QoS and Cost Aware Service Brokering Using Pattern Based Service Selection in Cloud Computing

Mandeep Devgan, Kanwalvir Singh Dhindsa

***Abstract*—In this paper an effective Services selection mechanism has been introduced for creating a practically useful Service Broker. Selection of Services is based on characteristics such as performance, reliability and cost, ranking and integrity are also considered. In Cloud computing a service broker is responsible for routing the user requests to the most appropriate Services. Traditionally, user of a service issues service request with some specific characteristics to a service broker and the broker searches all available Services with specified service and with a certain level of the user satisfaction. But, how can we select a set of available services from a query of service user with the some restriction? To solve this issue, we propose a service selector for service broker that can denote the restriction of similar services into a service test data, which is a set of similar cloud services, and select a set of services that provide a certain level of service consumer's satisfaction. We first identify the performance, reliability and cost of services which could be important for a cloud service consumer while requesting and then represent them in a knowledge base. And then we implement a Usage Pattern based selection mechanism to handle a service request with Limitation and the selection method is experimented on a simulated service test data. First Part also involves the testing and comparison of the Usage Pattern mechanism with traditional selection mechanism. In Part II, some of functionally similar Cloud services with different non-functional characteristics are modeled and each web service is differentiated with their non-functional properties. The usage pattern based mechanism is then incorporated with a user interface for consumer, so that user can request the service-broker for a set of best cloud services in terms of required levels of non-functional characteristics. The Usage Pattern based service selection mechanism in the service-broker will give a set of best services according to the required level of consumer satisfaction. The consumer can then select any service from the set and invoke it through its Uniform Resource Locator (URL) address. This last Part leads to the concept of an automated service broker satisfying the needs of the consumer with usage pattern. This usage pattern-based Broker would be able to satisfy the consumer requests better than a traditional broker by finding more cloud services and at the same time giving consumer the flexibility to describe its requirements in a flexible and more realistic manner.**

***Keywords*—Cloud Computing, Service Broker, Service Selection, Usage Pattern.**

I. INTRODUCTION

Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. Capital and operational costs can be cut using cloud computing [11]. Cloud computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. A parallel to this concept can be drawn with the electricity grid, wherein end-users consume power without needing to understand the component devices or infrastructure required to provide the service. There are three main stakeholders in the Cloud Computing, which are the service producers, service brokers and service consumers [12]. Service producers implement softwares, computing platforms and computing infrastructure related components and publishes some of them as cloud services onto service directories. A service consumer issues service requests with precise limitation to a service broker and the broker searches a set of available services for the service consumer. While selecting a service the service broker takes into consideration the minimum required level matching between requirements and service. A service broker is an important part of the cloud services model of modern computing, which handles queries about the available service and provides results to the consumer [21]. When several similar cloud services are available, their characteristics like performance, reliability and cost become significant [26]. Then the cloud services can be differentiated with this information and can be used by a good service selection mechanism for discovering a set of best available services during the discovery time. This paper proposes a cloud service selection mechanism for an Intelligent Service Broker, parameters such as performance; reliability and cost etc. are used for searching a service. Furthermore, a Usage Pattern Matching method (UPM) is used in mapping the queries to services. Usage pattern of a particular service in a particular region can help in predicting the service scheduling.

Manuscript received on May, 2013.

Mandeep Devgan, Department of Information Technology, Chandigarh Engineering College, Landran, Mohali, India.

Kanwalvir Singh Dhindsa, Department of CSE/IT, Baba Banda Singh Bahadur Engineering College, Fatehgarh Sahib.

