User-Centric Cloud Service Broker

Hanan Elazhary
Computer Science Department,
Faculty of Computing & Information Technology,
King Abdulaziz University
Jeddah, Saudi Arabia
Computers and Systems Department,
Electronics Research Institute,
Cairo, Egypt

ABSTRACT
Cloud computing technology has gained enormous attention due to its promising capabilities such as virtualization, elasticity and the pay-per-use paradigm. Theoretically, cloud computing can offer Everything as a Service (XaaS). Selecting suitable cloud services matching the Quality-of-Service (QoS) requirements of the user is one of the prominent problems in the literature. A considerable number of research studies attempted to address this problem from different perspectives such as service discovery, service matching and ranking (against QoS requirements of the user) in addition to QoS evaluation and monitoring. In this paper, we argue that we need to integrate all those functionalities to help the cloud service user make more informed selection decisions. Accordingly, we propose a comprehensive user-centric Cloud Service Broker (CSB). We describe the architecture of this broker and discuss how it integrates and orchestrates the different required functionalities. We also discuss different possible methods to realize and implement each of its modules and pinpoint open points of research that need to be explored further. As a proof of concept, we present an example prototype implementation of CSB and discuss a case study using this prototype to justify the advantage of the integration. Towards this goal, we propose a novel evaluation-aware matching and ranking technique that integrates cloud services evaluation results with their matching and ranking against the user QoS requirements for more informed selections of suitable cloud services by taking into consideration the credibility of the cloud service providers.

General Terms
Distributed Systems

Keywords
Cloud Computing, Cloud Service Selection, Quality of Service

1. INTRODUCTION
Cloud computing is a relatively new technology that has emerged as a successor to Grid computing with a similar goal of providing computing services to end users analogous to other utilities such as water and electricity [1]. Nevertheless, the power of cloud computing lies in its inherent capabilities including virtualization (to offer virtual resources based on physical machines); the ease of configuring the resources by the end users; elasticity (where a user does not have to reserve a fixed amount of resources throughout the duration of the service, but variable amounts according to his/her applications usage) and the pay-per-use paradigm (where users pay only for the resources that they use). This is in addition to the convenient on-demand ubiquitous access to the cloud services anywhere and at any time [2].

The three prominent offerings of cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Nevertheless, this classical vision has been extended such that the cloud can provide Everything as a service (XaaS) [3]. This is especially important in fields such as Big Data where huge amounts of data need to be stored and processed [4] and mobile applications that suffer from the limited mobile resources and delegate a portion of the required processing power and storage to the cloud [5]. Cloud computing is also useful for Enterprise Resource Planning (ERP), E-learning and E-Government [6].

One of the prominent problems of cloud computing is the cloud service selection problem. Cloud service providers offer services with different attributes and variable capabilities and cost [7]. Additionally, each may partially match the Quality-of-Service (QoS) requirements of the user requiring a compromise. The fact that the number of service providers is enormous and is consistently increasing makes the task of searching, browsing and comparing the available services overly tedious. This is in addition to the fact that the different providers do not display information about their services in a uniform way and do not even use a unified terminology. Moreover, end users are not always professional enough to analyze the different offerings and make informed service selection decisions [8], which calls for a cloud service broker to help them with the selection.

This paper presents a literature review discussing different perspectives in the literature for addressing the problem. This is followed by an explanation of the architecture of a proposed comprehensive user-centric Cloud Service Broker (CSB) explaining its different modules and how it integrates and orchestrates the different functionalities. We discuss different possible ways to realize and implement each module and point out open points of research. As a proof of concept, to justify the advantage of the integration of several functionalities in such a comprehensive broker, we present an example implementation of a prototype based on CSB in addition to a case study using this prototype. The contributions of the paper can be summarized as follows:

- Pinpointing different aspects of the cloud service selection problem and presenting prominent research efforts in each.
- Designing the comprehensive user-centric cloud service selection broker CSB that integrates all the necessary functionalities in an orchestrated manner.
- Discussing different possible approaches to realize and implement each of the modules of CSB to highlight open points of research to stimulate further research studies.

- Presenting an example prototype implementation of the broker and a case study to justify the advantage of the integration of all the required functionalities in such a comprehensive broker.
- Proposing a modeling technique for the cloud services and for their evaluation.
- Proposing a novel evaluation-aware technique for integrating cloud services evaluation results with their matching and ranking (against the user QoS requirements) for more informed selection decisions by taking into consideration the credibility of the cloud service providers.

The rest of the paper is organized as follows: Section 2 presents related research in the literature. The architecture and example prototype implementation of CSB are presented in Sections 3 and 4 respectively. The case study based on this prototype is presented in Section 5. Finally, Section 6 presents the conclusion and summarizes the directions for future research and enhancements of CSB.

