A Web Service Selecting Model Based on Measurable QoS Attributes of Client-Side

Shu-jia Wang
School of Software
Shanghai Jiao Tong University, SJTU
Shanghai, P.R.China
ggoodd@sjtu.edu.cn

Hao-peng Chen
School of Software
Shanghai Jiao Tong University, SJTU
Shanghai, P.R.China
chen-hp@sjtu.edu.cn

Abstract— **As more and more Web Services appear on the public Internet, Quality of Service (QoS) becomes one of the most important factors for Web Service selection. Meanwhile, accuracy and speed of Web Service selection come to be the new barriers. Focusing on those QoS attributes that are measurable on client-side, this paper proposes a new Web Service selecting model, extending the general searching architecture. In our model, a multiple -level cache architecture is implemented to speed up the selecting process. And inside of the architecture, similarities of clients and caches are taken into account to improve the accuracy of selection with historical service information**.

Keywords-web service; selecting; QoS; cache; similarity

I. INTRODUCTION

Web Services have become one of the most promising technologies in distributed computing. Meanwhile, QoS, giving a way to distinguish and rank services with similar functionality, is taken into account during Web Service selection, as more and more Web Services come out.

In order to improve the accuracy with the constraint of QoS, feedback mechanism is implemented during the process of Web Services selection. By making use of historical information of all the clients having experienced the service, average values of QoS attributes about this service is gained. But since clients reside in a heterogeneous environment, QoS experienced by clients for the same Web Service can vary widely. Therefore, QoS based on server-side may not be too much valuable for clients, and the average QoS of all clients is also not very efficient.

In the general Service-Oriented Architecture (SOA), client requesting a service has to send the request to the Service Registry at first. After receiving a reply about service provider's information, the client could access the service provider to get the service. As the communication between client and Service Registry is usually not so fast, the requesting and responding processes become the bottleneck of QoS that clients experience besides the service itself.

II. RELATED WORK

QoS is an important factor in selecting Web Service providers when there is more than one provider offering the same or similar service. As a result, there are many different definitions for QoS of Web Service. In [1], attributes of QoS are defined as availability, accessibility, integrity, performance, reliability, regulatory and security. It points out that QoS attributes should contain availability, response time, throughput and security etc in [2]. Based on the unique features of SOA, availability, performance, reliability, dynamic discoverability, dynamic adaptability, dynamic composability are summarized as six quality attributes[3].

Web Service Discovery mechanisms have been reviewed extensively in the work of [11]. In general, these mechanisms do not usually take into account QoS concerns such as the response time. Without affecting the existing UDDI search facilities, reference [9] describes a mechanism for discovery in a generalized environment that stores a dynamic number of categories of different Web Services, and has implemented UDDI search wrapper to take into account possible QoS characteristics available in transparent way.

Currently, there are two popular ways to identify the QoS. One is the description of service provider; and the other is the feedback from clients that have implemented the service. There are some solutions that support Web Service selection based on QoS of service provider. One is UDDIe [6] in which QoS information is advertised by the provider. WSME [7] enables requirements to be specified by both clients and providers.

Reference [4] analyzes the factors that contribute to the effective performance experienced by a client and shows the importance of client grouping and profiling, in which will contribute critical information to be used for dynamic selection of Web Service based on performance. A high level web service recommendation based on performance experienced by the client is also proposed in [5], which establishes an on-going analysis framework to help build Web Service profiles and client profile to estimate the client-side performance.

III. FRAMEWORK OF NEW WEB SERVICE SELECTING MODEL

Clients are end users of Web Services, and QoS of a Web Service is expressed on the client-side at last. So it makes sense to do the Web Service Selection based on QoS properties of the client-side. QoS experienced by client-side is affected by many factors [4]. For example, physical location is a useful factor as the clients residing closely experience a similar network

This paper is supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.659

environment. So that client grouping is important for the selection of Web Service.

