients. Thus, there is no need to invest in an infrastructure or to buy software licenses. For providers, costs of installation, hosting and maintenance are optimized since many users access to the same application. Examples of SaaS include Google Drive [10] (formerly Google Docs) and Salesforce CRM [11].

PaaS [12] offers a software layer or a development environment as a service on which users will build and deploy their own applications. That way, users won't need to manage the infrastructure while keeping control of the deployed applications and configuring the hosting environment. Examples of PaaS include Salesforce's Force.com [13], Google App Engine [14] and Microsoft Windows Azure [15].

IaaS [16] provides as a service basic storage and computing resources such as servers, network equipments, data warehouses… These resources will be used to run users' own applications. Usually, IaaS satisfies best the end-users' needs of interoperability and portability [17] because they choose the various blocks that compose the infrastructure used. Examples of IaaS include Amazon Elastic Compute Cloud [2] and Microsoft SQL Azure [15].

Cloud services can be deployed in various models [18], depending on the use case, the provider's business model... The most widespread deployment models are Public, Private, Community and Hybrid.

A Public Cloud [18] is an open Cloud provided by an organization to the general public. It can be accessed via a network, usually the Internet. However, the fact that the Cloud is public doesn't imply that services are offered for free or that the data exchanged by its means is not confidential.

A Private Cloud [18] is offered to the sole use of one organization that either manages it or delegates its management to a third-party. The main advantage of this deployment model is that there are no limitations regarding bandwidth or security, since the resources are exclusively used by the organization.

A Community Cloud [17] is a Cloud shared by organizations belonging to the same community. They can manage their Cloud themselves or delegate the chore to a third-party.

A Hybrid Cloud [19] contains two or more of the Clouds above interconnected by standard or proprietary technologies.

In addition to these four deployment models, new ones are emerging, like the On-Site Private Cloud [17] and the Special Purpose Cloud [20].

The On-Site Private Cloud is a Cloud intended for the private use of a sole organization, just like the Private Cloud. However, it is hosted by the organization, either in a centralized or distributed way. The security aspect is also managed by the organization.

The Special-Purpose Cloud provides, on top of standard resources, additional methods regarding specific use cases. An example that illustrates this model is Google's App Engine with the specific capacities it offers to document management.

Using a Cloud service presents many advantages to end-users, such as:

– Cost reduction: since users purchase only the resources they need, without surplus, they don't need to invest in infrastructure or maintenance;
– Ubiquitous access: instant and uninterrupted access to computing and storage resources is granted to any user who has a network connected machine;
– Scalability: users can easily adapt the available resources to their specific needs;
– Capacity: users can add resources as required.

We are interested, in this work, in the search and selection of cloud services. This research area has been subject to many contributions [21, 22, 23, 24, 25]. In the same way, we propose, in this paper, a new method which allows Cloud users to find a Cloud service that meets their requirements. Our approach is based on the principle of the Skyline [26]. One of this work's main contributions is building an Agent that uses the Skyline to determine which Cloud services best meet the users' requirements.

This paper is organized as follows. We expose, in the next section, some related works. In section 3, we present some principles of the Skyline method. Then, we propose our prototype of a Cloud Service Research and Selection System in section 4. We expose the algorithm we used in section 5. In section 6, we develop and present a proof-of-concept of our system and finally we conclude in section 7.

2. RELATED WORK

With the increase use of Cloud Computing, one of the major needs today is to have a system that allows searching among various Cloud services to select the ones that best match users' requirements. There are several studies in the literature which deal with this subject like [21, 22, 23, 24, 25].

Kang and Sim [21] presented a Cloud portal with a Cloud service search engine based on similarity. The user specifies the 3 types of requirements of the Cloud services they are looking for, namely functional requirements (category of service), technical requirements (OS, CPU, memory, storage space...) and cost requirements (price and timeslot range). Then the search engine consults the adopted Cloud ontology to calculate an aggregated similarity and returns the list of matching Cloud services ordered by this similarity.

In another work [22], Kang and Sim presented Cloudle, a search engine that carries three main functionalities, which are query processing, similarity reasoning and rating. Cloudle is based on the same principle seen in [21] as it consults a Cloud ontology to compute the similarity between Cloud services and returns a list of results sorted by aggregated similarity.

Han and Sim [23] built a Cloud Service Discovery System (CSDS) that consults a Cloud ontology to compute the similarity between Cloud services and return a list of results matching the user's query.

Yoo et al. [24] present a resource selection service based on Cloud ontology. Its main objective is to search and select virtualized resources that answer users' requirements. The resource selection service uses a Cloud ontology to virtualize physical resources and generate new Virtual Ontologies (VOns). These VOns are combined into new resources for which a degree of similarity is computed to determine the ones that meet best the user's requirements.

Zeng et al. [25] propose a service matching algorithm and a service composition algorithm to search through Cloud services and compute the semantic similarity between them, the main goal being to determine whether two given Cloud services are interoperable. The resulting Cloud services are ranked based on QoS information.

Although these works have tackled the question of research and selection of Cloud services, most of them chose to use similarity [27] to determine which Cloud service is the most similar to the user's quest.

Similarity is used to determine the degree to which two Cloud services are alike by decomposing them into concepts and comparing these concepts among them [28]. A Cloud service may be represented as a node having many children nodes, which are the concepts. These concepts have also many children nodes. Thereby, to determine the similarity between two concepts, we calculate the number of parent nodes they have in common [23].