2. RELATED WORK

Several research studies in the literature have been concerned with the automated discovery and ranking of suitable cloud services matching the user QoS requirements. For example, Liu et al. [9] used agents to retrieve information about the candidate cloud services by accessing the Web services of the cloud service providers. They consulted an ontology to translate the user QoS requirements to be matched against the retrieved information. Consumer agents select the best service among the candidate ones using service ontology analysis. An alternate approach involves the use of search engines to discover the candidate cloud services. For example, Han and Sim [10] utilized Google search engine for this purpose. They consulted an ontology to match the user QoS requirements against the extracted information about those candidate cloud services determining the similarities and the equivalences. A semantic similarity measure was used to determine the best matching service. Gong and Sim [11] developed a centroid-based search engine that utilizes the k-means clustering algorithm for finding the best matching cloud services against the user QoS requirements. They used a similarity matrix to rank the discovered candidate services. Nevertheless, they only addressed the clustering process without real crawling of the cloud service providers Web pages.

The Service Measurement Index (SMI) framework (https://spark.adobe.com/page/PN39b/) specifies seven top-level attributes for comparing cloud services (accountability, agility, assurance, cost, performance, security & privacy, and usability). Each of those attributes is defined in terms of a set of low-level SMI attributes that are based on ISO standards by the Cloud Service Measurement Index Consortium (CSMIC). To adopt the SMI framework, a set of measurable Key Performance Indicators (KPIs) should be developed to evaluate the SMI attributes. Garg et al. [7] proposed a technique for comparing and ranking cloud services based on quantitative SMI attributes only. The user is allowed to specify relative weights for those attributes. Nevertheless, the authors did not consider qualitative attributes. Additionally, although they proposed monitoring the provided cloud services, they do not address this issue explicitly in the paper.

Several other research studies in the literature have been concerned with techniques for selection among a number of candidate cloud services against the user QoS requirements. For example, Rehman et al. [12] studied several Multi Criteria Decision Making (MCDM) techniques including Analytic Hierarchy Process (AHP) [13] for selecting one of thirteen candidate IaaS services using five service attributes. They showed that different techniques have different capabilities and hence do not make the same selection. Nevertheless, TOPSIS [14] is more suitable when the number of services is large because of its computational simplicity, while ELECTRE [15] and PROMETHEE [16] are better when the number of services is small and the number of service attributes is large. The authors concluded that this problem has to be investigated further. Whaiduzzaman et al. [17] provided a taxonomy and survey of some of the MCDM techniques that have been applied for cloud service selection in the literature. Sun et al. [18] made a similar survey but considered also optimization-based approaches, logic-based approaches and various other techniques such as using an ontology to map applications to suitable PaaS according to their requirements [19]. The authors identified types of services (IaaS, PaaS, SaaS or general) to which each technique has been applied. Ultimately, they made the same conclusion regarding the need for further research studies and identified some open research issues such as:

- Lack of a standard registry for cloud service publication, querying and rating; the Cloudservicemarket Website is the first platform that allows providers to publish their services for users to browse and rate.
- Lack of a specialized search engine for the automatic search and update of cloud service information.
- Lack of a standard normalization technique for cloud service attributes for uniform cloud services specifications.
- Lack of a practical technique for quantifying subjective opinions of the users regarding the weights of the different cloud service attributes and for dealing with qualitative attributes and fuzzy expressions in addition to considering interdependency of the attributes.
- Lack of a reliable monitoring technique for ensuring the sustainability of provisioning the user QoS requirements throughout the duration of the service.

It seems that cloud service selection will be studied indefinitely due to the complexity of the problem that typically involves a large number of services and a large number of qualitative, quantitative, and fuzzy service attributes. For example, a recent research study [20] proposed a cloud service selection technique based on fuzzy ontology and MCDM.

Other than the SMI framework effort, few research studies in the literature addressed the problem of the normalization of the cloud service attributes (uniform specification of the cloud services). For example, Youseff et al. [21] developed a unified Cloud ontology for better understanding of the cloud technology. Binz et al. [22] developed the Topology and Orchestration Specification for Cloud Applications (TOSCA) to describe composite cloud applications and their management in a modular and portable fashion.

Some other researchers have been concerned with the expert evaluation of cloud services. For example, Hwang et al. [23] developed cloud performance models for evaluating IaaS, PaaS, SaaS and hybrid clouds and used them to evaluate Amazon IaaS EC2 using various benchmarks such as Cloud Suite [24], HiBench [25], TPC-W [26] and YCSB [27]. They concluded that we need to develop application-specific benchmarks especially for Big Data analytics and machine

learning intelligence. Antoniou [28] developed the SkyMark framework for the performance evaluation of IaaS clouds using complex workloads that stress the compute, memory and disk components. He made a similar conclusion regarding the need for applying more diverse and realistic workloads.

Provisioning and monitoring of QoS has been studied extensively in networks [29, 30] and extends to cloud computing to ensure sustainable provisioning of user QoS requirements. In a recent research study, Rodrigues et al. [31] addressed the problem of monitoring cloud computing environments and compared fourteen different monitoring solutions with respect to their goals and capabilities. The authors pinpointed several open points of research such as translating high-level cloud services into low-level metrics against the infrastructure layer.