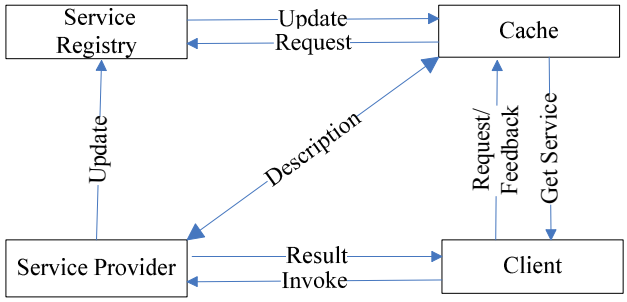

Figure 1. Framework of the new WS selecting model

Among so many attributes of QoS shown in Sector 2, we focus on measurable attributes, such as latency, transfer rate and response throughput. Based on these measurable QoS attitude on client-side, we propose a new Web Service selecting model, of which the framework is shown as Fig 1.

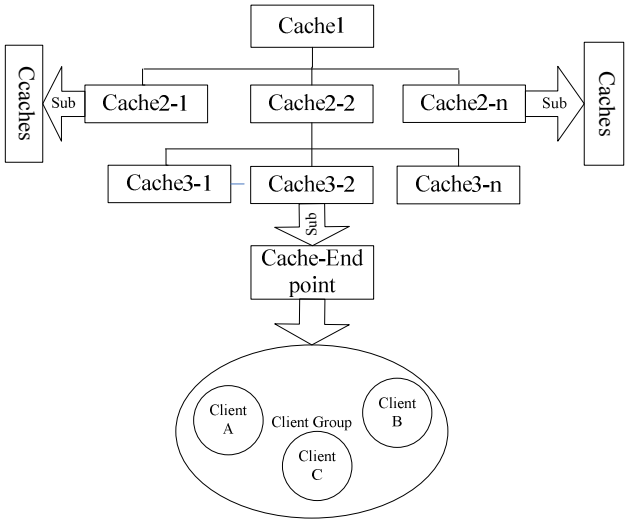

Figure 2. Cache architecture

First, clients are clustering in group mainly according to their network environments. Because one group of clients shares a unique Cache to request services and get information of the comfortable service. It is much faster for the clients to access their Cache than to access the Service Registry.

The Cache is multiple-level architecture as shown in Fig 2. Service Registry communicates with top-level Caches which map several down-level Caches. Every Cache of end point not only keeps historical information of services implemented by clients of its group, but also a proxy that clients request services. There are several tables kept in Cache as follows.

- ServicesList: ServicesList keeps QoS information of services that clients of this group have gotten. The information should be kept the same with that in Service Registry and the approach will be illustrated in the latter section.
- RecordsOfServiceImplements: For every service in the ServicesList, there is at least one record in the RecordsOfServiceImplements. Clients that have accessed this service as well the actual service implementing quality are included in the record.
- SimilarityTable: The similarity of clients or Caches is stored in this table. This table will updates regularly according to records in RecordsOfServiceImplements.

```
selection(function, QoS)
1 if ServiceList_localCache : set_service ← service_function
2    for each item in set_service
3       do if item.num ≥ N
4          then
5             calculate QoS_estimate
6                if QoS_estimate matches QoS
7                   return item
8 else do
9          ServiceList_up−levelCache : set_service ← service_function
10         repeat steps 2 − 7
11      until goto Service Registry
12 return service_function,QoS
13 add service_function,QoS to Caches
```

Figure 3. Selection algorithm

As shown in Fig 3, when a client wants to apply a service, it first sends its request to the cache of its group. Then the cache traverses the ServicesList. If there is a service satisfying the functional requirement as step 1 in Fig 3, local Cache will research the historical records of this service in RecordsOfServiceImplements. Then as step 3-7, if there are enough records for estimation, with the similarity of SimilarityTable, an estimated QoS attribute is gained to match the client's request. If no service could match the request, the selection goes to step 9 that Cache sends it to the up-level Cache. After step 10, that means no services are found, the top Cache will access the Service Registry for the service. In the end, this new service will be added into the ServicesList of all researched Caches. After implementing the service, QoS and other related information will be sent to the local Cache.