Furthermore, these works allow users to specify the requirements they want the Cloud services to match. However, we think that these requirements, especially the technical ones, need to be split into two categories: fixed (OS, Provider…) and variable (CPU, Memory, storage space...). When a user searches for a Cloud service, they usually would like to have the best possible value of the variable technical requirement (such as the maximum memory) with the minimum cost. That is why, instead of using them as fixed requirements, we optimize them by using them as dimensions in the Skyline.

There's also the need to specify, for each cloud service, which industry it is meant for (Education, Enterprise, Healthcare, Legal, Finance…) and under which category it falls (Email, CRM, Human Resources…). This helps the search to be more relevant.

Another concern is, since there are no Cloud computing standards yet, especially regarding ontology, each work uses its own defined ontology. The main risk is that of having to rebuild the systems presented if/when a standard unified ontology is adopted [29].

The research and selection of a Cloud service among a set of Cloud services is a preference problem. To deal with this problem, we propose, in this paper, a new approach based on the principle of the Skyline [26]. Using the Skyline allows the user to specify the criteria they want to optimize and to get the Cloud services that are not dominated by any other Cloud service, that is to say Cloud services for which there exists no better Cloud service for all the criteria specified. We present hereafter some principles of the Skyline.

3. SKYLINE

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The Skyline [26] was introduced to meet the needs of users desiring to select a set of points that optimize their requirements from a large set of data. Each point contained in the Skyline is not dominated by any other point, thus being better than all the points not contained in the Skyline for at least one criterion, and being equal to or better than them for all the other criteria. A criterion used by the Skyline is called dimension.

For example, if the user is looking to rent a car at the minimum price with the maximum engine power, the Skyline will contain all the cars that are not dominated by any car outside of the Skyline. In other words, for each car returned in the Skyline, there is no car outside the Skyline that is better than it in both dimensions. Thus, a user will find their favorite car in the Skyline, no matter how they weight their preferences toward the dimensions.

There are two major ways to compute the Skyline [26]. One is to extend existing database systems with the logical Skyline operator [26]. The other is to use algorithms.

The extension of existing databases is, we think, an intuitive way to compute the Skyline. It consists of using standard SQL instructions and extending them with a new clause, SKYLINE OF [26], which can be translated into nested loops, as shown in Figure 1.

```
SELECT * FROM carsTable as table1
WHERE carsTable.constructor = 'Constructor 1' AND NOT EXISTS
        (SELECT * FROM carsTable as table2 WHERE table2.price <= table1.price
        AND table2.enginePower >= table1.enginePower
        AND (table2.price < table1.price OR table2.enginePower > table1.enginePower))
```

Figure 1: Example of SQL queries to compute the Skyline.

This method, although simple, has the inconvenience of using loops, which leads to having very complex SQL queries, especially when the number of Skyline's dimensions is high. This complexity results in a poor performance and an additional computational cost.

Another way to compute the Skyline is by using algorithms. The advantage of using algorithms is that they can be applied to compute any Skyline, no matter how many dimensions it has. Many algorithms may be used such as the Block-Nested Loops algorithm (BNL) [26], the Divide and Conquer algorithm (D&C) [30, 31], B-Tree [32], etc.

In our approach, we used the BNL algorithm. We think that it is the best in our case. The BNL algorithm consists of comparing tuples among them to determine the ones that are not dominated by any other. It is done by keeping dominating tuples in the main memory and by comparing each new tuple to them. In each iteration, a new tuple is read from the input list of tuples. If the new tuple is dominated by one of them, it is eliminated. If it dominates a tuple in the list, the dominated tuple is eliminated, and the new tuple is added to the list to be compared to future tuples. If the new tuple is incomparable, which means that it is neither dominated by nor dominating any tuple in the main memory, it is added to the list.

At the end of all iterations, only tuples that are not dominated by any other tuple are kept in the main memory. These tuples are part of the Skyline.

The BNL algorithm has a high performance, especially if the Skyline is small. Its complexity [33] varies between O(n) in the best case and $O(n^2)$ in the worst case, n being the length of the input tuples' list.

Our approach is based on this algorithm. It involves the introduction of several agents. These agents represent a prototype of a Cloud Service Research and Selection System consisting of a user interface, a user's query processing agent, a pre-processing Skyline agent, a cloud services research and selection agent and a database. We present this prototype in the next section.

4. A CLOUD SERVICE RESEARCH AND SELECTION SYSTEM (CSRSS)

As mentioned above, the prototype of the Cloud Service Research and Selection System consists of a user interface, a user's query processing agent, a pre-Skyline processing agent, a Cloud services research and selection agent and a database. It is illustrated in Figure 2.

Journal of Theoretical and Applied Information Technology
28th February 2014. Vol. 60 No.3
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

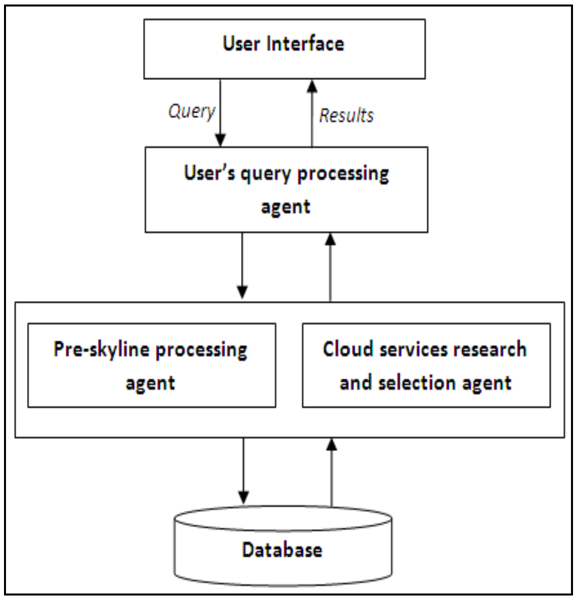

User Interface
Query
Results
User's query processing agent
Pre-skyline processing agent
Cloud services research and selection agent
Database

