A Monitoring based Multi-Agent Filtering Approach for Web Service Selection

Raja Bellakhal[1], Fatma Siala[1] and Khaled Ghédira[2]
[1]National School of Computer Science, University of Manouba, Manouba, Tunisia
[2]Higher Institute of Management, University of Tunis, Tunis, Tunisia

Keywords: Web Services Filtering, Multi-Agent System, Monitoring, Quality of Service (QoS), Negotiation, SOAP Messages.

Abstract: During Web services selection processes based on the negotiation approaches, systems initially search for services that comply with the users' functional requirements. Then, based on the retrieved set of functionally similar services, the negotiators start negotiation in order to come up with an agreement about the QoS parameter preferences. Here the problem occurs when the number of services retrieved during the first step is huge. In such case, the performance of the Web service selection process based on the QoS requirements can be degraded. Examining the specifications of all retrieved services is certainly a waste of time. Instead, it is more reasonable to remove all services that are unavailable and under the users' requirement expectations before the start of the negotiation. To deal with this issue, in this paper we propose a multi-agent based filtering approach that adopts a Web service monitoring method allowing the filtering and the selection of the best candidate Web services for the selection process. The results of the conducted experimentations demonstrate that adopting an agent-based filtering process decreases the CPU time of the overall Web service selection process.

1 INTRODUCTION

With the growth in the number of functionally similar Web services, efficient systems become primordial to assist the users during the selection of the suitable services that comply with their specific requirements including the QoS parameters. Since Web services are operating in dynamic and changing environments, their QoS parameters change quite frequently. Actually, these parameters are dynamic and out of the providers' control as for example, the response time would depend on the bandwidth, the transmission delay, the packet size, etc.

A series of Web service selection systems that support the search for Web services based on the users' preferences in terms of the QoS parameters has been presented. The dominating system among them considers that QoS parameters are static (Bentahar et al., 2008; Karray et al., 2013). Once the quality of services is defined in the description and published in the UDDI (Universal Description Discovery and Integration) registry, they remain unchangeable. In order to solve the problem of static parameters, approaches based on statistic (Zhou and Chen, 2009) network based techniques (Chen et al., 2010; Benaboud et al., 2016) and negotiation (Napoli et al., 2013; Linlin et al., 2013) are proposed. Network and statistic based approaches have the inconvenience of over resources consumption, and the occurrence of conflicts between the clients' and providers' preferences. The negotiation has the advantage of solving discrepancies among the clients' and the providers' conflicting preferences observed during the aforementioned approaches.

The negotiation is a two-step process. During the first step, the system searches for services that match the clients' functional requirements, while during the second step, the best service is selected from the set of the functionally similar services. A huge number of candidate Web services selected during the Web services discovery step can affect the selection system performance during the selection step. Imagine that during the second step, all the discovered candidate providers go through many rounds negotiation processes with the client. Certainly, this will take a long time. Moreover, implementing a negotiation system that simulates

Bellakhal, R., Siala, F. and Ghédira, K.
A Monitoring based Multi-Agent Filtering Approach for Web Service Selection.
DOI: 10.5220/0006900301450152
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 145-152
ISBN: 978-989-758-324-7

the human real negotiation interactions and that considers all discovered services is not an easy task. The idea is to filter out the services that are unavailable, and do not fit the users' preferences. In order to deal with the aforementioned problems, we extend the negotiation system by adopting a filtering approach based on a monitoring method. Before the start of the negotiation process, the system must search for the best candidate services.

The filtering process is based on monitoring steps and evaluation of QoS parameters. Monitoring of the Web services involves collecting information about the real Web services performances during runtime. After each monitoring step, a set of services are filtered out based on their QoS values, and only adequate services are retained.

The reminder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the negotiation framework and the filtering process. Section 4 presents and reports the experimental results. Finally, the paper is concluded in section 5.

2 RELATED WORK

There is a wide range of research around the Web service selection systems based on QoS parameters. Some of it is introduced to enhance the matching techniques, the number of returned results, and to optimize the selection system complexity. However, these approaches consider that the QoS parameters are static. Actually, Web services operate in a changing environment, and systems designed to discover them must consider the QoS parameters dynamic. To tackle the problem of static parameters, two series of approaches are proposed. The first series is founded on network techniques and statistic methods (Zheng et al., 2012), (Huang, 2013) and (Benaboud et al., 2016). The disadvantages of these approaches are mainly related to the over resource consumption, the occurrence of conflicts between the clients and the providers, and the lack of Web service selection models. The second series is based on negotiation models.