IV. ESTIMATE BASED ON SIMILARITY

Clients in a same or similar environment have similar experience about the QoS of a same service. We make a client grouping based on static factors similarity, such as location, network, and so on.

To simplify the description of our approach, we formalize the motivating problem at first.

$$C = \{C_1, C_2, ..., C_m\} \quad (1)$$

$$S = \{S_1, S_2, ..., S_n\} \tag{2}$$

$$Q_{ij} = \{q1_{ij}, q2_{ij}, ..., ql_{ij}\} \tag{3}$$

C is a group of clients with a limited similarity and C_i (1 <= i <= m) denotes a client. S is a set of services where S_i (1 <= i <= n) denotes one service. Q_{ij} is a vector representing the quality of S_j measured by client C_i. l denotes the number of regarded QoS properties. qk_{ij} denotes the value of the kth property of QoS of service S_j measured by client C_i.

To identify similarity of clients, we use a Euclidean Distance function as a basic similarity measure. For multitude property of QoS, if there is a set of QoS constraint, every single property should be calculated and considered separately, since different property has less connection with others. So, we discuss similarity with single property of QoS first.

Similarity of two clients C_a and C_b together having implemented services S_n is determined as:

$$D_{ab} = \sqrt{\sum_{k=1}^{n} (q_{ak} - q_{bk})^2} \tag{4}$$

D_{ab} is the Euclidean Distance of client a and client b. q_{ak} and q_{bk} is the quality of service k that client a and client b experienced. The n is the number of the records taken into account. The less the D_{ab} is, the more similar client a and b are. When a and b are the same client, the D_{ab} is 0. Euclidean Distance is efficient to identify the similarity between clients relatively. But it is not convenient to estimate the unknown value of QoS, so we extend the formula.

$$\overline{q_x} = \sum_{i=1}^{n} q_{xi} / n \tag{5}$$

$$W_{ab} = \sqrt{\sum_{i=1}^{n} \left(q_{ai} - \overline{q_a}\right)^2 / \sum_{i=1}^{n} \left(q_{bi} - \overline{q_b}\right)^2} \tag{6}$$

In (6), W_{ab} is the similarity weighting between client C_a and C_b on the property. For an unknown service S_x for client a, we can get the estimated value of the property with other clients that have implemented service S_x before.

$$q_{ax} = \overline{q_a} + \sum_{j=1}^{m} (q_{jx} - \overline{q_j}) W_{aj} / m \tag{7}$$

q_{ax} is the estimated QoS value of service S_x that client C_a maybe experienced. C_j (0 <= j <= m) is client that has implemented service S_x before. W_{aj} is the similarity weighting between C_a and C_j.

When there is no historical information of the service requested in the local group, the request will be delivered to up-level cache to search the service in other caches of the same level. Every cache matches one group of clients or one group of low-level caches. Average values of properties of QoS of all services are kept in the cache.

$$\overline{CQ_{ij}} = < \overline{q1_{ij}}, \overline{q2_{ij}}, ..., \overline{ql_{ij}} > \tag{8}$$

$$\overline{q_{lj}} = \sum_{i=1}^{m} q_{lij} / m \tag{9}$$

Formula (8) represents the quality of Service S_j measured by cache CA_i. As shown in (9), the average value of the kth property of QoS of Service S_j is measured by cache CA_i, and ql_{ij} denotes the average value of k property of QoS of Service S_j measured by the client C_l under Cache CA_i. To be mentioned, if cache CA_i is not the lowest level cache, ql_{ij} is the average value of the kth property of QoS of service S_j measured by the cache CA_l under Cache CA_i.

$$\overline{Cq_x} = \sum_{i=1}^{n} \overline{q_{xi}} / n \tag{10}$$

$$CW_{ab} = \sqrt{\sum_{i=1}^{n} (\overline{q_{ai}} - \overline{Cq_a})^2 / \sum_{i=1}^{n} (\overline{q_{bi}} - \overline{Cq_b})^2} \tag{11}$$

When no enough records of service historical information, the local cache will deliver the request to the up-level cache and search the service in other same level caches. Similarity among caches is calculated as (11) where average values of properties of QoS in caches take place of clients in (10).

$$q_{ax} = \overline{q_a} + \sum_{j=1}^{m} (q_{jx} - \overline{q_j}) W_{aj} / m \tag{12}$$

Having found the enough QoS records of the service, an estimated value for client C_a is calculated by (12), where CW_{ab} is used to take place of the similarity W_{aj} of client C_a and the clients in records.