Figure 2: A schema representing the Cloud Service Research and Selection System

The user's interface allows users to interact with the system by selecting the requirements the Cloud services must meet and view the returned results. It also allows the users to add Cloud services by filling in their attributes such as the name, the provider, the bandwidth, the OS, etc. We think that these requirements are the common ground to existing and upcoming Cloud ontologies [22, 23, 24, 29, 34].

The user's query processing agent extracts the requirements contained in the user's request and sets them into two categories (see tables 1 and 2):

– Requirements that are fixed, such as the provider's name, the service model, the OS…;

– Requirements that are to be optimized, such as the price (to be minimized) and the bandwidth (to be maximized). These requirements will be used as the Skyline's dimensions.

The Cloud Services Research and Selection Agent (CSRSA) connects to the database and executes a SQL query, which predicates are the fixed requirements returned as a result by the user's query processing, to select all the Cloud services that meet these fixed requirements.

Requirement	Value
Provider	Microsoft IBM Amazon…
Service Model	IaaS PaaS SaaS
OS Serie	Windows Mac Unix…
OS Distribution	Windows XP Windows Vista Windows 7 Linux…
CPU Manufacturer	Intel IBM AMD…
CPU Gamme	Pentium Intel 64…
Industry	General Education Healthcare…
Category	General CRM E-procurement…

Table 1: Example of fixed requirements

The Pre-Skyline Processing Agent (PSPA) prepares the results extracted from the database by the CSRSA for the running of the Skyline operator. The Cloud services returned and their dimensions are stored as tuples. The dimensions used are the user's requirements that are not "fixed", and thus are to be optimized, such as the price (to be minimized), the bandwidth (to be maximized), the network latency (to be minimized)…

Table 2: Value range of the dimensions used in the Skyline

Dimension	Range Value
Storage space	0.14 – 3999.98
Memory	128 – 16000
Bandwidth	0 – 10
Latency	0 – 10000
Price	1 – 2000
CPU speed	50 – 3060

The CSRSA uses the Skyline, on the set of tuples returned by the PSPA, to determine which Cloud services are in the Skyline and meet the

Journal of Theoretical and Applied Information Technology
28th February 2014. Vol. 60 No.3
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

user's preferences. We present hereafter the algorithm.

5. ALGORITHM

As seen previously, the CSRSA uses the Skyline, on the set of tuples returned by the PSPA, to determine which Cloud services are in the Skyline and meet the user's preferences. To do so, the agent uses the BNL algorithm as showed in Figure 3. Every tuple p is an n-dimension tuple. The dimensions are stored in the list L_D in the same order they compose the tuples. For each dimension, an indication is given whether it is to be minimized, maximized or different.

```
– L_P : input list of tuples for which the Skyline is
        to be computed
– L_D: input list of dimensions
– p, q: tuples
– L_S : output list of the tuples forming the
        Skyline

Function ComputeSkyline
   Foreach p in L_P do
         If L_S = Ø Then
         L_S = {p}
         Else
            Foreach q in L_S – {p} do
                result = Compare (p, q, L_D)
                If result = count (L_D) then
                     L_S = L_S + {p} – {q}
               Elseif result # 0 and q is the last
                        tuple in Ls then
                        L_S = L_S + {p}
                Else
                     Goto (*)
               End IF
            End Foreach
      (*) End If
   End Foreach
 Return L_S
End Function
```

Figure 3. The algorithm used to compute the Skyline

The function **Compare (p, q, L_D)** is the core of the algorithm. It compares the tuples p and q in all the dimensions in the list L_D. The result returned varies between 0 (when q dominates p) and n (when p dominates q), n being the number of dimensions. Any other result in this range means that p and q are not comparable. In the next section, we present the implementation of the algorithm and its performance.

6. EXPERIMENTATION AND RESULTS

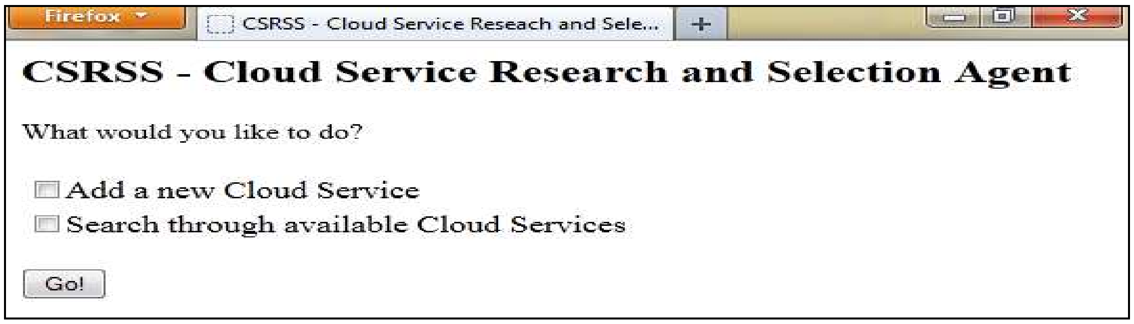
The platform we used for the experiments is an HP workstation with a 3.30 GHz processor, 4 GB of main memory, Windows Server 2007 as operating system and MS SQL Server 2008 as DBMS. The algorithm is implemented using ASP.net to obtain a web-based system that can be accessed from any web client anytime the user is connected to the Internet.