In (Napoli et al., 2013) a market based negotiation mechanism among providers and users that request a QoS aware service is presented. The advantage of such a system is to consider the dynamic aspect of the QoS parameters. The providers may change the QoS values according to their provision strategies. The authors describe the negotiation as a bilateral process between two agents. During the negotiation, agents exchange offers and counter-offers. In the aforementioned approach the negotiation could end without any agreement since the process can terminate if a deadline expires. Moreover, only providers are able to generate offers whereas the clients can only evaluate them. The contract net iterative protocol is adopted. The authors note that this protocol has a communication overhead, so its performance degrades drastically. In (El-Awadi et al., 2014) the authors present an SLA based negotiation approach. The SLA contract must contain dynamically the updated QoS values. In this work, a negotiation engine is responsible for achieving an agreement. Once the agreement is found, the monitoring system checks whether the parameters related to the retrieved services are equal, under, or over the threshold defined in the SLA. Such a method may not always guarantee an agreement between the negotiators, since the generated SLA can be withdrawn if the monitored QoS attribute does not match the values defined in the SLA contract. In (Bellakhal and Ghédira, 2016), an agent selection system based on a hybrid negotiation is presented. This system introduces new aspects that were neglected by the existing systems. The first aspect concerns the simulation of certain characteristics observed during the real negotiation interactions between negotiators. The second aspect relates to making the negotiation dynamic by enabling the negotiators to change their negotiation strategies during the negotiation. The final aspect considers the dependency between the concurrent negotiation processes. The presented system is based on two negotiation models, namely the argumentative and the game-based negotiation models. In the presented approach, a monitoring method that guarantees the correspondence between the values generated by the providers and the real Web services performance is adopted. In (Ouadah et al., 2018), a hybrid approach based on multi-criteria decision method to select the best service from functionally similar services is adopted. This approach is based on the reduction of the decision space of the best candidate services for the selection process. The adopted Web services selection algorithm is based on aggregation of two criteria namely, the user opinions about Web service performance and the QoS parameters values extracted from invocation history data.

The main issue with the aforementioned works is related to the large number of candidate services that are involved in the selection process. This may make the process of finding the services that comply with the specific users' requirements longer and harder. Moreover, in most of these works, the monitoring of

Web service performance is either conducted during or after the selection process. On the one hand, adopting a monitoring process during the selection process can overload the system and postpone the process termination. On the other hand, intercepting the Web services performance after the selection process termination can result in the process failure. In this case, the real selected services performance is different from the users' preferences.

3 THE AGENT-BASED FILTERING PROCESS

In this paper, we adopt an agent-based filtering method over the set of discovered services in order to come up with a final set of the best candidate Web services for the agent-based negotiation process. In the following, we start by describing briefly the adopted negotiation framework as a basic element of the proposed approach. Next, we focus on the main objective of this paper, namely the presentation of a novel filtering approach.

3.1 The Negotiation Framework

We adopt a multi-lateral negotiation framework composed of concurrent sub-bilateral negotiation processes. Several negotiation rounds are conducted between competitive client agent's instances and provider agents. The client and provider agents are in continuous competition in order to get the best deal. They exchange offers and counter-offers while adopting concession or trade off strategy. The evaluation of the offers is based on the utility function presented by Zheng et al., (Zheng et al., 2012).

The adopted negotiation framework is based on five agents namely, the provider agent, the client agent, the intermediate agent's instances, the client agent's instances and the coordinator agent.

- The provider agent

  The provider agents involved in the negotiation process represent the discovered Web services. In the core of each provider agent is implemented a negotiation model by specifying its negotiation strategy and protocol. The number of provider agents depends on the discovered Web services. An instance of the provider agent is run for each retrieved Web services.

- The original client agent

  The first role of the original client agent is to provide to the client agent's instances information about the user's preferences such as QoS values and their related weights. Its second role is to register offers resulting from agreements between the provider and the client agent's instance.

- The intermediate agent's instance

  Dependencies among the concurrent sub-bilateral negotiation processes are ensured by the intermediate agent's instances. They communicate to the client agent's instances information about the offers of the competitor providers. This information is exploited by the provider in order to propose more interesting offers for the client and get the deal.