3. THE PROPOSED CLOUD SERVICE BROKER (CSB) ARCHITECTURE

Figure 1 shows a block diagram of the proposed CSB architecture. As shown in the figure, it is composed of six modules: the Cloud Service Discovery (CSD) module, the Service Normalization (SN) module, the Matching & Ranking (MR) module, the Decision Support (DS) module, the Service Evaluation (SE) module and the Service Monitoring (SM) module. In addition to those modules, CSB includes two data stores: the cloud service repository and the user log. They are discussed in more details in the following sub-sections.

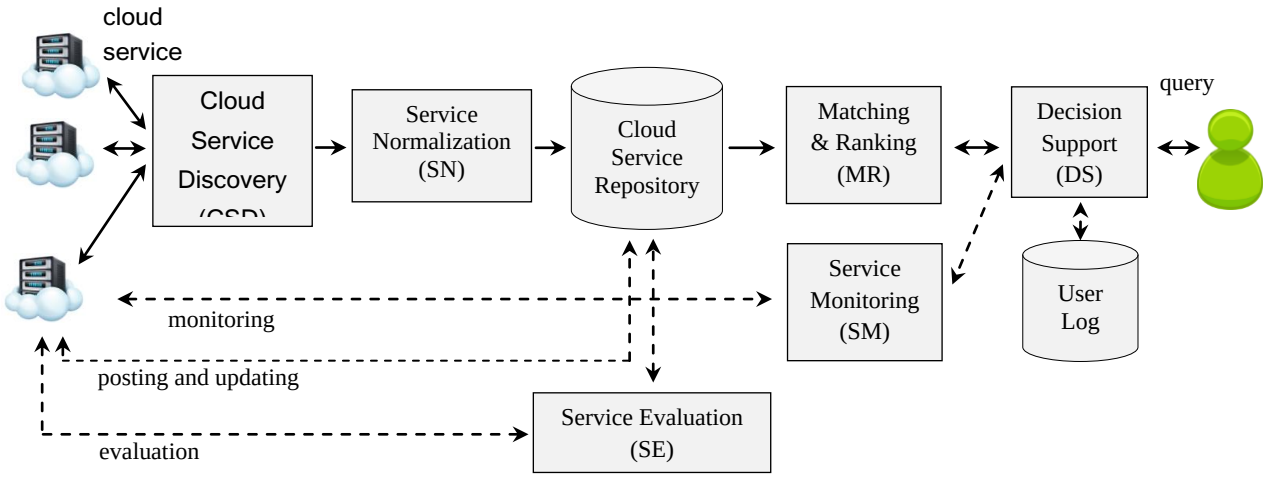

Fig 1: A block diagram depicting the architecture of the proposed cloud service broker (CSB)

3.1 The Cloud Service Discovery (CSD) Module

The CSD module is responsible for discovering candidate cloud services according to the user QoS requirements. Two approaches are generally used in the literature for this purpose: mobile agents [9] and search engines [10, 11] as discussed in Section 2. Unfortunately, mobile agents are associated with security threats since they are processes that can affect the servers they visit or, alternatively, can be modified on those servers [32]. In other words, unless there is mutual trust between the cloud servers and the visiting agents, cloud search engines might be a better option.

In addition to specifying an approach for discovering candidate cloud services, there should be a specification of the means of information extraction. As previously noted, Liu et al. [9] assumed the existence of service provider Web services for supplying the necessary information. It is also possible to process HTML documents and other Semantic Web documents including RDF, RDFs and OWL such as the Swoogle search engine developed by Ding et al. [33]. Natural language processing techniques can be also used for this purpose [34]. Other possibilities include extracting information from the data exchange XML files [35] or JSON files [36]. The extracted information is stored in the cloud service repository.

In a possible scenario, cloud service providers are able to post information about their services manually in the cloud service repository and update them as needed. Obviously, this requires some sort of access control to ensure that no cloud service provider tampers with the information of the others.

3.2 The Matching and Ranking (MR) Module

The MR module is responsible for matching the user QoS requirements against the candidate services in the cloud service repository to return one or more possible matches ranked according to their relevance. A good example is the ranking technique proposed by Garg et al. [7] though they considered only quantitative attributes. Other possible techniques include the MCDA techniques, optimization-based techniques and logic-based techniques discussed in Section 2.

Since the user QoS requirements are typically expressed in natural language, semantic matching and quantization of the QoS requirements might be required. As previously noted, Liu et al. [9] consulted an ontology for translating user QoS requirements to be matched against information about the candidate cloud services. Consumer agents select the best service using service ontology analysis. Similarly, Han and Sim [10] consulted an ontology to match the user QoS requirements against candidate cloud services information determining the similarities and the equivalences. A semantic similarity measure is used to determine the best matching service.

An ideal technique should be able to deal with all types of qualitative and quantitative attributes whether nominal, binary, ordinal or numeric. It should be also able to deal with all types of numeric attributes including interval-scaled and ratio-scaled attributes both discrete and continuous. This is in addition to considering dependent attributes. At the same time, it should be computationally efficient. This is one of the open and most active areas of research in the literature.