6.1 CSRSS Interface

The CSRSS start page, as shown in Figure 4, allows the user to either add a new Cloud service to the database or search for Cloud services that match their requirements.

If the user chooses to add a new Cloud Service, they are taken to another page where they first enter the name of the Cloud service in question so a search can be made to make sure that it doesn't already exist in the database. Afterwards, the user enters the different information such as the Cloud service's provider, model (IaaS, PaaS or SaaS), industry, memory, price... This insert page is shown in Figure 5.

If the user checks the second option (Search through available Cloud Services like shown in figure 4), they are taken to the CSRSS page that allows to make an advanced search through the database and to compute the Skyline of the returned results.

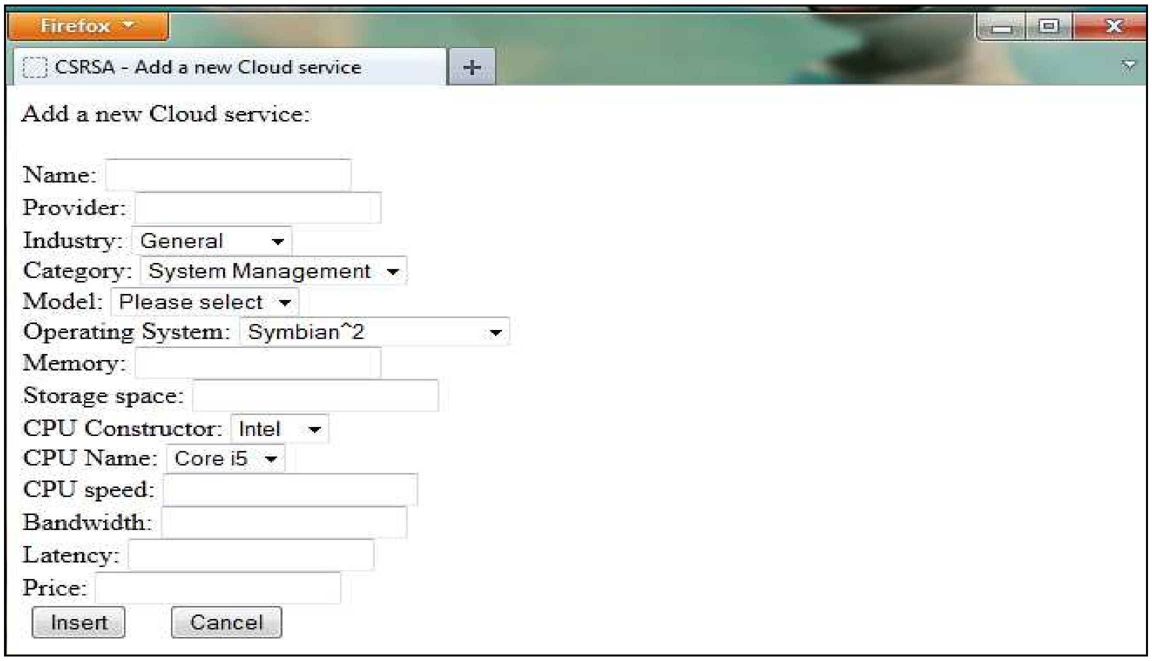


CSRSS - Cloud Service Research and Selection Agent

What would you like to do?