- The client agent's instance

  The role of the client's instances is to represent concurrently the original client agent's behavior. They negotiate with the provider agents by adopting different negotiation models. Client agents are launched as much as the provider agents. In the core of each client instance agent is implemented a negotiation model by specifying its negotiation strategy and protocol.

- The coordinator agent

  The first role of the coordinator agent is to register each agreement concluded between the client and provider agents. Its second role is to inform the original client agent about the offers resulting from agreements between the negotiators.

3.2 The Filtering Process

Figure 1 depicts the filtering process and the agents involved in it, namely the monitor agent, the coordinator agent and the launcher agent. The monitor agent has the role of intercepting information about the real Web services performance. This information is collected and used by the coordinator agent in order to filter out unavailable and inappropriate services. The representation of the monitors as agents has the advantage of gaining time. Once the monitor agents are launched by the launcher agent, different monitoring processes start by intercepting simultaneously the Web services performance.

The filtering is a multi-step process based on a multi-round monitoring method. The filtering process and the monitoring processes are conducted respectively in the core of the coordinator agent and the monitor agents. The monitoring refers to the control in runtime of the Web service performance before the beginning of the negotiation process. This has the effect of avoiding the overload of the system by reducing the processing time. During each step, a multi-round monitoring is adopted over the set of services filtered during the previous step. Each Web service belonging to the set of selected services will be invoked according to λ rounds. The value of λ is defined by the developer.

We focus on the selection of the candidate Web services for the negotiation phase. We suppose that *SWS1* is the set of the initial Web services discovered in response to the user's request expressed in terms of keywords and its initial QoS requirements. The adoption of the keyword match method during the discovery process seems too simplistic here. Our main problem in this paper is not to propose a full discovery and selection system, but rather to study the impact of a filtering approach over the overall CPU time related to the Web service selection process.

The user's requirements are defined by the favorite QoS values and their related weights. The set *SWS1* will be the input of the first step in the filtering process consisting in leaching out all unavailable services. For this purpose, a monitoring technique based on the SOAP[1] message is adopted. Since QoS parameters are dynamic and change over time, a monitoring process is essential for measuring their values in real time. The QoS parameters are out under the control of the provider. They depend on the network bandwidth, the propagation delay, the transmission delay, etc. Actually, we are not responsible for hosting the Web services, so it is impossible to control their performances. In order to solve this problem, we adopt the SOAP message calls. When a service is invoked, a request is sent to the service in the form of a SOAP message and the response is sent back to the client as a SOAP message too. According to these request and response messages, the values of two QoS parameters, namely the response time and the availability[2] are deduced. The equations (1) and (2) are used to compute respectively their values.

- Response time

  The response time is the difference between the time when the SOAP response is received and the time when the user's request is sent as a SOAP message. Every time, a Web service is called, the response time is computed.

Response Time = Time taken to complete the response - Time taken for user request (1)

- Availability

  Availability is the probability that the system is ready for immediate consumption when invoked. After a service call, when no response is returned, the availability is set to 0.

$$Availability = 1 - \frac{downtime}{\text{unit time}} \quad (2)$$

We also consider the price as a business parameter. The price is proposed by the provider and we assume that its value is constant.

n this paper, we focus on three parameters namely, the response time, the availability and the price. However, our system can be extended to handle other parameters such as the reliability, the integrity, and the accessibility of services. As is mentioned in (Karthikeyan and SureshKumar, 2014) these parameters can be estimated based on cost or time. The monitor is a part of the Web service based application. The system sends a service invoke, and receives a reponse over the well-known XML and SOAP messages. While invoking a given Web service, the monitor records information about the Web service invoking time.

When the service response is received, the monitor records information about the Web service response time, and the availability. A monitoring process is launched over the services belonging to the initial set *SWS1*. During the invocation of the Web services, information about each service is recorded in data storage. This information is used to deduce the availability of each Web service. Based on the computed availability values, the services are classified into available and unavailable services. The first step ends when all available services are classified into the set *SWS2*.

[1] http://www.soapuser.com/basics1.html

[2] The Technical Guidance for the implementation of INSPIRE View Services.