3.3 The Decision Support (DS) Module

The DS module is responsible for helping the user in selecting the most suitable cloud service matching his/her QoS requirements. When the user submits a query to the DS module describing the requirements, it consults the MR module to match those requirements against the candidate services in the cloud service repository to return one or more possible matches ranked according to their relevance.

Since none of the candidate services may satisfy the user QoS requirements to the letter, the user should be given the chance to give relative weights to his/her QoS requirements and prioritize them. Besides, the DS module should be able to negotiate the relevant services with the user specifying the pros and cons of each. The user log information specifying the user previous selections may be also used in this process.

The above discussion assumes that the users are professional enough to specify their QoS requirements. Since this is not always the case, the DS module should be able to deduce the required QoS requirements automatically by analyzing the user applications. This is another area of research that deserves considerable attention.

3.4 The Service Normalization (SN) Module

The SN module is responsible for normalizing the cloud service specifications. Most of the normalization techniques in the literature have been developed for Web services such as the Web Service Description Language (WSDL) [37] and the Semantic Markup for Web Service Description Language (WSDL-S) [37]. Normalization of the service specifications is crucial for smooth matching of the services against the user QoS requirements. In addition to the SMI framework effort, few researchers attempted to address this issue [21, 22] as discussed in Section 2 and this calls for further research studies.

3.5 The Service Monitoring (SM) Module

The SM module is responsible for monitoring the services provided to the users by the cloud service providers to ensure that the users get what they pay for, that they pay per service according to the Service Level Agreements (SLAs) and that the provisioning of the QoS requirements is sustained throughout the durations of the services.

This is an essential part of the cloud service broker since the users cannot do it on their own and cannot rely on the service providers for this purpose and so need a monitoring third party. Several monitoring solutions exist in the literature [31], but they are inadequate to cover all monitoring requirements calling for further research studies.

3.6 The Service Evaluation (SE) Module

The SE module is responsible for evaluating the services offered by the different cloud service providers. Unlike the SM module, this module does not monitor the services provided to specific users; it evaluates the services offered by the service providers with the goal of ensuring the accuracy of the information in the cloud service repository.

This is of ultimate importance to preserve the credibility of the cloud service broker that should not rely solely on information obtained from the service providers. Techniques for cloud service evaluation include the different cloud benchmarks and complex workloads discussed earlier [23-28].

4. AN IMPLEMENTATION OF CSB

In Section 3, we described the architecture of the proposed cloud service broker, CSB. We also discussed several possible ways to realize and implement each of its modules. In other words, various implementations can be developed based on this architecture. Each implementation may include some or all of those modules. Nevertheless, regardless of the implementation, the functionalities of the implemented modules should be integrated to help the user make more informed decisions regarding the selection of the relevant cloud service according to his/her QoS requirements.

In this section, we present the details of an example prototype implementation of CSB including the DS module, the MR module and the SE module in addition to the cloud service repository as shown in Figure 2. To integrate the functionalities of those modules, we propose a novel evaluation-aware matching and ranking technique. This matching and ranking technique takes into consideration not only the results of the evaluation of the services offered by the cloud service providers, but also the history of the service providers in satisfying or violating their SLAs.

In this example implementation, we assume that the cloud service providers publish information about their offered services in the cloud service repository. The user query is handed in to the DS module that forwards it to the MR module for matching and ranking. The MR module matches the user QoS requirements against the information in the cloud service repository and returns one or more matching services to the DS module ranked in order of priority. The DS modules helps the user make an appropriate selection. The SE module conducts regular evaluation of the services offered by the cloud service providers in the repository and updates their information accordingly. With each violation of the published information, the service provider is penalized resulting in a decrease in its rank. The rate of decrease increases with each violation.

4.1 The Cloud Service Repository

In the cloud service repository, each cloud service provider has a unique ID represented by the following equation:

$$service\ ID = \{type, function, < P_{1a}P_{1e} >, < P_{2a}P_{2e} >, \dots, n\} \quad (1)$$

In this equation:

- The *type* refers to the service type such as IaaS, PaaS, SaaS, or Big Data as a Service (BDaaS).
- The *function* refers to any further specification of the type such as the specification of the software offered in SaaS.
- P_{ia} refers to the advertised value of the service attribute (or performance metric) number *i* of the service provider.
- P_{ie} refers to the evaluated value of the service attribute (or performance metric) number *i* of the service provider.
- The value *n* is a measure of the frequency with which the evaluated values of the service attributes of the service provider do not match the advertised ones; the

computation of *n* will be explained in the next sub-section.

It is worth noting that a cloud service provider can have more than one ID in case more than one type of service and/or more than one function are offered.

4.2 The SE Module Implementation

The SE module uses several benchmarks to regularly evaluate the offered services of the different cloud service providers. In other words, it evaluates the service attributes of each service provider on regular basis indicating whether or not the evaluated values match the advertised ones in the cloud service repository. The result of the evaluation of each service provider is represented by the following equation:

$$E(service\ ID) = \{< P_{1e} >, < P_{2e} >, \dots\} \quad (2)$$

In this equation:

- The *service ID* is the unique ID of the cloud service provider under evaluation.
- P_{ie} refers to the evaluated value of the service attribute (or performance metric) number *i* of the service provider.