This approach will attempt to achieve user satisfaction by offering a cloud service selection mechanism (CSSM), which promises higher levels of user satisfaction and at the same time being light and simple. The service selection is done, based on performance, reliability and cost. Dr. Lotfi A. Zadeh first introduced usage pattern in 1965 in his paper "Usage Pattern Sets" [27] in which he detailed the mathematics of Usage Pattern set theory. In 1973 he proposed his theory of usage pattern [11]. In another paper [4], Dr. Zadeh claims that the real world is pervasively imprecise and uncertain. This means that most of the concepts can and should be represented as a matter of degrees in order to make them realistic and more satisfactory as opposed to the best fit approach. The two-valued logic is not always sufficient to answer every question, whether it is about how good looking a person is or how warm the water should be in a washing machine?

Our approach is the same in case of cloud services. Because it is not always possible for a user to describe the performance and reliability related information clearly. While requesting, the service consumer can take advantage of requesting indefinite non-functional limitation. For example the required cost of a service can be expressed as 'around 10 $' by the consumer, instead of saying 'under 10 $,' which will definitely not give a chance to a service to be chosen with cost of 10 $ but a lot of other perfect matches for the requested performance and reliability. Therefore, a Usage Pattern Selector for the cloud services will make it possible to increase the level of satisfaction of the consumer by selecting a set of best services, which will suit the consumer's Quality of Service requirements. Cloud Services are categorized into Software as Service (SaaS), Platform as Service (PaaS) and Infrastructure as Service (IaaS) [22]. Semantics are applied to the cloud services, which help improve software reuse, composition and discovery and allow incorporation of legacy applications as part of business process integration [17]. We use metadata to add semantics to the cloud services, which provides simple and lightweight semantics [25]. The rest of the paper is structured as follows. Section 2 briefly discusses the related work. Section 3 describes the issues, our approaches to solve them and their specifications. Section 4 and 5 present the details of Parts I and II respectively along with the specifications of the experiments performed and their results. In Section 6, we represent the overall conclusions and reveal the opportunities for future work.

II. Related Work

Many researchers have considered web service consumer satisfaction in the past and they have come up with different solutions. Harney and Doshi suggest a mechanism of using expiration times for web services after which their QoS parameters are re-evaluated [23]. However, one issue in this approach is that it is computationally intensive and complicated. Several algorithms are involved for adaptive web process, policy implementation and for querying the producer for updated statistics. A similar effort for the improvement of consumer Satisfaction is done by Yolum and Sensoy [24], where they record the consumer experience with service producers and based on that, let the system decide which provider will be the best. However, this approach is more service provider oriented and does not discuss how a set of services can be selected amongst several services of the same functionality but different QoS values, in order to satisfy consumer requirements. There are also several previous efforts that handle the application of usage pattern with the cloud services in one way or the other but no one has ever proposed the Limitation selection and insertion mechanisms with minimal semantics using usage pattern for the improvement of end-to- end satisfaction level.

The closest to the work done in this project is Di Penta and Troiano's work in [5], in which they try to resolve the problem of automated discovery, which is faced while using genetic algorithms. As a solution, they relax the limitation by defining them as imprecise numbers. However, the focus of the paper is on matching the Limitation at the consumer side and at the broker side to obtain a fitness function. First of all, there is no implementation or experimentation that has been done or reported in this paper. Secondly the imprecise specification is only limited to the consumer and broker. However in our project we also allow the service producer to be able to describe its limitation in a Usage Pattern manner. Lin and et al [6] also apply usage pattern for the constraint Representation of the web services. It also applies QoS trade-off between the limitations but clearly the focus of this paper is towards the composition of the web services and the speed of the cloud services. The user satisfaction is not discussed in detail and again the insertion mechanism is not even touched. Tong and Zhang also apply usage pattern for imprecise QoS service limitation and implement a ranking algorithm in order to rank service according to the values of their non- functional characteristics in [22]. But unlike our approach they do not take the inaccurate and Usage Pattern limitation from the user but preset and precise values are taken. Our approach is to give consumer the liberty to use Usage Pattern words like "around" while describing the limitation.