☐ Add a new Cloud Service
☐ Search through available Cloud Services

Figure 4: The CSRA start page

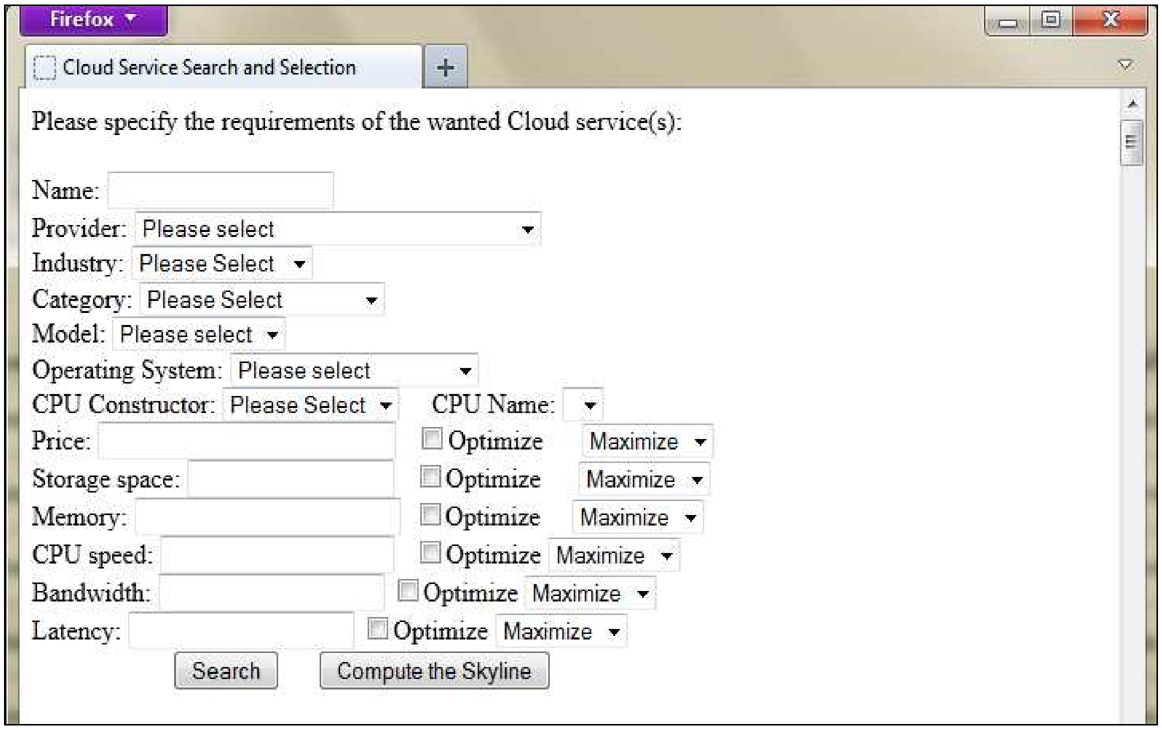


Add a new Cloud service:

Name:	
Provider:	
Industry:	General
Category:	System Management
Model:	Please select
Operating System:	Symbian^2
Memory:	
Storage space:	
CPU Constructor:	Intel
CPU Name:	Core i5
CPU speed:	
Bandwidth:	
Latency:	
Price:	

Figure 5: The CSRA page to add a new Cloud service

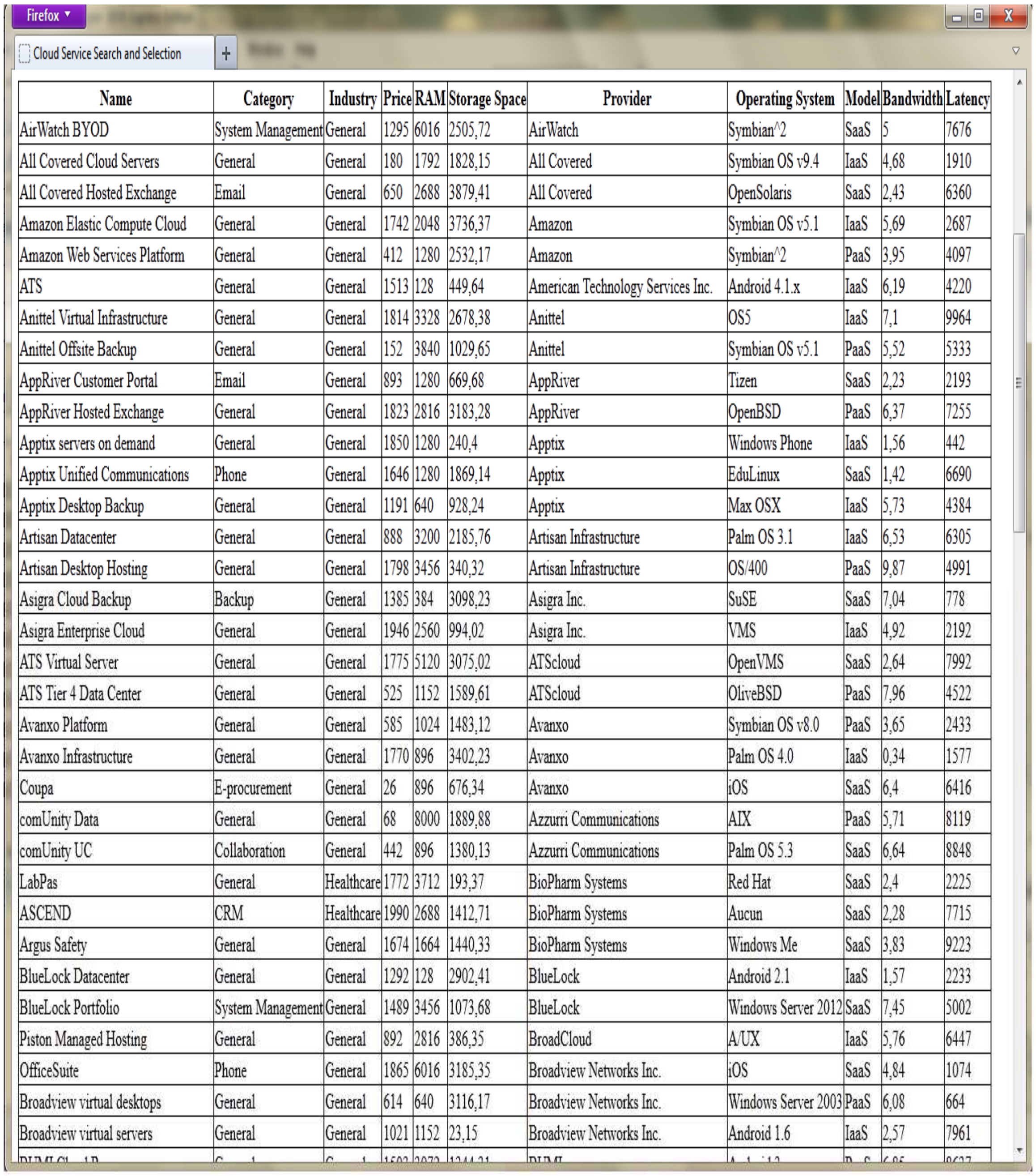
The user can fill out one or many information about the Cloud service(s) they are searching for, such as illustrated in Figure 6. For information such as price, memory, storage space, bandwidth... they can either give a specific value or specify that they are the dimensions to be used when computing the Skyline. For each dimension, the user specifies if it is to be minimized, maximized or different. The results are returned in a table as shown in Figure 7.