In other words, the evaluation results in an evaluated value for each of the service attributes of the service provider.

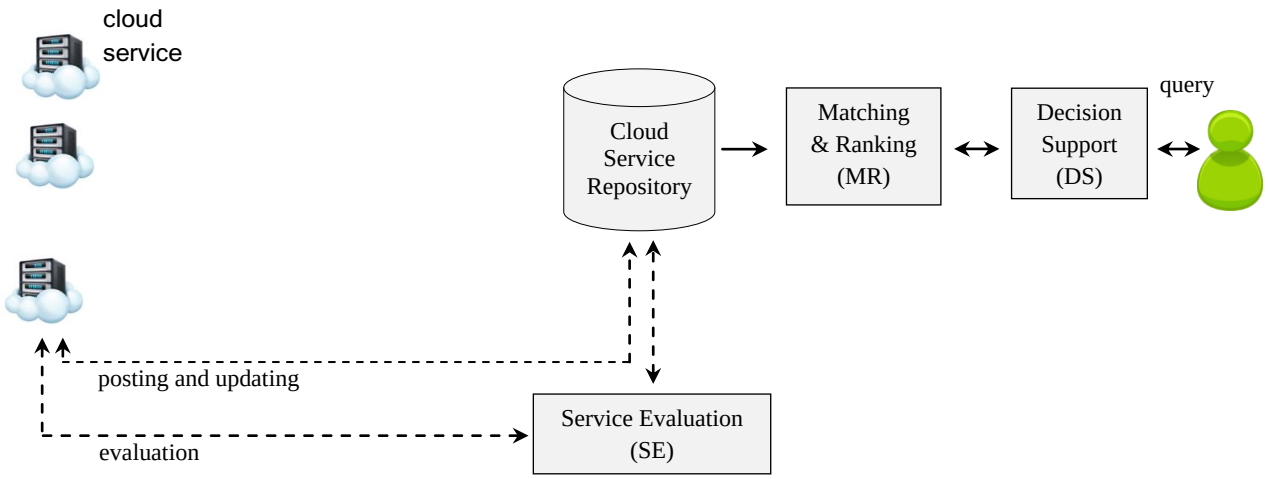

Fig 2: A prototype implementation of CSB

In case the evaluated value of any of the service attributes of the service provider does not match the advertised one, the evaluated value in the cloud service repository is updated accordingly. Additionally the value of *n* is incremented. On the other hand, if none of the evaluated values of the service attributes of a given service provider differs from the advertised ones, *n* is decremented. This continues with each evaluation until the value of *n* is reset to 0 (since a negative value of n is meaningless and hence not allowed).

4.3 The MR Module Implementation

The MR modules receives the QoS requirements of the user from the DS module. It starts by selecting the candidate services in the cloud service repository according to the service *type* and *function* specified by the user. The second step involves matching the service attributes of those services against the QoS requirements of the user and ranking the relevant ones in order of priority. In this implementation, we extend the relative rankings technique proposed by Garg et al. [7]. They start by computing a relative ranking for each service attribute of each candidate service such that the summation of the relative rankings of a given service attribute over all services is equal to one. For example, the relative rankings of the security attributes of three services may be 0.25, 0.5 and 0.25 respectively. The details of computing the relative rankings of the different attributes are beyond the scope of this paper. To rank the services, they compute for each the aggregation of the relative rankings of all of its attributes and order them accordingly. Nevertheless, the user is allowed to specify relative weights for the attributes such that the sum of the relative weights across all the attributes is equal to one. In such a case, the value of each attribute is multiplied by its relative weight before the aggregation.

We extend this technique by considering the evaluated values of the different service attributes rather than their advertised ones. Additionally, we multiply the final rank of each service by the following penalty factor:

$$penalty\ factor = e^{-n^2 * c} \quad (3)$$

In this equation:

- *c* is a constant value representing a penalty increment.
- The value *n* is a measure of the frequency with which the service provider is penalized as previously explained and is obtained from the cloud service repository.

Our first thought was to use a simple exponential function (simple penalty factor) as follows:

$$simple\ penalty\ factor = e^{-n * m * c} \quad (4)$$

In this equation:

- *m* is a multiple of *c*.

Using such a simple penalty factor, the rank of the penalized service decreases exponentially with each penalty. Figure 3(a) shows the curves of three example simple penalty factors with a value of *m* equal to 1, 2, and 3 respectively and a value of *c* equal to 0.025. It is clear that in case of a simple penalty factor, the service penalty increases with the increase of *n*, but it increases only exponentially.

We decided to select the penalty factor (equation 3) such that *m* itself increases with each penalty. The dashed curve shown in Figure 3(a) is that of the selected penalty factor. As shown in the figure, it intersects the three simple penalty factor curves successively with each penalty. Figure 3(b) clarifies this more by depicting the curve of the penalty factor against a set of simple penalty factors with values of *m* ranging from 1 to 5.