Perryea and Chung is one of the big inspirations for this project, as they introduce the community-based architecture for the web service composition and automatic discovery [2]. We like the idea of service community and implement similar community with test data of cloud services in our project. However the paper is mostly focused on the web service composition and not about the satisfaction level of the end-to- end communication. Also they do not use the SACSDL mechanism, which is a way to introduce the minimal semantics for such lightweight services. Instead they use the OWL-S, which is very complicated, and limits the reuse and integration of the service with OWL-S only, where as SACSDL offers annotations independent of what ontology language is used.

III. Issues and Solving Approach

In order to solve the problem of cloud service consumer satisfaction several efforts have been done in the past. However there are two issues need more attention:

1) How can a cloud service consumer be given the flexibility to roughly specify his quality of service, cost and other related requirements while requesting for a web service?

2) How can a simple and light mechanism which promises to offer such flexibility to the service consumer be implemented and compared with traditional mechanism and put to real time use in cloud service technology?

A. System at a Glance

These issues are tackled in this paper by following an incremental approach in two Parts. In the first Part, the first issue is resolved in this paper by using Usage Pattern concepts. First the cost and quality of service characteristics of the cloud services are identified and stored in a knowledge base. Then a service selection mechanism is implemented which utilizes the very basics of Usage Pattern theory. We call this mechanism 'Usage Pattern Selector for Broker '. Usage Pattern theory suggests a way of processing data so that to allow partial set membership as opposed to crisp membership in the case of best fit approach [22]. This mechanism handles a service request with limitation and experimented on a simulated service test data. The mechanism is then tested and compared with the 'Traditional Selector for Broker' in order to show that it works better. After making sure that pattern selector based broker is satisfying the consumer requests better than the Traditional broker, we move towards the further integration and testing of pattern based selector with some simulated cloud services.

The second issue is how to integrate the Usage Pattern mechanism with the real cloud services. Cloud services are described with the help of their non-functional characteristics which can be described with the help of semantics. In Part II, we use the latest recommendation of W3C in order to semantically annotate web services. These recommendations are used to reference the ontology that describes the non-functional and QoS characteristics of web services. In order to utilize this information we use the semantics fetcher. The overall structure of the system is shown in Fig. 1.

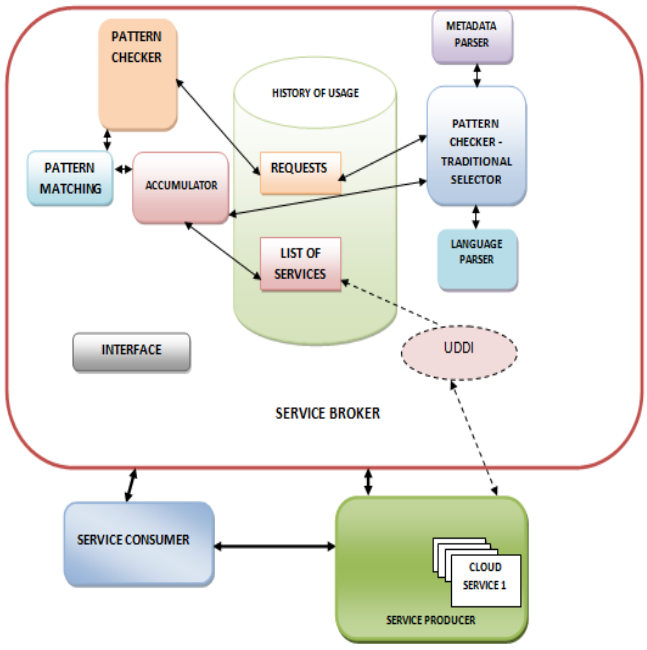

Fig. 1. System Overview

In Part II, a set of cloud services is simulated in simulation environment. So the consumer will request the service broker for a set of best cloud services in terms of her required levels of cost and rating etc. The Usage Pattern selection mechanism in the service-broker will yield a set of best services according to the required level of consumer satisfaction. The consumer can then select any service from the set and invoke it through its Uniform Resource Locator (URL) address. This last Part leads to the concept of an automated service broker satisfying the needs of the consumer with usage pattern.