V. UPDATE INFORMATION OF SERVICES

The most important fact for Cache-mechanism is that the information in caches must keep the same with that in the Service Registry. Since service providers maybe update or delete the service description, once there is difference between caches and Service Registry, the service request will fail.

Service Registry, taking UDDI for example, is usually organized around two fundamental entities that describe business and the service they provide. First entity is business entity, including provides information, service entity elements that represent the services, a category bag to categorize the business, and a unique key. Second entity is service entity, including information such as name and description, a unique service key, a category bag to categorize the service, a list of binding templates encoding the technical service information, and a reference to its host with a business key.

To clearly indicate the update for services of Service Registry, we abstractly describe the model of services published in Service Registry as following form:

$$< Key, Name, Description, AccessPo\,\text{int}, OverviewURl > \tag{13}$$

In (13), *Key* uniquely identifies a service registered in the Service Registry. *Name* is a readable service name. A general description of the service is stored as *Description*. *Access Point*

is the interface of the service. *OverviewURL* denotes the access location of the WSDL document. When a client requests this service, it can get the detailed description of service through the URL stored in *OverviewURL*, and then access this service by the address identified by *AccessPoint*.

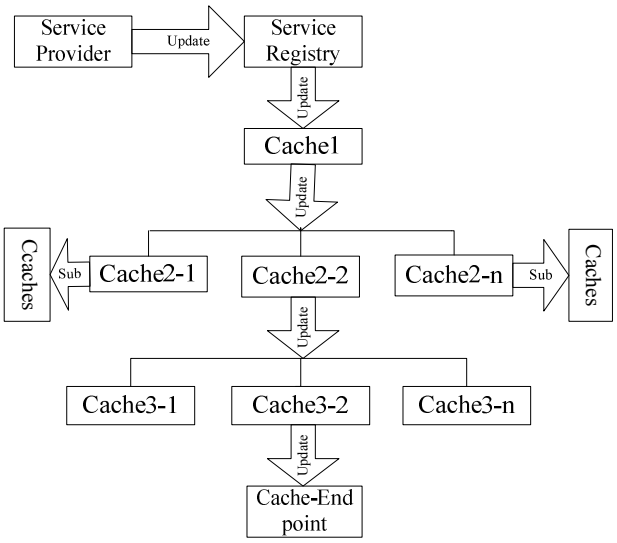

Figure 4. Update information of Caches

In our new model, there is a list for services to store caches of top level that keep information of the service as

$$< Key, CacheList >$$
$$CacheList = \{Cache1-1, Cache1-2, ..., Cache1-n\} \quad (14)$$

Key is the service key in (13). *CacheList* is caches having records of the service identified by *Key*. When updated information arrives, Service Registry updates the information and then pushes it to the up-level caches. After updating, the up-level caches push the information to the down-level caches as Fig 4. At last, all the service information in Caches and Service Registry are fresh.

VI. SIMULATION EXPERIMENTAL RESULTS

Aiming to test the feasibility of this model, five clients are chosen to imitate the real clients requesting web services. Client A, client B and client C are in one group, and D, E, F are in other groups. In the following table, the latency for each client implementing different services is recorded. It needs to be mentioned that the latency is an average value in this table.