In effect, we increase the rate of the penalty for the service providers who consistently violate their SLAs. In other words, the implemented MR module considers not only the services advertised by the cloud service providers, but also the log of the evaluation trail.

4.4 The DS Module Implementation

The implemented DS module accepts a query from the user and sends the QoS requirements to the MR module to match the candidate services and return a set of suitable services ranked according to their priority. The DS module does not select the highest ranking service directly, but negotiates the selection with the user for many reasons such as:

- Assume a service has a low ranking due to a penalty and (a) the penalty is because the evaluated value of one of the attributes does not match the advertised one, (b) the evaluated value of this attribute is higher than that of the other services. In such a situation, the user has the right to accept the risk and select this risky service.
- None of the services match the QoS requirements of the user to the letter.

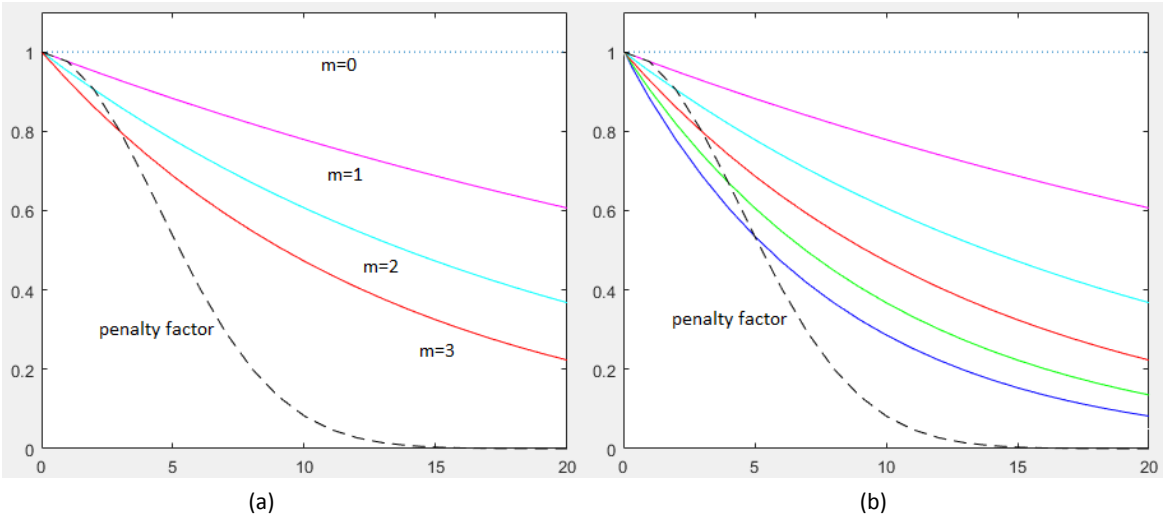

Fig 3: Several possible penalty factors

5. CASE STUDY

In this section, we provide a case study using the example prototype implementation of CSB presented in Section 4 as a proof of concept to justify the advantage of the integration of the different functionalities in such a comprehensive broker. In this case study, we consider three IaaS service providers: Amazon EC2, Windows Azure and Rackspace. A number of service attributes (performance metrics) have been collected by Garg et al. [7] from several evaluation studies. Nevertheless, since many assumptions are made in this case study, we will use the metrics of those service providers but assume they belong to other three service providers A, W and R respectively. We make the following assumptions:

- The collected values are equal to both the advertised values and the evaluated values of the service attributes of the three providers.
- The three services are all of the same *type* and have the same *function* and so we compare them only according to the evaluated values of the service attributes and *n*.

Using the relative ranking technique explained in Section 4, we obtain the following relative rankings matrix based on the evaluated values:

$$RR = \begin{pmatrix} 0.2500 & 0.3360 & 0.3812 & 0.4073 & 0.2846 & 0.2500 \\ 0.5000 & 0.3125 & 0.2671 & 0.3338 & 0.1181 & 0.5000 \\ 0.2500 & 0.3516 & 0.3517 & 0.2589 & 0.5973 & 0.2500 \end{pmatrix}$$

In the above matrix, the elements in the rows are the relative rankings of the attributes of the service providers A, W and R respectively. The elements in the columns, on the other hand, are the relative rankings of the attributes *security*, *agility*, *assurance*, *cost*, *performance*, and *accountability* respectively. By aggregating the relative rankings of all the attributes across each service provider, we obtain the rankings 1.9091, 2.0315, 2.0595 for the three providers respectively. Accordingly, R obtains the highest ranking while A has the lowest.

Schad et al. [38] showed considerable variabilities in the CPU performance of Amazon EC2 over the weekdays in addition to variabilities in the network performance. Accordingly, it is reasonable to assume that the evaluated values of the attributes of any service provider may not match the advertised ones. Accordingly, when we consider the evaluated values, we get different relative rankings for the attributes, a different relative ranking matrix and different rankings for the service providers.

