A Cloud Services Research and Selection System

Manar ABOUREZQ
Research Computer Sciences Laboratory (LRI)
Computer Sciences Department, Faculty of Sciences
University Mohammed V - Agdal, Rabat
manar.abourezq@gmail.com

Abdellah IDRISSI
Research Computer Sciences Laboratory (LRI)
Computer Sciences Department, Faculty of Sciences
University Mohammed V - Agdal, Rabat
idriab@gmail.com

Abstract*—Cloud computing is a technology that has emerged in the last decade and that is transforming the IT industry. Our interest in this work is the research area of Cloud services search and selection systems. These systems allow Cloud users to search through Cloud services and find the ones that match their requirements. In this paper, we present a Cloud Service Research and Selection System based on an adaptation of the Skyline algorithm. The system selects Cloud services that best meet the users' requirements from a database of Cloud services. We also present some experimental results which approve our method.*

Keywords— Cloud Computing, Cloud Services, Skyline, Block-Nested Loops Algorithm

I. INTRODUCTION

Cloud computing is reshaping the way enterprises function [1]. It refers to the applications, hardware and datacenters delivered as services over the Internet [2] to replace the local use of computers with a centralized use. It is not a new concept: in 1960, John McCarthy predicted that « *Computing may someday be organized as a public utility just as the telephone system is a public utility* » [3].

Cloud Computing enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources [4]. These resources are stored, used and managed by a third-party in a way that is transparent for end-users. They can be provisioned and released in a rapid and simple way.

The services reachable via Cloud may be divided into three categories [2]: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Each one has specific characteristics and is more suited to certain cases.

SaaS [5] 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.

PaaS [6] 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.

IaaS [7] 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 [8] because they choose the various blocks that compose the infrastructure used.

Cloud services can be deployed in various models [9], depending on the use case, the provider's business model... The most widespread deployment models are Public [9], Private [9], Community [8] and Hybrid [10].

A Public Cloud [9] is an open Cloud accessible to the general public via a network (usually the Internet) and provided by an organization.

A Private Cloud [9] is reserved for the sole use of one organization that either manages it or delegates its management to a third-party.

A Community Cloud [8] is shared by organizations that belong to the same community and that can either manage it themselves or delegate it to a third-party.

A Hybrid Cloud [10] 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 [8] and the Special Purpose Cloud [11].

The On-Site Private Cloud [8] is, like the Private Cloud, intended for the private use of a sole organization. However, the later hosts it and manages the security aspect.

The Special-Purpose Cloud [11] provides, on top of standard resources, additional methods regarding specific use cases.

We are interested, in this work, in the research and selection of cloud services. This research area has been subject to many contributions [12, 13, 14, 15, 16]. 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 [17]. 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.

978-1-4799-3824-7/14/$31.00 ©2014 IEEE

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

II. 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 [12, 13, 14, 15, 16].

Kang and Sim [12] developed a Cloud portal that contains a search engine based on similarity. This engine uses the requirements specified by the user to consult the adopted Cloud ontology to calculate an aggregated similarity and return the list of matching Cloud services sorted by this similarity.

In another work [13], Kang and Sim presented Cloudle, a search engine based on the same principal seen in [12].

Han and Sim [14] built a Cloud Service Discovery System (CSDS) using a Cloud ontology to compute the similarity between Cloud services and return the list of matching Cloud services.

Yoo et al. [15] presented a resource selection service based on Cloud ontology to virtualize physical resources and combine them into new resources for which a degree of similarity is computed to determine the ones that best meet the user's requirements.

Zeng et al. [16] proposed a service matching algorithm and a service composition algorithm that search through Cloud services and determine the interoperability between two given Cloud services.

Although these works have tackled the question of research and selection of Cloud services, most of them chose to use similarity [18].

Similarity allows determining the degree to which two Cloud services are alike by decomposing them into concepts [19]. It represents each Cloud service 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 [14].

Furthermore, 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...). We use the later as dimensions [17] in the Skyline to optimize them.

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 having to rebuild the systems presented if/when a standard unified ontology is adopted [20, 21].

To deal with the problem of research and selection of a Cloud service among a set of Cloud services, we propose, in this paper, a new approach based on the principle of the Skyline [17]. 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.

III. SKYLINE

The Skyline [17] 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.

To compute the Skyline, we can extend existing database systems with the logical Skyline operator [17]. It is done by extending standard SQL instructions with a new clause, SKYLINE OF [17], which can be translated into nested loops. This method leads to very complex SQL queries, especially when the number of dimensions is large.

To deal with this problem, many algorithms was proposed like the Block-Nested Loops algorithm (BNL) [17], the Divide and Conquer algorithm (D&C) [22, 23], B-Tree [24], etc.

We use the BNL algorithm because we think it is the best one in our case. This algorithm has a high performance, especially if the Skyline is small. Its complexity [25] 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 and 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-Skyline processing agent, a Cloud services research and selection agent and a database. We present this prototype in the next section.