B. The Usage Pattern Concept

The usage of a Cloud client can sometimes have a repetitive behavior. This can be caused by the similarities between tasks that the Cloud client is running or the repetitive nature of human behavior. Given the self-similar nature of web traffic, it follows that current usage patterns of online services have a probability of having already occurred in the past in a very similar form. Therefore we can infer what the system usage will be for a Cloud client by examining its past usage and extracting similar usages .A careful analysis of the service request logs of any service can provide better information about the usage patterns of that service by same or different customers. SB broker in cloud computing can use this concept to improve its service scheduling process. This concept is normally used in predicting the user behavior in many activities. Accuracy of Prediction improves with time and increase in number of patterns. Usage pattern concept can be easily modeled using set theory [26]. Just like the normal set operations, Usage Pattern sets also have some basic operations that can be performed. Most common operations are compliment, intersection and union.

In our research, since we are using usage patterns to predict the service scheduling, we will need to find match between history and current Usage patterns.

IV. PART I - SERVICE SELECTION AND LIMITATIONS

In Part I, we first identify the Performance, Reliability and Cost related characteristics of cloud services which could be important for a service consumer while querying and then represent them in a knowledge base. After that we implement a usage pattern based selection mechanism to handle a service request with constraint, the selection method is experimented on a simulated service test data. Part I also involves the testing and comparison of the Usage Pattern based selection mechanism with traditional selection mechanism.

A. Quality of Service and other Characteristics of a Cloud Service

First we identify and define the QoS and non-functional properties of the web services: QoS properties such as performance and reliability and non-functional properties such as cost, rating, and integrity are described respectively.

Performance

Performance is a QoS characteristic, which represents the execution time of a service. Most of the time the performance of a system is determined by the efficiency of that system, which is further defined as amount of work done in a period of time. Therefore, execution time of a service can

be a good

Reliability

The reliability of a service is also a QoS characteristic and can be determined by analyzing the number of times a particular service works well over a certain number of its invocations.

Cost

The cost of a service is a self-explanatory non-functional characteristic. Cost is the amount of money charged for a certain number of invocations of a service. For example a particular service can cost 10 cents for 100 invocations while another can cost 80 cents for 100 invocations.

Rating

The rating of a web service is a non-functional characteristic that is determined by the consumer and is optional. Since the broker tends to take off the work from the consumer's shoulders as much as possible, rating a particular service is left optional. However, if a consumer choose to participate and has a mechanism to provide its feedback, the rating can be stored at the broker's end.

Integrity

The integrity of the web service is also a non-functional characteristic and is determined by the broker. The rating is about a particular service whereas the integrity is about a particular service producer. Since the consumer does not have the producer's information, it can only rate the service. However based on that rating a broker can determine the integrity of a particular service producer.

Security

The security of the web service is also a non-functional characteristic and is also determined by the broker. The security of a cloud service is composed of security at various levels of cloud service architecture.

B. Traditional Selector for Broker

For the traditional selector the best fit approach is applied which determines whether a service satisfies the conditions or not. For example if time (T) and reliability (R) are the QoS parameters for a web service, it determines whether the execution time is less than or equal to the required level AND reliability is greater than or equal to a required level provided by the consumer

$$(\text{Time} \leq \text{RL1}) \text{ AND } (\text{Reliability} \geq \text{RL2}) \quad \text{(I)}$$

Equation I shows the relation where RL1 and RL2 are the rough values provided by the consumer and T and R are the actual QoS values for the service described in the test data. So the consumer requests to select a service that has an execution time at most c1 (ms) and a reliability of at least c2 (%). Based on equation (1) we can definitely have a single value function for Time and one for reliability.

C. Experiments and Results (Comparison and Testing)

For the comparison and testing purposes in the, which is shown in Fig. 2, is implemented in which the same input is given to both the traditional and Usage Pattern selection mechanisms. Both the mechanisms also share the same test data so that there are no unfair circumstances. A knowledge base is created which is a set of 600 test data of cloud services divided in 4 groups. Each group is having 150 cloud services, of 2, 3, 4 and 5 numbers of Limitations. Each test data is tested with 200 randomly and automatically generated input values, which are supposedly provided by the consumer. Finally, the results are generated by averaging 100 such iterations.

