

Design and Implementation of a Policy-Based Monitoring System for Web Services*

LI-DER CHOU, CHIEN-CHI CHEN AND PO-CHIA TSENG

Department of Computer Science and Information Engineering

National Central University

Chungli, 320 Taiwan

E-mail: cld@csie.ncu.edu.tw

Follow the all-pervading of Web Services on consequence of the Quality of Service (QoS) for Web Services becomes one of the most important issues progressively. Network service which is a dynamic system has all sorts of unpredictable factors. Those as we mentioned above are crucial to the QoS of Web Services, such as availability, reliability and security. In the paper the issue of effective service monitoring for the sake of improving the QoS of Web Services is considered. A policy-based web services monitoring system comprising a service monitor control center and a plurality of service monitors is proposed in the paper to achieve the requirements of service requesters, where the service monitor control center collects and monitors the status of service monitors. Policies can be setup in terms of Service Level Agreement (SLA) and states of service monitors. For load balancing, monitoring tasks are assigned to the service monitors in the light of SLA and policies for effectively monitoring QoS of web services. Accordingly, the novel policy-based web services monitoring system is implemented, and its superiority and feasibility are demonstrated.

Keywords: web services, quality of service, service monitor, policy-based, service level agreement

1. INTRODUCTION

The Web Services technology had comprehensively applied to many areas in recent years [1]. Web Services is a kind of software component, which is based on existed network protocols and related standards, such as eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP) [2], and Web Services Description Language (WSDL) [3] and so forth. By using above standards, the Web Services technology can be used to serve the other applications, such as Service Oriented Architecture (SOA) [4], network management [5], employment Services [6], or bioinformatics [7]. It is shown that the Web Services technology is open-standard and interpretable across platforms and languages. So the Web Services can be integrated dynamically, loading balanced, and ease of units upgrade, like a distributed system. Moreover, the heterogeneous system can easily communicate and integrated with each other.

There are unpredictable factors that will affect the service because of the Web Service is a dynamic system [8]. Thus, the service providers are hardly able to guarantee the

Received October 31, 2007; accepted June 3, 2008.

Communicated by Jonathan Lee, Wei-Tek Tsai and Yau-Hwang Kuo.

* This research was sponsored in part by Department of Industrial Technology, Ministry of Economic Affairs under the grant No. 96-EC-17-A-02-S1-029, and by National Science Council, R.O.C. under the grants No. NSC 96-2628-E-008-004 and NSC 96-2627-E-008-001.

real-time Quality of Service (QoS) in the best-effort network environment, such as wireless networks, heterogeneous networks [9, 10], and optical networks [11]. In order to improve the QoS [12], the provider and requester have first to monitor QoS parameters. And the QoS parameters for the network traffic, such as reliability, availability, security, and performance, affect quality greatly. Besides, if requesters hope the QoS to be guaranteed, there must be a mechanism to define QoS. So the concept of Service Level Agreement (SLA) has become an important issue. The SLA is a contract which defines the criterion of QoS, and the QoS for requester can be protected by concluding the SLA. The main capabilities of SLA are to estimate whether the service provider achieves their QoS, understand the part of service needed to be improved, and provide related solutions to convince the requesters.

In order to confirm QoS of the Web Services exactly, there must be a mechanism to monitor the activities between providers and requester [13]. Except to an agent-based management for Web Service, reference [14] and reference [15] presented an agent-based management mechanism and a scheduling mechanism for Web services, respectively. The policy-based management system also provides a mechanism to monitor. There is an important role called the Service Monitor (SM). The SM have first to register its information in the Universal Description Discover Integration (UDDI) server and the SM can extract and collect data of QoS in the activates between providers and requesters. Furthermore, the SM will analyze the data and response to the data collection center, like UDDI server. The service provider can provide the requesters for preferred quality [16].

If there is no enough capability for SM to monitor activities, there is no accuracy for the monitoring result [17]. So, in addition to monitoring providers and requesters, the SM also requires the management of performance for itself. In order to manage the performance of SM, we proposed and implemented a policy-based Service Management Center (SMC) for Web Services. The performance of SM depends on many factors, such as the clock rate of Central Processing Unit (CPU), the size of memory, the access time of storage media, and the amount of service being monitored. So the management of performance for SM is necessary. In this paper, we proposed an architecture including the monitoring of CPU loading, system performance and the amount of service to being monitored. The architecture is centralized and policy-based so as to adapt the SLA, that is to say, the administrators can establish policies according to the requirements of SLA. And then the administrators dispatch the monitoring jobs to SM [18]. This paper is focused on improvement efficiency of monitoring mechanism.

There would be a common performance issue of centralized architectures. While the loading is increased, the centralized service management center might a bottleneck. The problem has been addressed and solved by a feasible solution in [19] by using clusters. Also, the traffic scheduler is solution to solve the problem of bottleneck [20]. Thus, this research focuses on the feasibility of policy-based management system. And we would discuss effectiveness of the architecture of centralized monitoring system.

This paper is organized as follows. In section 2, we describe the related work of monitoring mechanism. Next, the architecture of policy-based monitoring systems will be explained in section 3. The implementation and experimental results are discussed in section 4. Finally, conclusions are given in the last section.

2. RELATED WORK

2.1 Related Researches of Web Services QoS

The Web Services is widely adapted to integrated enterprises resources by more and more enterprises. Thus, the Web Services is growing and growing so that the QoS becomes an important issue for the service providers and requesters. However, the Web Services is characterized by ease to vary and unpredictable. Due to these reasons, maintaining the QoS between service providers and requester is a difficult issue.