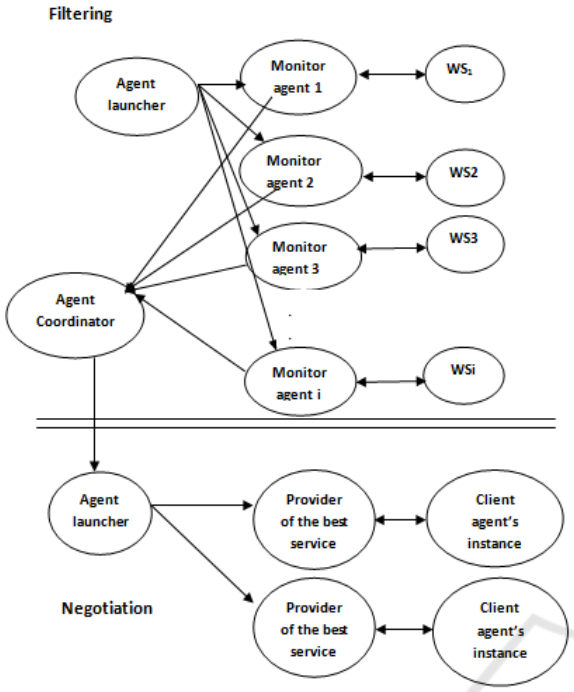

Figure 1: The agent based filtering process.

In the second step, the input of the filtering process is the set *SWS2*, and its output is *SWS3*, the set of services having a high and medium quality. In the course of this step, services with poor quality are filtered out. By definition, poor quality is related to QoS parameters that are far from the users' expectations expressed in terms of QoS parameters. Since these parameters change over time and depend on external factors, it is necessary to intercept their values continuously. Thus, a second monitoring process over the selected services belonging to the set *SWS2* is required. The problem here is set when a single monitoring process round is adopted. In this case, the choice of the services that must be removed can be modified. A single monitoring process round is when the service is intercepted only at a unique time *t*. Actually, the QoS parameters change frequently. During a small fraction of time, their values continuously fluctuate. For example, we consider two services W*S1* and W*S2*. The response time of each service is recorded at two different times *t1* and *t2*. The response time values of W*S1* and W*S2* at *t1* are equal respectively to 2s and 6s. However, at *t2*, the response time are 5s for W*S1* and 3s for W*S2*. We conclude that at *t1*, W*S1* is better than W*S2*; however by comparing their values at *t2*, W*S2* performs better than W*S1*. So, when applying the filetring process at *t1, WS2* will be filtered out although of it is better than W*S1*. To avoid such mis-evaluation of the Web service performance, a multi-round monitoring process is needed to assess the real Web service performance. Each service belonging to the set SWS2 is called repeatedly in order to record and store the values of the QoS parameters. These values are then compared, sorted and used to deduce the minimum and maximum QoS values related to each Web service.

In order to compute the users' utility generated by the overall QoS parameters, we adopt the utility function presented by Zheng et al., (Zheng et al., 2012). A weighted sum function noted as U(m) represented by equation 3 is adopted to compute the general utility of a given offer *m* containing *n* negotiation objects (QoS parameters).

$$U(m) = \sum_{i=1}^{n} W_i \times u_i(x) \qquad (3)$$

We consider that x is the value of a given QoS parameter, and x_{best} and x_{worst} are respectively their best and worst values. $u_i(x)$ the normalized value of x is defined by equation 4.

$$u_i(x) = \frac{(x - x_{worst})}{(x_{best} - x_{worst})} \qquad (4)$$

By comparing the utility generated by the maximum and minimum QoS values with the user's utility generated by its preferred values, the system will categorize the services according to their performance into mediocre, medium and best services.

We note by Min_{resp} and Max_{resp} respectively, the minimum and the maximum values of the response time recorded during the repetitive monitoring process, while Min_{avai} and Max_{avai} are the minimum and the maximum values related to the availability. We also consider Val_{resp} and Val_{avai} the preferred users' values of respectively the response time and the availability. We note by U_{min}, U_{max} and U_{pref} the user's utilities generated repectively by the user's worst, best and preferred QoS values. Here, we assume that the user's utility increase (resp. decrease) when the web service reponse time and price values decrease (resp. increase) and when the avalability value increase (resp. decrease). The minimum utility of the client (U_{min}) will be equal to the sum of the weighted maximum response time and price values with the weighted minimum availability value. Howerver, the maximum utility of the client (U_{max}) will be equal to the sum of the minimum weighted response time and price values with the maximum weighted availability value.