TABLE I. HISTORICAL SERVICES RECORDS OF CLIENTS

Clients	S1(ms)	S2(ms)	S3(ms)	S4(ms)	S5(ms)
A	495	201	346		
B	412	150	280	378	
C	320	109	229	254	
D	2289	1838	2005	2179	1904
E	1967	1682	1863	1923	1789
F	763	325	453	518	486

According to the table, the similarity of A, B, C could be gotten through (6). The similarity described by Euclidean Distance is shown in the following table. This table is the SimilarityTable stored in the Cache of this client group.

TABLE II. SIMILARITIES AMONG CLIENTS

Clients	A	B	C
A	1	1.12216	1.38904
B	0.89114	1	1.23783
C	0.71992	0.80787	1

Client A would request service S4 and sends the request to the Cache. The Cache first accesses the ServicesList and if the service exists, then accesses the RecordsOfServiceImplements that keeps the record of the service feedback by clients in this group.

TABLE III. LAST LATENCY RECORDS OF SERVICE S4

Service	Client	Latency(ms)	Date
S4	B	377	06/05/2008
S4	C	251	06/05/2008
S4	A	430	06/04/2008
S4	B	368	06/04/2008
S4	B	380	06/02/2008

An estimated quality value could be got by (7). Then, Q_{a4} = 436.14ms. Without the similarity, the average Q_{avg} is 361ms. And the real result for Client A is 434ms. So, this model supplies a more approximate value.

TABLE IV. AVERAGE LATENCY RECORDS IN CACHES

Caches	S1(ms)	S2(ms)	S3(ms)	S4(ms)	S5(ms)
Ca1	409	153.33	285	316	
Ca2	2128	1760	1934	2051	1846.5
Ca3	752	316	430	509	480
Ca4	908	387	513	624	532

When no records about service S5 requested by Client A are stored in this Cache, the request is delivered to up-level Cache to search the service. In the up-level Cache, information as following is kept.

Similarities among Caches are calculated as in the following Table 5. By (12), using similarity of Caches to stand for similarity of clients, we can get the estimated value of service S5 for client A. Q_{a5} = 332.82ms compared with the average Q_{avg} is 1393ms. And the real value is 341ms. The estimated value is much more accurate than the average value.

TABLE V. SIMILARITIES AMONG CACHES

Caches	Ca1	Ca2	Ca3	Ca4
Ca1	1	0.14776	0.57964	0.47834
Ca2	6.76764	1	3.92277	3.23725
Ca3	1.72522	0.25492	1	0.82525
Ca4	2.09055	0.30890	1.21176	1

VII. PERFORMANCE OF THIS MODEL

The efficiency of this WS selecting model depends mostly on two facts. One is how fast the aim web service is found. The other is the accuracy of the estimation based on the similarity and historical records.

Cache mechanism is based on the theory that services being requested are possible to have been accessed before.

For our model, once a service has been accessed, time of requesting this service in future will be shortened very much. This model can provide a remarkable improvement about access time in most situations except one, that the service has never been implemented by any clients at all, meaning Service Registry is still accessed at last after all levels of Caches have been visited.

Compared with the average value of QoS properties of overall clients, our model offers a more accurate estimated value. By calculating similarities among clients and Caches, the actual network environments and local equipment performance as well other effective factors are taken into account. And the closer those clients are in the Cache architecture, the more accurate the estimation is.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new model for Web Service Selection based on performance experienced by clients. Since different client may have a heterogeneous environment, we take advantage of similarity of clients in a similar environment to estimate the quality of service, taking place of the average quality, so that the accuracy of Web Service Selection with constraint of QoS is improved apparently. Along with the similarity, Cache-mechanism for services is applied in this model to speed up the selection. With this model, Web Service Selection becomes more efficient than before.

Multitude attributes will be taken into account in the future research. Now this model just focuses on one single measurable attribute of QoS, but future Web Service Selection would be with more complex constraint of QoS. Further, how to deal with composed services is also a challenge in this Web Service Selection model.

REFERENCES

[1] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services," http://www.ibm.com/developerworks/library/ws-quality.html, January 2002.

[2] Menasce D.A., "QoS Issues in Web Services," IEEE Internet Computing, November-December 2002, vol.6, pp. 72-75.