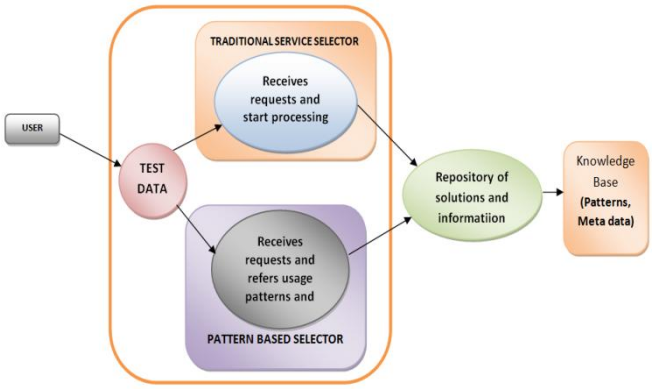

Fig. 2. Comparisons of Service Selection Mechanisms

The results of the experiment show a significant improvement in the consumer satisfaction as we move from hard to soft limitation. The results are divided into Response time, Query Processing time and Cost. The support perspective is based on average number of services selected by each mechanism. The Usage Pattern mechanism's performance is significantly better than the traditional mechanism and can be easily noticed in the graph. Table I, II and III shows the results and small difference between Usage Pattern and traditional mechanism that can be seen. However, it is observed that as we increase the number of the number of requests, the Usage Pattern mechanism performs even better than the traditional one. We call 'increasing the number of requests' as 'moving towards the real world', as in real web services the number of non-functional and QoS limitations can be much more than ones in this paper.

Requests/ User/Hour	Response Time(ms)	
	Traditional Service Selector	Usage Pattern Based Selector
100	52.69	50
200	58.78	50.14
300	58.46	50.08
400	58.64	50.16
500	58.56	51
600	59.25	51

Table I. Response Time Comparison

Requests/ User/Hour	Query Processing Time(ms)	
	Traditional Service Selector	Usage Pattern Based Selector
100	3.07	0.48
200	9.14	0.50
300	8.89	0.50
400	8.99	0.50
500	8.88	0.51
600	9.65	0.49

Table II. Query Processing Time Comparison

Requests/	**Cost($)**

User/Hour	Traditional Service Selector	Usage Pattern Based Selector
100	12.34	1.21
200	12.54	1.5
300	12.62	1.63
400	12.96	2.01
500	13.03	2.06
600	13.54	2.26

Table III. Cost Comparison

Above comparisons clearly reveal that Traditional Service Selector is no longer beneficial in cloud environment. When number of requests increases the Response Time, Query Processing Time and Cost increase in case of Traditional Service Selector.