Please specify the requirements of the wanted Cloud service(s):

Name:
Provider: Please select
Industry: Please Select
Category: Please Select
Model: Please select
Operating System: Please select
CPU Constructor: Please Select CPU Name:
Price: Optimize Maximize
Storage space: Optimize Maximize
Memory: Optimize Maximize
CPU speed: Optimize Maximize
Bandwidth: Optimize Maximize
Latency: Optimize Maximize

Figure 6: The CSRA Search And/Or Computation Of The Skyline Page


Cloud Service Search and Selection

Name	Category	Industry	Price	RAM	Storage Space	Provider	Operating System	Model	Bandwidth	Latency
AirWatch BYOD	System Management	General	1295	6016	2505,72	AirWatch	Symbian^2	SaaS	5	7676
All Covered Cloud Servers	General	General	180	1792	1828,15	All Covered	Symbian OS v9.4	IaaS	4,68	1910
All Covered Hosted Exchange	Email	General	650	2688	3879,41	All Covered	OpenSolaris	SaaS	2,43	6360
Amazon Elastic Compute Cloud	General	General	1742	2048	3736,37	Amazon	Symbian OS v5.1	IaaS	5,69	2687
Amazon Web Services Platform	General	General	412	1280	2532,17	Amazon	Symbian^2	PaaS	3,95	4097
ATS	General	General	1513	128	449,64	American Technology Services Inc.	Android 4.1.x	IaaS	6,19	4220
Anittel Virtual Infrastructure	General	General	1814	3328	2678,38	Anittel	OS5	IaaS	7,1	9964
Anittel Offsite Backup	General	General	152	3840	1029,65	Anittel	Symbian OS v5.1	PaaS	5,52	5333
AppRiver Customer Portal	Email	General	893	1280	669,68	AppRiver	Tizen	SaaS	2,23	2193
AppRiver Hosted Exchange	General	General	1823	2816	3183,28	AppRiver	OpenBSD	PaaS	6,37	7255
Apptix servers on demand	General	General	1850	1280	240,4	Apptix	Windows Phone	IaaS	1,56	442
Apptix Unified Communications	Phone	General	1646	1280	1869,14	Apptix	EduLinux	SaaS	1,42	6690
Apptix Desktop Backup	General	General	1191	640	928,24	Apptix	Max OSX	IaaS	5,73	4384
Artisan Datacenter	General	General	888	3200	2185,76	Artisan Infrastructure	Palm OS 3.1	IaaS	6,53	6305
Artisan Desktop Hosting	General	General	1798	3456	340,32	Artisan Infrastructure	OS/400	PaaS	9,87	4991
Asigra Cloud Backup	Backup	General	1385	384	3098,23	Asigra Inc.	SuSE	SaaS	7,04	778
Asigra Enterprise Cloud	General	General	1946	2560	994,02	Asigra Inc.	VMS	IaaS	4,92	2192
ATS Virtual Server	General	General	1775	5120	3075,02	ATScloud	OpenVMS	SaaS	2,64	7992
ATS Tier 4 Data Center	General	General	525	1152	1589,61	ATScloud	OliveBSD	PaaS	7,96	4522
Avanxo Platform	General	General	585	1024	1483,12	Avanxo	Symbian OS v8.0	PaaS	3,65	2433
Avanxo Infrastructure	General	General	1770	896	3402,23	Avanxo	Palm OS 4.0	IaaS	0,34	1577
Coupa	E-procurement	General	26	896	676,34	Avanxo	iOS	SaaS	6,4	6416
comUnity Data	General	General	68	8000	1889,88	Azzurri Communications	AIX	PaaS	5,71	8119
comUnity UC	Collaboration	General	442	896	1380,13	Azzurri Communications	Palm OS 5.3	SaaS	6,64	8848
LabPas	General	Healthcare	1772	3712	193,37	BioPharm Systems	Red Hat	SaaS	2,4	2225
ASCEND	CRM	Healthcare	1990	2688	1412,71	BioPharm Systems	Aucun	SaaS	2,28	7715
Argus Safety	General	General	1674	1664	1440,33	BioPharm Systems	Windows Me	SaaS	3,83	9223
BlueLock Datacenter	General	General	1292	128	2902,41	BlueLock	Android 2.1	IaaS	1,57	2233
BlueLock Portfolio	System Management	General	1489	3456	1073,68	BlueLock	Windows Server 2012	SaaS	7,45	5002
Piston Managed Hosting	General	General	892	2816	386,35	BroadCloud	A/UX	IaaS	5,76	6447
OfficeSuite	Phone	General	1865	6016	3185,35	Broadview Networks Inc.	iOS	SaaS	4,84	1074
Broadview virtual desktops	General	General	614	640	3116,17	Broadview Networks Inc.	Windows Server 2003	PaaS	6,08	664
Broadview virtual servers	General	General	1021	1152	23,15	Broadview Networks Inc.	Android 1.6	IaaS	2,57	7961

Figure 7: The table of results returned by the CSRA

6.2 Performance

We generated over 50000 Cloud services with random values for each dimension within the ranges specified in section 4. We executed our program varying the size of the input from 100 to 50000 cloud services, and the number of dimensions from 1 to 6. We then measured the execution time and the size of the Skyline. The results are represented in Figure 8 and Figure 9.