[3] Si Won Choi, Jin Sun Her, and Soo Dong Kim, "QoS Metrics for Evaluating Services from the Perspective of Service Providers," DOI 10.1109/ICEBE.2007.107.

[4] Niko Thio, Shanika Karunasekera., "Client Profiling for QoS-Based Web Service Recommendation," the 12th Asia-Pacific Software Engineering Conference (APSEC'05).

[5] Niko Thio, Shanika Karunasekera, "Web Service Recommendation Based on Client-Side Performance Estimation," the 2007 Australian Software Engineering Conference (ASWEC'07).

[6] Thomas Erl, "Service-Oriented Architecture a Field Guide to Integrating XML and Web Services," Prentice Hall PTR, 2004.

[7] Eric Newcomer, Greg Lomow, Understanding SOA with Web Services, pp. 123-124.

[8] V. Deora, J. Shao, W. A. Gray, N. J. Fiddian, "Modeling Quality of Service in Service Oriented Computing," Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06).

[9] Yannis Makripoulias, Christos Makris, Yiannis Panagis, Evangelos Sakkopoulos, Poulia Adamopoulou and Athanasios Tsakalidis, "Web Service Discovery Based on Quality of Service," 1-4244-0212-3/06/©2006 IEEE.

[10] Bin Li, Xiao-yan Tang, Jian Lv, "the Research and Implementation of Services Discovery Agent in Web Services Composition Framework," Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou (ICMLC'05), 18-21 August 2005, Volume 1, pp.78 – 84.

[11] C.Schmidt and M.Parashar, A peer-to-peer approach to web service discovery in World Wide Web, vol. 7 (2), pp. 211-229.

[12] Maximilien, E.M., Singh, M.P., "A framework and ontology for dynamic Web services selection," Internet Computing, IEEE Volume 8, Issue 5, Sept.-Oct. 2004, pp.84 – 93.

[13] Dr. Srinivas Padmanabhuni, Bijoy Majumdar, Mohit Chawla, Ujval Mysore, "A Constraint Satisfaction Approach to Non-functional Requirements in Adaptive Web Services," International Conference on Next Generation Web Services Practices (NWeSP'06).

[14] Niko Thio, Shanika Karunasekera, "Automatic Measurement of a QoS Metric for Web Service Recommendation," 2005 Australian Software Engineering Conference (ASWEC'05).

[15] Tao Yu, Yue Zhang, Kwei-Jay Lin, "Modeling and Measuring Privacy Risks in QoS Web Services," IEEE. Enterprise Computing, E-Commerce, and E-Services, 26~29. 2006(6).

[16] N.W. Lo_, Chia-Hao Wang, "Web Services QoS Evaluation and Service Selection Framework - A Proxy-oriented Approach," 1-4244-1272-2/07/©2007 IEEE.

[17] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA Service-Oriented Architecture Best Practices, Prentice Hall, 2005.

[18] BangYu Wu, Chi-Hung Chi, Shijie Xu, "Service Selection Model Based on QoS Reference Vector," 2007 IEEE Congress on Services (SERVICES 2007).

[19] Dr. Srinivas Padmanabhuni, Bijoy Majumdar, Mohit Chawla, Ujval Mysore, "A Constraint Satisfaction Approach to Non-functional Requirements in Adaptive Web Services," International Conference on Next Generation Web Services Practices (NWeSP'06).

[20] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, Tony Newling, Patterns: Service-Oriented Architecture and Web Services, IBM Redbooks, ibm.com/redbooks.

[21] Rashid, M.M. Alfa, A.S. Hossain, E. Maheswaran, M, "An analytical approach to providing controllable differentiated quality of service in Web servers," IEEE. Parallel and Distributed Systems. 16(11):1022~1033. 2005(11).

[22] Liangzhao Zeng, Boualem Benatallah. Etc, "QoS-Aware Middleware for Web Services Composition," IEEE Transactions on Software Engineering 30(5):311-327. 2004(5).