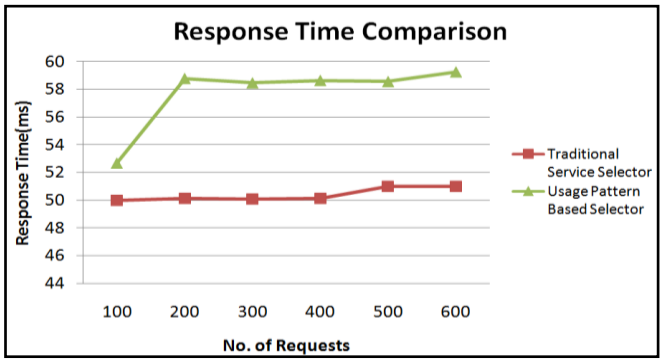

Fig. 3.Comparison of User Request Response Time

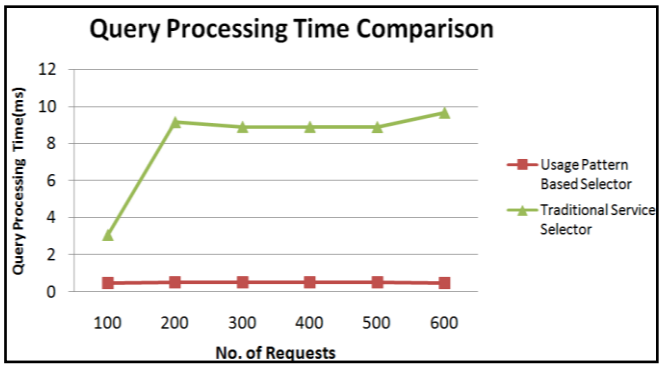

Fig. 4.Comparison of Query Processing Time

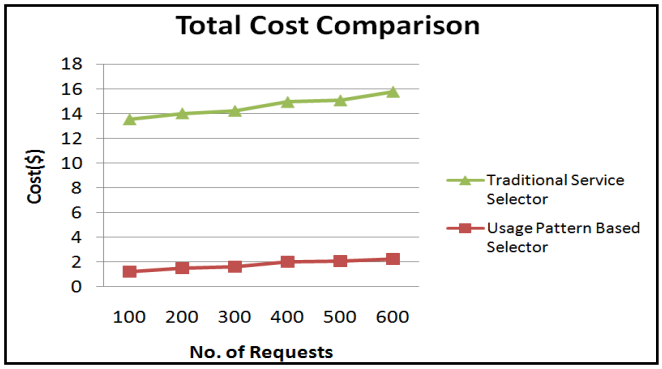

Fig. 5. Comparison of Total Cost

V. PART II-USAGE PATTERN SELECTION MECHANISM

Sometimes the client of a cloud service may exhibit a repetitive behavior. This can be caused by the nature of human activities or similarities between tasks that the Cloud client is performing. Given the self-similar nature of web traffic, it follows that current usage patterns of online services have a probability of having already occurred in the past in a very similar form. Therefore we can infer what the system usage will be for a Cloud client by examining its past usage and extracting similar usages.

The pattern strategy has two inputs: a set of past Cloud client usage traces and the present usage pattern that consists of the last usage measures of the Cloud client. Cloud clients working in the same application domain have a higher similarity in resource usages. Due to this similarity it follows that the most relevant historic resource usage data that can be used comes from Cloud clients working in the same application domain. Therefore it would make sense to isolate historical data based on application domains before usage [3].

A. Inspecting Sources of Data

We need to have a better way to choose the pattern that would give more relevant results and avoid pollution as much as possible. The pattern should be influenced by the time it takes to service a request on the server. By analyzing the data sources we have obtained the running time in mili seconds of each request with the results given in Table IV. The conclusions here are that, for all practical purposes, a pattern length that is a minimum or even a median of the time it takes for a request to be executed is unusable when dealing with servers that have a similar usage to the Cloud applications described above. In practice we have used the average of the request service time and have obtained good results.

B. Denotation of cloud services

In order to interpret the cloud services with their non-functional information and Quality of Service characteristics (Performance, Reliability, Cost), we used the Meta Data. In real life scenarios, automated cloud service brokering is often challenging because the service descriptions may involve complex constraints and require flexible semantic matching. Furthermore, cloud providers often use non-standard formats leading to semantic interoperability issues. In this paper, we formulate cloud service brokering under a service oriented framework, and propose usage pattern based cloud service discovery and selection system. The proposed system supports dynamic semantic matching of cloud services described with complex constraints.

C. Pattern Selector for Broker

The Pattern Selector mechanism for Broker is implemented to interact with Accumulator and User Requests. Pattern Selector utilizes the Repository of solutions and usage patterns to fetch the 'Reference Pattern' elements from the usage log files. Furthermore, the Metadata obtained from the usage history and current request is referred to get the values for The QoS and the cost related characteristics of the web service. The Java-based Document Object Model (JDOM) is used to interact with the XML files.