The set of the mediocre (worst) services are determined by comparing the utilities generated by

the users' QoS preferred values with the utility generated by the maximum response time and price with the minimum availabiltiy value recorded during the monitoring process. Here, we assume that the price does not change during the monitoring rounds. A service is a part of the set of the worst services only if U_{min} is lower than U_{pref}. If the user's utility generated by the user's worst values of QoS parameters recorded during the monitoring is lower than the utiliy generated by the preferred user's parameters, then the chance that such a service does not match the user's expectation in the future will be high.

During the third step, if the performance of services belonging to the set *SWS3* is quite similar, another criterion must be considered in conjunction with U_{min} and U_{max} in order to distinguish between the services.

The second criterion is related to the variation of the QoS parameters values. The reliability of the Web service depends on the variation of its QoS parameters. If the variation of these parameters is high, then the service is considered as unstable and not reliable. However, when this variation is minimal the service is qualified as stable and reliable. The lower the variability of the QoS parameters, the better the service is considered. During a many rounds of monitoring process, different values of response time and availability are recorded. Many variation values coresspond for each Web service. The worst variation is determined for each service by comparing the different variation values. The worst value matches the maximum variation.

- The variation

  The variation is the difference between a QoS value recorded during a current monitoring round, and another QoS value recorded during a previous monitroing round.

We note by $VarRt_{resp}$ and $VarRt_{avai}$ the maximum variation rate of respectively the response time and the availability. Med_{resp} and Med_{avai} are the medians of the maximum variation values respectively of the response time and the availability. The services with the worst variation values that are greater than the median are considered of lower performance, while the services with the worst variation values that are lower than the median are of better performance.

The categorization of Web services into medium and high quality depends on two factors namely, the variation, the U_{min} and $U_{max.}$

A Web service is ranked of a high performance quality only if conditions 1, 2 and 3 are checked:

Condition 1: U_{pref} is lower than or equal to $U_{min.}$ The user's goal is to increase its own utility. The first condition requires that the utility generated by the preferred response time, availability and price is lower than or equal to the utility generated by the maximum service response time and price as well as the minimum service availability recorded during the monitoring.

Condition 2: U_{pref} is lower than or equal to $U_{max.}$

Condition 3: $VarRt_{resp}$ and $VarRt_{avai}$ are lower than or equal to respectively the Med_{resp} and $Med_{avai..}$

A Web service is regarded as having a medium quality if condition 4 or condition 5 is checked.

Condition 4:
- U_{pref} is lower than or equal to U_{min}
- $VarRt_{avai}$ is greater than or equal to Med_{avai}

Condition 5:
- U_{pref} is lower than or equal to U_{max}
- $VarRt_{resp}$ is greater than or equal to Med_{resp}

4 EXPERIMENTATION

To get an in-depth investigation of the proposed approach, we have implemented a negotiation system considering the filtering process based on the Java programming language and the multi-agent platform MADKIT. We have deployed our system in a Toshiba satellite L775 version with Intel(R) Core (TM) i5 2410M CPU 2.3 GHz CPU and 4GM RAM to simulate the environment. By these experiments, we want to highlight three aspects. First, we prove the role of the proposed filtering approach in decreasing the CPU time of the Web service discovery process. Second, we demonstrate that an increase of the monitoring rounds has no negative effect on the CPU time of the selection process. Third, we prove the scalability of the implemented system. In order to implement the QoS monitoring process in the core of each monitor agent, we adopt the SOAP message calls. We use the framework SAAJ[3] (SOAP with Attachments API for Java) in order to implement the SOAP message calls. We design a user interface that enables the user to sets its preferences in terms of the values and weights of the QoS parameters as is presented in the previous section. According to these initial preferences, the set of functionally similar Web services is selected. Information about the URL of each service is extracted from the UDDI registry and they are communicated to the launcher agent. The

[3] http://docs.oracle.com/javaee/5/tutorial/doc/bnbhg.html

first role of the launcher agent is to create for each Web service a monitor agent. Each monitor agent has as attribute the URL of a Web service. The second role of the launcher agent is to create an instance of the coordinator agent. The monitor agents start the interception of the Web services by invoking them simultaneously.