IV. THE CLOUD SERVICES RESEARCH AND SELECTION SYSTEM

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 as illustrated in Figure 1.

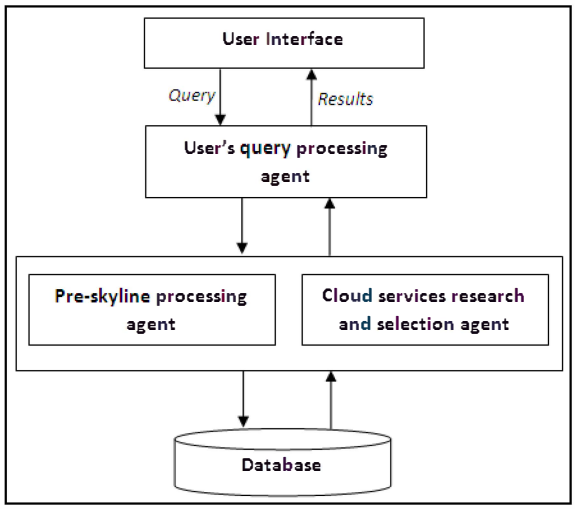

Fig. 1: A schema representing the Cloud Service Research and Selection System

The user's interface allows users to interact with the system. They select the requirements (name, provider, service model, bandwidth…) that Cloud services must meet and view the returned results via this interface. We chose the requirements that are the common ground to existing and upcoming Cloud ontologies [13, 14, 15, 20, 21].

The user's Query Processing Agent extracts the requirements contained in the user's request and sets them into two categories: fixed and variable as shown in table 1. The latter are to be optimized (minimized or maximized) and will be used as dimensions of the Skyline.

Type	*Requirement*
Fixed	Provider
	Service Model
	Industry
	Category
	OS Serie
	OS Distribution
	CPU Manufacturer
	CPU Gamme
Variable	Storage Space
	Memory
	Bandwidth
	Latency
	Price
	CPU Speed

Tab. 1. The requirements processed by the query processing agent

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.

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 variable requirements.