The execution time doesn't vary much when the number of dimensions is less than 3 or the size of the input is less than 10000. The maximum execution time is 25 s when computing a 6-dimensional Skyline for 50000 Cloud services. As for the Skyline size, it is rather small compared to the input size and tends to converge for all sizes once the number of dimensions is more than 5.

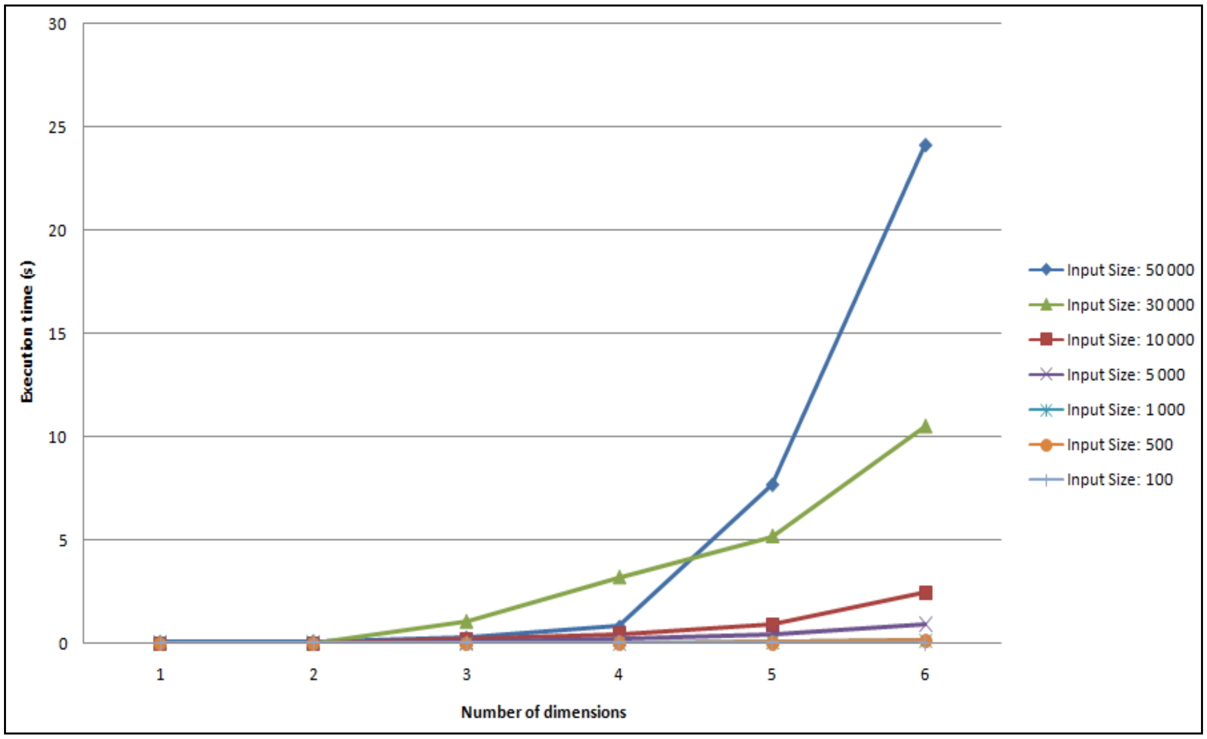


Figure 8: Execution Time / Number Of Dimensions For Different Input Sizes

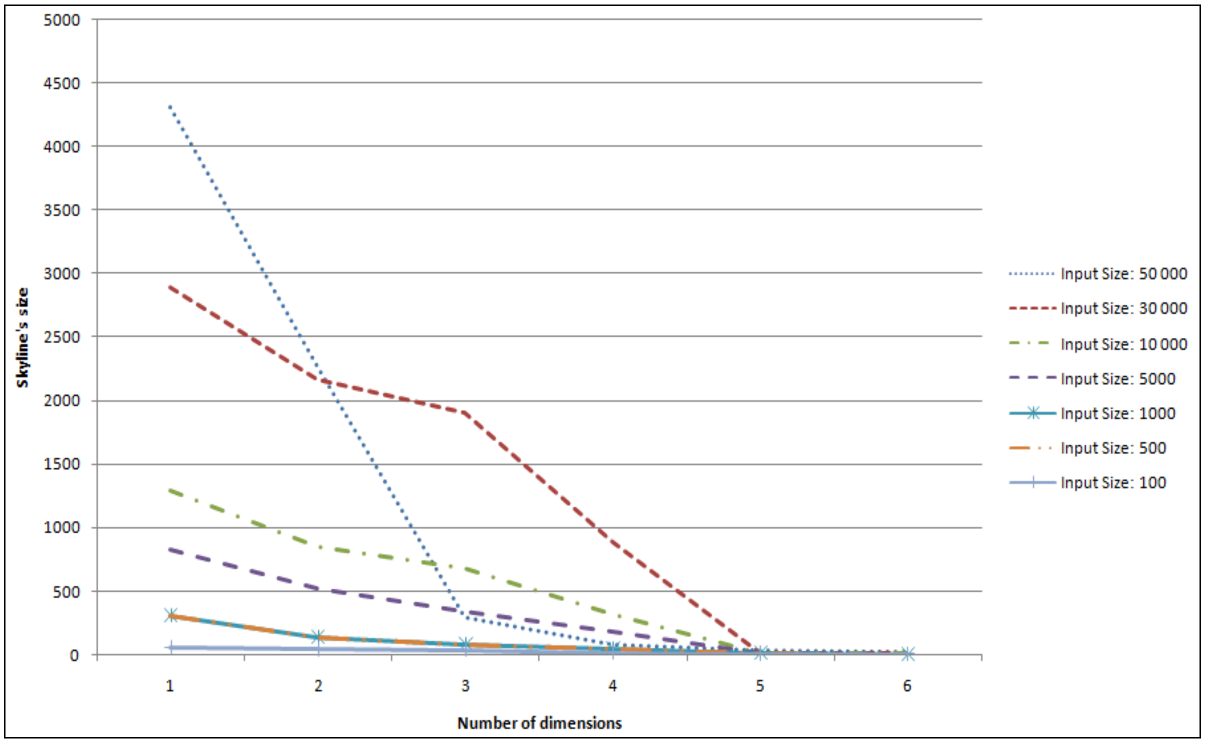


Figure 9: Skyline's Size / Number Of Dimensions For Different Input Sizes

7. CONCLUSION

In this work, we have developed an algorithm which allows searching and selecting Cloud services that meet the user's requirements. Our approach is based on the BNL Skyline algorithm. The experimental results show that with our method we can process a large volume of data Skyline in less than 25 s. We can conclude that our approach gives very promising results. Note that our algorithm is general and can be adapted to any similar problem.

REFERENCES

[1] "Gartner's top 10 Strategic Technology Trends for 2013", Gartner, October 2012
[2] Amazon, http://aws.amazon.com/fr/ec2/
[3] Google, https://appengine.google.com
[4] IBM, http://www.ibm.com/cloud-computing
[5] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", IEEE Grid Computing Environments Workshop, IEEE Press, 2008
[6] I. Foster, "There's Grid in them thar Clouds", January 2008, http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html
[7] P. Mell and T. Grance, "The NIST definition of cloud computing", NIST special publication, 2011
[8] A.Fox, G. Rean, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing", Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28, 2009
[9] L. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition", ACM SIGCOMM Computer Communication Review, Vol. 39, Number 1, January 2009
[10] Google Drive, https://drive.google.com/
[11] Salesforce, http://www.salesforce.com/
[12] D. Cheng, "PaaS-onomics: A CIO's Guide to using Platform-as-a-Service to Lower Costs of Application Initiatives While Improving the Business Value of IT", Tech. rep., LongJump, 2008
[13] Force, http://www.force.com
[14] Google App Engine, https://appengine.google.com/
[15] Windows Azure, http://www.windowsazure.com
[16] L. Karadsheh, "Applying security policies and service level agreement to IaaS service model to enhance security and transition", Computers & Security, Vol. 31, Issue 3, May 2012, pp. 315-326