In order to invoke a given Web service, an instance of the *soapConncetion* class is created. Then, the method call is applied over the latter instance. The line code that represents the call method is as follows: *SOAPMessage soapResponse = soapConnection.call(createSOAPRequest(), url)*. The method *createSOAPRequest()* creates a SOAP request message envelop which contains the HTTP address of each Web service. This method returns a SOAP message. When the SOAP response is received, the *printSOAPResponse(soapResponse)* is used in order to de-serialize the content of the SOAP envelop. When a monitor agent terminates the monitoring process, it sends the collected information to the coordinator agent. The latter has the role of making the filtering process or to launch another monitoring process as is explained in the previous section.

In the first series of tests, we set the number of Web services and the monitoring rounds respectively to 30 and 5. Here, the choice of the number of monitoring rounds is independent of Web services number. In Figure 2, we have reported the CPU time produced by varying the user's QoS preferences in terms of the values and the weights. In these tests, we consider three QoS parameters namely, the price, the response time and the availability. The results show that when the filtering method is adopted, the values of the CPU time range between 2.1 and 3.21 minutes. However, in the opposite case the values exceed 5 minutes. In fact, during the filtering, the services with QoS parameters that are far from the user's preference are filtered out. In case the filtering is omitted, these services can make the negotiation process longer.

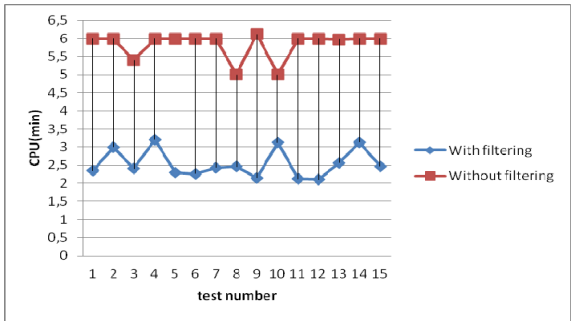

Figure 2: Comparison of the CPU time (λ=5 and 30 available and unavailable WSs).

Indeed, the gap between the real Web service performance and the user's preference causes conflicts and makes the negotiation process longer. Moreover, during the normal negotiation, unavailable services can result in an endless negotiation process which prevents negotiators from achieving an agreement. During the filtering process such services are removed.

In the second series of tests, we change the value of λ and we keep the same number of Web services specified in the first test. We consider λ1=20. The results are depicted in Figure 3. By comparing Figure 2 with Figure 3 we deduce that the CPU time increases slightly when the number of the filtering rounds increases. However, it remains in most cases lower than the CPU time recorded when the filtering is ommitted. In fact, when the monitoring rounds become higher, the filtering process takes more time to come up with the best candidate services. In overall, the increase of the filtering rounds has no great consequences on the CPU time.

In the third series of tests, we launched our system by varying the initial number of Web services. We consider in the initial set of Web services 10, 20 and 30 Web services respectively in the first, the second and the third tests. The results are reported in Figure 4.

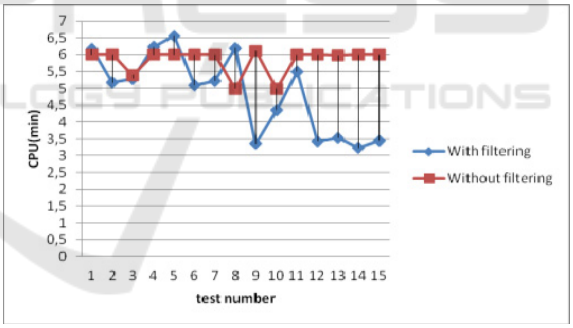

Figure 3: Comparison of the CPU time (λ1=20 and 30 available and unavailable WSs).

We note that in the case where 10 Web services are considered, the CPU time values range between 0.5 and 1.89 minutes. When 20 Web services are considered, the CPU time values range between 1 and 2 minutes whereas, in the case of 30 Web services the values range between 2 and 3 minutes. We conclude that when the number of services passes from 10 to 20, the CPU time was not greatly influenced. However, when the number passes from 20 to 30, we notice that the CPU time has generally increased by around 1 minute. From these results, we conclude that the increase in the CPU time and the Web services number are not proportional. This

means that an increase in the Web service number by a given value *x* will not always result in an increase in the CPU time by a fixed value *y*. Other factors influence the CPU time such as the performance of the selected services, the distance between the user's preferences and the Web services QoS parameters as well as the number of unavailable services. The fewer the unavailable services are, the better the CPU time will be. In the favourable cases, when all services are available in the set of initial Web services, the first step of filtering process will be omitted. This will speed up the negotiation process and makes shorter the CPU time.