According to our proposed technique, we not only compute the relative rankings using the evaluated values instead of the advertised ones, but also multiply the ranking of each service provider by a penalty factor in case *n* is not equal to 0. So, assuming, for example, that the value of *n* for the service provider R is equal to 1 and that the value of the penalty increment *c* is 0.025, the ranking of this provider is multiplied

by the penalty factor $e^{-1^2*0.025}$, which is equal to 0.975 and so the ranking of S becomes 2.008. In other words, W gets the highest ranking instead of R. If R is penalized one more time, its ranking is multiplied by the penalty factor $e^{-2^2*0.025}$, which is equal to 0.904 and so the ranking of S becomes 1.863 and obtains the lowest ranking. It is clear that integrating the results of the evaluation with the matching and ranking process helps the user make more informed decisions and helps him/her avoid service providers with low credibility.

6. CONCLUSION

This paper presents CSB, a proposed comprehensive user-centric cloud service broker. CSB is designed to help users select suitable cloud services matching his/her QoS requirements. The paper discusses related research in the literature highlighting the different research directions dealing with different aspects of the problem. Accordingly, it describes the architecture of CSB explaining how it integrates and orchestrates all those functionalities.

The different possibilities for realizing and implementing the different modules of CSB are discussed with prominent examples from the literature and open points of research are highlighted. These can be summarized as follows:

- Efficient and accurate techniques for the automatic extraction of information about the different offered cloud services from the Web pages of the cloud service providers.
- Techniques for the semantic matching and quantization of the user fuzzy QoS requirements and those expressed in natural language.
- Computationally inexpensive techniques for matching the user QoS requirements against the candidate services to select the relevant ones and rank them in order of priority.
- Matching and ranking techniques that are able to deal with all types of qualitative and quantitative service attributes whether nominal, binary, ordinal or numeric and with all types of numeric attributes including interval-scaled and ratio-scaled attributes both discrete and continuous.
- Negotiation techniques with the user in case a compromise is needed when none of the candidate services satisfy all the QoS requirements.
- Techniques for the automated specification of the QoS requirements relevant to the user application.
- Normalization techniques for the cloud service specifications.
- Monitoring techniques that ensure the user receives the required QoS requirements specified in the SLA throughout the duration of the service.
- Evaluation techniques for evaluating the services offered by the different cloud service providers so as not to rely on the information they provide.

As a proof of concept, to justify the advantage of the integration of several functionalities in such a comprehensive broker, we present an example implementation of a prototype based on CSB in addition to a case study using this prototype. In this example implementation, we propose a modeling technique for the cloud services and for their evaluation. We also propose a novel evaluation-aware matching and ranking technique for integrating evaluation results of the cloud services with their matching and ranking against the user QoS requirements for more informed selection decisions by taking into consideration the credibility of the cloud service providers.

As a future work, we intend to continue working on enhancing the implementations of the different modules of CSB. In other words, different design options will be explored, tested and compared in various scenarios to implement the most advantageous ones. Additionally, we intend to develop a complete implementation of the broker and employ it in real cloud environments. More case studies will be considered and the future enhancements and results will be reported in subsequent papers. The paper stimulates further research to enhance the different modules of CSB and promote its adoption for the sole benefit of the cloud services users.

7. REFERENCES

[1] Foster, I., Zhao, Y., Raicu, I., and Lu, S. 2008. Cloud computing and grid computing 360-degree compared. In Proceedings of IEEE Grid Computing Environments Workshop, Austin, TX, USA, (2008).

[2] Mell, P. and Grance, T. 2011. The NIST definition of cloud computing. Special Publication 800-145, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, (2011).

[3] Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N. and Hu, B. 2015. Everything as a Service (XaaS) on the Cloud: Origins, current and future trends. In Proceedings of the 8th IEEE International Conference on Cloud Computing, New York, USA, (2015), 621-628.

[4] Elazhary, H. 2014. Cloud computing for Big Data. MAGNT Research Report (2014).

[5] Elazhary, H. 2015. A cloud-based framework for context-aware intelligent mobile user interfaces in healthcare applications. Journal of Medical Imaging and Health Informatics 5(8) (2015) 1680-1687.

[6] Youssef, A. 2012. Exploring cloud computing services and applications. Journal of Emerging Trends in Computing and Information Sciences 3(6) (2012).

[7] Garg, S., Versteeg, S., and Buyya, R. 2013. A framework for ranking of cloud computing services. Future Generation Computer Systems 29 (2013) 1012-1023.

[8] Brock, M. and Goscinski, A. 2011. Enhancing Cloud computing environments using a cluster as a service. In: Cloud Computing: Principles and Paradigms, John Wiley & Sons, (2011), 193-219.

[9] Liu, D., Xing, W., Che, X., and Bao, P. 2015. A centralized service discovery approach for agent-based cloud computing system. The Open Cybernetics & Systemics Journal 9 (2015) 526-535.

[10] Han, T. and Sim, K. 2010. An ontology-enhanced cloud service discovery system. In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, (2010).

[11] Gong, S. and Sim, K. 2014. CB-Cloudle: A centroid-based cloud service search engine. In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, (2014).