The accumulator with selector mechanism populates the test data after fetching the information about the non-functional and QoS characteristics of the cloud services. The test data once populated by the accumulator

can be used by the selector in order to select the best set of services.

D. Experiments and Results (Pattern Selector integrated with Cloud Service)

Since the comparison of the Usage Pattern mechanism with a traditional mechanism was already done in Part I, the challenge now is to test the Usage Pattern mechanism with real web service technology. After that integration with the help of the tools and technologies like CloudSim, CloudAnalyst system is tested with three use case scenarios.

One of the scenarios considered is the Response time comparison in both the cases. The results are shown in Table IV. As we can see average response time, average request processing time are higher in case of Service Broker with Traditional Selector.

Service Broker with A/B	Avg. Response Time(ms)	Avg. Request Processing Time(ms)	Avg. Cost($)
Traditional Selector	57.7	8.1	12.8
Pattern Selector	50.4	0.5	1.8

Table IV. Results for Traditional and Pattern based Selectors

VI. CONCLUSIONS AND FUTURE SCOPE

Both the QoS and non-functional characteristics of the cloud services can be represented in an imprecise manner during the discovery time. The selection mechanism can take the advantage of the usage pattern to increase the consumer satisfaction. The Usage Pattern and traditional mechanisms were compared and tested and it can be noticed that the Usage Pattern mechanism satisfies the consumer with higher levels of satisfaction from performance and cost related aspects. Also, as we increase the number of requests more improvements can be seen. Usage pattern also offers a natural way of ranking different services as it associates each element of the result ser to some degree to which it satisfies the consumer. The Usage Pattern mechanism is also less susceptible to the changes in the situations. The Usage Pattern based service selection mechanism can be integrated and used with the current web service technology.

In future more work can be done on the implementation of usage pattern in the service oriented computing paradigm. Just like the web service consumer was satisfied in this project by giving it a flexibility to express its non-functional requirements in a Usage Pattern manner, the service producer can also be given this flexibility by letting him provide his service characteristics in a Usage Pattern manner. In addition, service producers can also represent both the QoS and non-functional characteristics of the cloud services during the publication time. The emerging technologies allow the service producers to describe the semantics for each service in a simple manner. By inserting a reference of a semantic description for the non-functional and QoS properties in existing web service description, a service broker can fetch the information from the published cloud services.

Also the service brokering can be improved by having a periodic mechanism to automatically gather the non-functional information from the URL's and updates the knowledge base periodically with the new values. Through this way, a service consumer does not need to provide the broker with his service information but he can only change his service information on his URL's.

REFERENCES

[1] Hull, R., Su Jianwen. (2010), *Tools for design of composite cloud services*, Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, New York, pp. 958 – 961

[2] Perryea, C., Chung, S. (2011), *Community-Based Service Discovery*, Proceedings of the IEEE International Conference on cloud services (ICWS'06), Washington D C, pp. 903 – 906

[3] Zadeh, L. A. (2009), Usage Pattern, Computer IEEE, Vol. 21 (4), pp. 83-93

[4] Zadeh, L.A. (2012), Soft Computing and Usage pattern, Software IEEE, Vol. 11(6), pp. 48 – 56

[5] M. Di Penta, L. Troiano, Using Usage pattern to Relax Limitation in GA-Based Service Composition. Late breaking paper presented at the Genetic and Computation Conference (GECCO 2005), 2011

[6] M. Lin, J. Xie, H. Guo, H. Wang, Solving QoS–driven Web Service Dynamic Composition as Usage Pattern Constraint Satisfaction. Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE '05), p. 9–14, 2005.