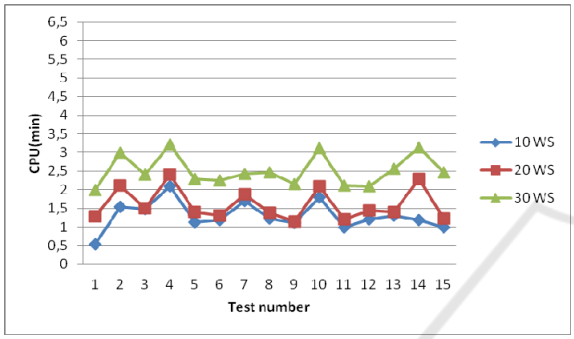

Figure 4: The System Scalability.

5 CONCLUSIONS

With the rapid growth of Web services providing functionally similar services, advanced discovery systems based on the QoS parameters must be adopted. The challenge is to come up with an efficient system while keeping its generated results reliable. Actually, the existing discovery systems generate a huge number of candidate services for the selection process. Considering all these services will be a waste of time. To deal with this problem we have presented a Web service filtering method based on a multi-round QoS monitoring method. The idea is to filter out services that are unavailable and under the users' expectations in terms of QoS requirements. Actually the QoS parameters are dynamic and change frequently depending on external factors. Adopting a multi-round monitoring process ensures an overview of the services performance. In order to distinguish between the Web service performance qualities, we introduced in the filtering process the variation criteria. The services that are characterized by a high fluctuation of their QoS parameters are considered unstable whereas those that have low fluctuations are considered stable. Future work is related to enhancement of the monitoring concept. In this work we consider the monitor is part of the WS based application. This can make sometime problems when resources are limited and the requests cannot be sent to the Web service. The latter issue needs to be addressed.

REFERENCES

Bentahar J., Maamar Z., Wan W., Benslimane D., Thiran P., and Subramanian S., (2008), Agent-based communities of Web services: An argumentation driven approach, *Service Oriented Comput. Appl., Vol.2, no. 4, pp. 219–238.*

Benaboud R., Maamri R., Sahnoun Z., (2016), PrefWS3: Web Services Selection System Based on Semantics and User Preferences, *Informatica, Vol 40, No 2.*

Bellakhal R., and Ghédira k., (2016), A multi-agent-based negotiation system for web service selection. *I. J. Knowledge and Web Intelligence 5(4): 253-286.*

Chen H-P., Zhang C., and. Yang G., (2010), A Model for Managing and Discovering Services Based on Dynamic Quality of Services. *Journal of Networks, Vol 5, No 8, pp.888-895.*

El-Awadi R., Esam M., Rizka M., Hegazy A., (2014), A Framework for Selecting Cloud Service Providers Based on Service Level Agreement Assurance, *Proceedings of the International Conference on Grid Computing and Applications (GCA).*

Huang. X (2013), UsageQoS: Estimating the QoS of Web Services through User Communities, *ACM Transactions on the Web (TWEB), Vol8 n.1, pp.1-31.*

Karray A., Teyeb R., Ben Jemaa M., (2013), A heuristic approach for web-service discovery and selection, *International Journal of Computer Science & Information Technology (IJCSIT), Vol 5, Issue 2.*

Karthikeyan, J & M, SureshKumar. (2014), Monitoring QoS parameters of composed web services. *2014 International Conference on Information Communication and Embedded Systems, ICICES 2014.*

Linlin. W, Saurabh. K. G, Rajkumar. B, (2013), Automated SLA Negotiation Framework for Cloud Computing, Cluster, Cloud and Grid Computing (CCGrid), *13th IEEE/ACM International Symposium.*

Napoli C. Di, Pisa P., and Rossi S., (2013). A market-based negotiation mechanism for QoS-aware service selection, *The 15th International Workshop on Agent-Mediated Electronic Commerce.*

Ouadah, A., Hadjali, A., Nader, F. et al. (2018), SEFAP: an efficient approach for ranking skyline web services, *Journal of Ambient Intelligence and Humanized Computing, pp 1–17.*

Zhou C., and Chen H., (2009), An Objective and Automatic Feedback Model for Qos Evaluation, *ICIS '09 Proceedings of the 2nd ICIS.*

Zheng, X., Martin, P. and Brohman, K. (2012), 'Cloud service negotiation: concession vs. tradeoff approaches', *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE Computer Society, pp.515–522.*