[12] Rehman, Z., Hussain, O., and Hussain, F. 2012. IaaS cloud selection using MCDM methods. In Proceedings of the 19th IEEE International Conference on e-Business Engineering, Hangzhou, China, (2012), 246-251.

[13] Saaty, T. 2008. Decision making with the analytic hierarchy process. International Journal of Services Sciences 1(1) (2008) 83-98.

[14] Behzadian, M., Otaghsara, S., Yazdani, M., and Ignatius, J. 2012. A state-of the-art survey of TOPSIS applications. Expert Systems with Applications 39 (2012) 13051-13069.

[15] Roy, B. 1991. The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31 (1991) 49-73.

[16] Brans, J. and Vincke, P. 1985. A preference ranking organisation method: (The PROMETHEE method for multiple criteria decision-making). Management Science 31(6) (1985) 647-656.

[17] Whaiduzzaman, M., Gani, A., Anuar, N., Shiraz, M., Haque, M., and Haque, I. 2014. Cloud service selection using multicriteria decision analysis. The Scientific World Journal 2014(459375) (2014).

[18] Sun, L., Dong, H., Hussain, F., Hussain, O., and Chang, E. 2014. Cloud service selection: State-of-the-art and future research directions. Journal of Network and Computer Applications 45 (2014) 134-150.

[19] Quinton, C., Romero, D., and Duchien, L. 2014. Automated selection and configuration of cloud environments using software product lines principles. In Proceedings of the 7th IEEE International Conference on Cloud Computing, Alaska, USA, (2014).

[20] Sun, L., Ma, J., Zhang, Y., Dong. H., and Hussain, F. 2016. Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection. Future Generation Computer Systems 57 (2016) 42–55.

[21] Youseff, L., Butrico, M., and Da Silva, D. 2008. Toward a unified ontology of cloud computing. In Proceedings of the Grid Computing Environments Workshop, Austin, TX, (2008).

[22] Binz, T., Breiter, G., Leymann, F., and Spatzier, T. 2012. Portable cloud services using TOSCA. IEEE Internet Computing 16(3) (2012) 80-85.

[23] Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W., and Wu, Y. 2015. Cloud performance modeling with benchmark evaluation of elastic scaling strategies. IEEE Transactions on Parallel and Distributed Systems 27(1) (2015) 130-143.

[24] Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, A., Ailamaki, A., and Falsafi, B. 2012. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. In Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, London, UK, (2012).

[25] Huang, S., Huang, J., Dai, J., Xie, T., and Huang, B. 2010. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In Proceedings of the IEEE 26th International Conference on Data Engineering, (2010), 41-51.

[26] Smith, W. 2005. TPC-W: Benchmarking an ecommerce solution. Intel, (2005).

[27] Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. 2010. Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM Symposium on Cloud computing, IN, USA, (2010).

[28] Antoniou, A. 2012. Performance evaluation of cloud infrastructure using complex workloads. Master Thesis, Delft University of Technology (2012).

[29] Elazhary, H. and Gokhale, S. 2004(a). Integrating path computation and precomputation for quality-of-service provisioning. In Proceedings of the 9th IEEE International Symposium on Computers and Communications, Alexandria, Egypt, (2004).

[30] Elazhary, H. and Gokhale, S. 2004(b). An integrated approach for QoS provisioning and monitoring. In Proceedings of IASTED International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria, (2004).

[31] Rodrigues, G., Calheiros, R., Guimaraes, V., Santos, G., Carvalho, M., Granville, L., Tarouco, L., and Buyya, R. 2016. Monitoring of cloud computing environments: Concepts, solutions, trends, and future directions. In Proceedings of the 31st ACM Symposium on Applied Computing, Pisa, Italy, (2016).

[32] Alfalayleh, M. and Brankovic, L. 2005. An overview of security issues and techniques in mobile agents. IFIP Advances in Information and Communication Technology 175 (2005) 59-78.

[33] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R., Peng, Y., Reddivari, P., Doshi, V., and. Sachs, J. 2004. Swoogle: A search and metadata engine for the Semantic Web. In Proceedings of the Conference on Information and Knowledge Management, (2004), 652-659.

[34] Selvadurai, J. 2013. A natural language processing based Web mining system for social media analysis. International Journal of Scientific and Research Publications 3(1) (2013).

[35] Myllymaki, J. 2001. Effective Web data extraction with standard XML technologies. In Proceedings of the 10th International World Wide Web Conference, Hong Kong, (2001), 689-696.

[36] Knoblock, C. and Szekely, P. 2015. A scalable architecture for extracting, aligning, linking, and visualizing multi-Int data. In Proceedings of SPIE Next Generation Analyst 9499 (2015).

[37] Herrmann, M., Aslam, M., and Dalferth, O. 2007. Applying semantics (WSDL, WSDL-S, OWL) in Service Oriented Architectures (SOA). In Proceedings of the 10th International Protégé Conference, Budapest, Hungary, (2007).

[38] Schad, J., Dittrich, J., and Quiane-Ruiz, J. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. In Proceedings of VLDB Endowment 3(1) (2010).