[7] McIlraith , A , S., Son, C, T., Zeng, H. (2001), Semantic cloud services, Intelligent Systems IEEE, Vol. 16(2), pp. 46 - 53

[8] Fung, C.K., Hung, P.C.K., Wang, G., Linger, R.C., Walton, G.H. (2005), A study of service composition with QoS management, cloud services IEEE, Digital Object Identifier 10.1109/ICWS.2005.19

[9] Jaeger, M.C., Muhl, G., Golze, S. (2005), QoS-aware composition of cloud services: a look at selection algorithms, cloud services IEEE, Digital Object Identifier 10.1109/ICWS.2005.95

[10] Birman, K (2005), Can cloud services scale up?, Computer IEEE, Vol. 38 (10), pp. 107 - 110.

[11] Zadeh, L.A. (1965), Usage Pattern sets. Information and Control 8, 18 Pasteur 96, 15-23 17, pp. 338 - 353

[12] Booth, D., Haas, H., McCabe, F., Newcomer, E., Michael, I., Ferris, C., Orchard, D. (2004), cloud services Architecture, CLOUD FORUMS Working Group, retrieved from http://www.w3.org/TR/ws-arch/ on May 2, 2007.

[13] Mika, P., Oberle, D., Gangemi, A., Sabou, M. (2004), Foundations for service ontologies: aligning OWL-S to dolce, Proceedings of the 13th international conference on World Wide Web, pp. 563 - 572

[14] Christensen, E., Curbera, F., Meredith, G., Weerawarana, F (2001), W3C Working Group , retrieved from http://www.w3.org/TR/wsdl on May 2, 2007 .

[15] W3C Schools, SOAP tutorial, retrieved from http: // www. w3schools. com/ soap / on May 5, 2007 .

[16] Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K. (2004), WSDL-S: A Proposal to W3C WSDL 2.0 Committee, METEOR-S: Semantic Web Services and Processes, retrieved from http://lsdis.cs.uga.edu/projects/wsdl-s/WSDL-S.pdf on May 08, 2007 .

[17] Miller, J., Verma, K., Akkiraju, R., Sheth, A., Schmidt, M., Farrell, J., Nagarajan, M. (2005), Web Service Semantics - WSDL-S, W3C, retrieved from http://www.w3.org/Submission/WSDL-S/ on May 08, 2007 .

[18] Organization for the Advancement of Structured Information Standards (2004), Introduction to UDDI:Important Features and Functional Concepts, retrieved from http://uddi.org/pubs/uddi-tech-wp.pdf on May 8, 2007 .

[19] Badidi, E., Esmahi, L., Serhani, M.A. (2005), A queuing model for service selection of multi-classes Q0S-aware Web services, Web Services ECOWS IEEE, pp. 9 - 17

[20] Degwekar, S., Su, S.Y.W., Lam, H (2004). Constraint specification and processing in Web services publication and discovery. Web Services, 2004 proceedings. IEEE International Conference. 6- 9 July 2004, pp 210 - 217

[21] H. Kreger (2001), Web Services Conceptual Architecture, IBM Software Group, May 2001.
[22] World Wide Web consortium (W3C), Web Service Activity Statement, retrieved from http://www.w3.org/2002/ws/Activity on June 03, 2007 .
[23] Harney, J., Doshi, P. (2007), Speeding up adaptation of web service compositions using expiration times, International World Wide Web Conference archive, Proceedings of the 16th international conference on World Wide Web, Pages: 1023 - 1032
[24] Yolum, P., Sensoy, M. (2006), A context-aware approach for service selection using ontologies, International Conference on Autonomous Agents, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, Pages: 931 - 938 .
[25] World Wide Web Consortium (W3C), Semantic Annotations for WSDL Working Group, retrieved from http://www.w3.org/2002/ws/sawsdl/ on February 25, 2008.
[26] De Cock, M, Chung, S., & Hafeez, O. (2007). Selection of Web Services with Imprecise QoS Constraints. Proceedings of WI-2007 (2007 IEEE/WIC/ACM International Joint Conference on Web Intelligence), p.535-541.
[27] World Wide Web Consortium (W3C), Semantic Annotations for WSDL and XML Schema - Specification retrieved from http: // www. w3. Org / TR/ sawsdl/ on February 25, 2008.