[17] S. Radack, "Cloud Computing: A Review of Features, Benefits, and Risks, and Recommendations for Secure, Efficient Implementations", NIST, ITL Bulletin, June 2012.
[18] S. Rao, N. Rao and E. Kusuma Kumari, "Cloud Computing: An Overview", Journal of Theoretical and Applied Information Technology, Vol. 9, No. 1, November 2009
[19] K. Sims, "IBM Blue Cloud Initiative Advances Enterprise Cloud Computing", 2009
[20] K. Jeffery and B. Neidecker-Lutz, "The future of Cloud Computing", European Commission, Information Society and Media
[21] J. Kang and K. M. Sim, "A Cloud Portal with a Cloud Service Search Engine", International Conference on Information and Intelligent Computing IPCSIT, Vol.18, 2011
[22] J. Kang and K. M. Sim, "Cloudle : An Agent-based Cloud Search Engine that Consults a Cloud Ontology", Cloud Computing and Virtualization Conference, 2010
[23] T. Han and K. M. Sim, "An Ontology-enhanced Cloud Service Discovery System", IMECS 2010 Vol. 1, March 17 – 19 2010, Hong Kong
[24] H. Yoo, C. Hur, S. Kim, and Y. Kim, "An Ontology-based Resource Selection Service on Science Cloud", International Journal of Grid and Distributed Computing, Vol. 2, No. 4, December 2009
[25] C. Zeng, X. Guo, W. Ou and D. Han, "Cloud Computing Service Composition and Search Based on Semantic", Cloud Computing, Vol. 5931, 2009, pp. 290-300
[26] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline operator", International Conference on Data Engineering (ICDE), 2001
[27] P. Resnik, "Semantic similarity in a taxonomy: an information-based measure and its application to problem of ambiguity in natural language", Journal of Artificial Intelligence Research, Vol. 11, 1999
[28] T. Andreasen, H. Bulskov and R. Kanppe, "From Ontology over Similarity to Query Evaluation", 2nd International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems

(ODBASE), 3-7 November 2003, Catania, Sicily, Italy
[29] L. Youseff, L. Butrico and M. Da Silva, “Toward a Unified Ontology of Cloud Computing”, Grid Computing Environments Workshop, November 2008
[30] H. Kung, F. Luccio, and F. Preparata, “On finding the maxima of a set of vectors”, Journal of the ACM, Vol. 22, Issue 4, October 1975
[31] F. Preparata and M. Shamos, “Computational Geometry: An Introduction”, Springer-Verlag, New York, Berlin, etc., 1985
[32] D. Comer, “The Ubiquitous B-Tree”, ACM Computing Surveys, Volume 11, June 1979
[33] L. Haas, M. Carey, M. Livny, and A. Shukla “Seeking the truth about ad hoc join costs”, The VLDB Journal, 1997
[34] D. Androcec, N. Vrcek and J. Seva, “Cloud Computing Ontologies: A Systematic Review”, MOPAS 2012, The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services, 2012, pp. 9-14.