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 shown in Figure 2. Every tuple p is a 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 = ∅ 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
```

Fig. 2. 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.

V. 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 2007 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.

We generated over 50 000 Cloud services with random values for each dimension. We executed our program varying the size of the input from 100 to 50 000 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 3 and Figure 4.

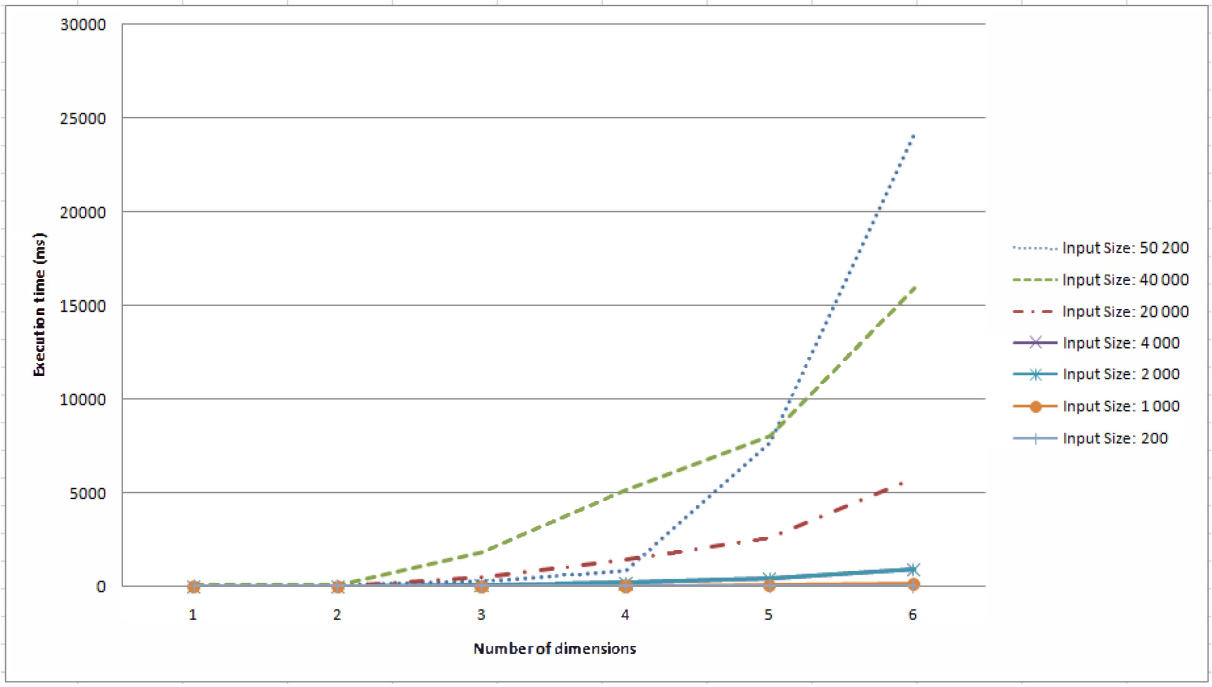

Figure 3: Execution time / number of dimensions for different input sizes

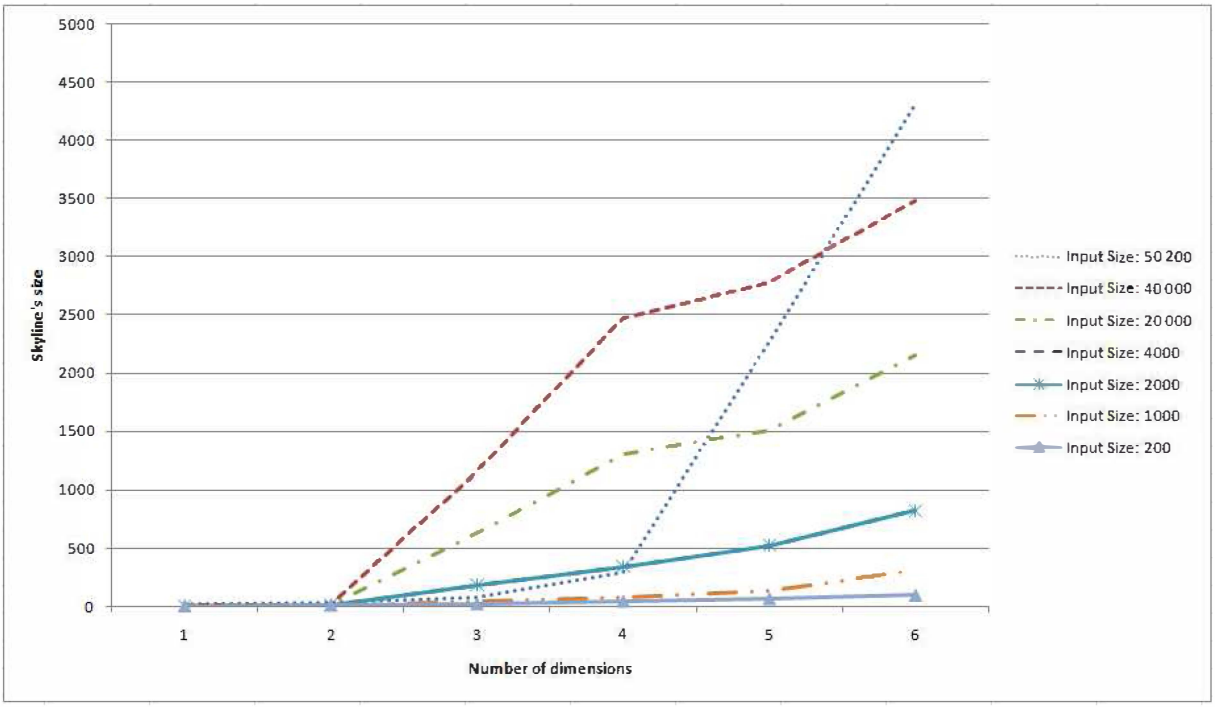

Figure 4: Skyline's size / number of dimensions for different input sizes

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 20 000. The maximum execution time is 24 s when computing a 6-dimensional Skyline for 50 200 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.

VI. CONCLUSION

In this work, we have developed a system that 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 in less than 25 s. We can conclude that our approach gives very promising results. Note that the algorithm we used 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] 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

[3] 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

[4] P. Mell and T. Grance, "The NIST definition of cloud computing", NIST special publication, 2011

[5] 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

[6] 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

[7] 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

[8] S. Radack, "Cloud Computing: A Review of Features, Benefits, and Risks, and Recommendations for Secure, Efficient Implementations", NIST, ITL Bulletin, June 2012.

[9] 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

[10] K. Sims, "IBM Blue Cloud Initiative Advances Enterprise Cloud Computing", 2009

[11] K. Jeffery and B. Neidecker-Lutz, "The future of Cloud Computing", European Commission, Information Society and Media

[12] 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

[13] J. Kang and K. M. Sim, "Cloudle : An Agent-based Cloud Search Engine that Consults a Cloud Ontology", Cloud Computing and Virtualization Conference, 2010

[14] T. Han and K. M. Sim, "An Ontology-enhanced Cloud Service Discovery System", IMECS 2010 Vol. 1, March 17 – 19 2010, Hong Kong

[15] 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

[16] 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

[17] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline operator", International Conference on Data Engineering (ICDE), 2001

[18] 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

[19] 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

[20] L. Youseff, L. Butrico and M. Da Silva, "Toward a Unified Ontology of Cloud Computing", Grid Computing Environments Workshop, November 2008

[21] 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

[22] 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

[23] F. Preparata and M. Shamos, "Computational Geometry: An Introduction", Springer-Verlag, New York, Berlin, etc., 1985

[24] D. Comer, "The Ubiquitous B-Tree", ACM Computing Surveys, Volume 11, June 1979

[25] L. Haas, M. Carey, M. Livny, and A. Shukla "Seeking the truth about ad hoc join costs", The VLDB Journal, 1997