Besides, there were different QoS for Web Services requirements in various Web Services. The QoS parameters and the selection of QoS parameters were described and proposed and in [21]. The comprehension of QoS must be clarified and defined. In this section, the QoS for Web Services requirements in [21] was introduced.

Performance

The performance of Web Services was defined as how fast a service request could be completed. It could be measured in terms of the followings parameters:

- **Throughput:** The amount of Web Services request can be served successfully in a period of time.
- **Process Time:** The length of time taken by a Web Services finish its sequence of activities.
- **Latency:** The round-trip delay (RTD) between sending a request and receiving the response.
- **Response Time:** The time required to complete a Web Services request.

And Response Time impacted the QoS most, and it was rather important than the others. When service requesters request services, service requesters take care the waiting time most [22, 23].

Reliability

Web services should be provided with high reliability. Reliability here represented the ability of Web Services to perform its required functions under stated conditions for a specified time interval. The reliability was the overall measure of Web Services to maintain its service quality. The overall measure of Web Services was related to the number of failures per day, week, month, or year. Reliability is also related to the assured and ordered delivery for messages being transmitted and received by service requestors and service providers.

Scalability

Web Services should be provided with high scalability. Scalability represented the capability of increasing the computing capacity of service provider's computer system and system's ability to process more users' requests, operations or transactions in a given time interval.

Capacity

Web Services should be provided with the required capacity. Capacity was the limit of the number of simultaneous requests which should be provided with guaranteed performance.

Robustness

Web Services should be provided with high robustness. Robustness here represents the degree to which Web Services could function correctly even in the presence of invalid, incomplete or conflicting inputs.

Exception Handling

Web Services should be provided with the functionality of exception handling. Since it was not possible for the service designer to specify all the possible outcomes and alternatives (especially with various special cases and unanticipated possibilities), exceptions should be handled properly.

Availability

Web services should be provided with high accuracy. Accuracy here was defined as the error rate generated by the Web Services.

2.2 Service Monitors

It's mentioned in [24], the service monitors could be separated by two parts. One was to probe the QoS, and the other was QoS broker. If the service requesters send a request for the QoS probing, QoS probing is to measure the QoS and record the data in QoS broker while the requesters had requested a service. QoS Broker would integrate the data and stored the data into the database. And the broker would generate a report about the QoS monitoring. In the database, the administrators could update the database periodically [16].

Besides, while the service requesters did not request the Web Services, QoS probing could pretend as service requesters to measure QoS by simulating the traffic. This was a mechanism to control the QoS Monitoring, and it could even add new classes of QoS based on the existed QoS classes.

The service must be published onto the Internet using the WSDL format in advance or discovered by service matchmaker [25]. If the service requester discovered the service, for the convenience of commit a SLA between service provider and requester, the service monitor would test the service by using URI in the WSDL of service. And the service providers and requesters could understand the QoS by the data provided by service monitor [26].

2.3 Policies for Service Monitors

In [27], management of a system had to involve some activities, such as monitoring the actions of system, managing the administrative policies, or modifying the behaviors of the system. For the convenience of administrators, the system had to handle the policies. In [27], policies also would affect the behavior pattern of the system. A policy was

not an action, but it was a sequence of objects needed to be finished or set of actions. If administrators affirmed some actions which only need to be carried out once, they did not need to establish a policy for this act. In [27], the six levels within a policy hierarchy were mentioned as follows:

- Societal policy (principles) such as ethics or laws [28];
- Directional policies (goals) such as organizational or corporate goals;
- Organizational policy (practices) such as contractual agreements or quality programs;
- Functional policy (targets) such as workload targets or quality measures [29];
- Process policy (guidelines) such as automated quality tracking, which may be partly encoded in computer programs [30];
- Procedural policy (rules) which are fully encoded in an executable computer language.

Because the performance of Service Monitor was affected by CPU resource and memory usage, the basic idea of policies for Service Monitor will be proposed as CPU-based, Memory-based, and CPU-Memory-based. Administrators could predefine policies for those resources and dispatch the monitoring jobs to different Service Monitors according to predefined policies [31].

3. ARCHITECTURE OF POLICY-BASED MONITORING SYSTEM FOR WEB SERVICES

In the paper, we design a policy-based monitoring system for Web Services, which is called Service Management Center (SMC), and it can manage the service monitors in the Web Services environment. As shown in Fig. 1, while a service requester wants to know the QoS of its processing services, QoS parameters can be monitored by the third party service monitor. However, single service monitor cannot measure all demanding QoS parameters for requesters in the meantime due to its limited capability. It is feasible to share the monitoring jobs between multiple service monitors. Thus, the SMC can assign jobs to multiple service monitors by using policies and inform service monitors to monitor QoS parameters. Besides, the SMC can collect the QoS parameters of other service monitors.

As shown in Fig. 2, a flow chart is given under the scenario with two service requesters, three service monitors, two services, and the SMC. Firstly, Service Requester 1 issues a QoS monitoring request to the SMC for the response time of a specific service. Then the SMC queries Service Monitor A about the response time of the specific service. Subsequently, Service Monitor A starts to monitor response time of the specific service. After response time of the specific service is recorded by Service Monitor A, and then the SMC starts to collect response time of the specific service from Service Monitor A. Finally, Service Requester 1 can acquire demanding QoS for the specific service from the SMC. In case of Service Requester 2, it issues a QoS monitoring request to the SMC for the response time and availability of a specific service. And Service Requester 2 similarly follows the procedure of Service Requester 1 to acquire the response time and availability of the specific service.

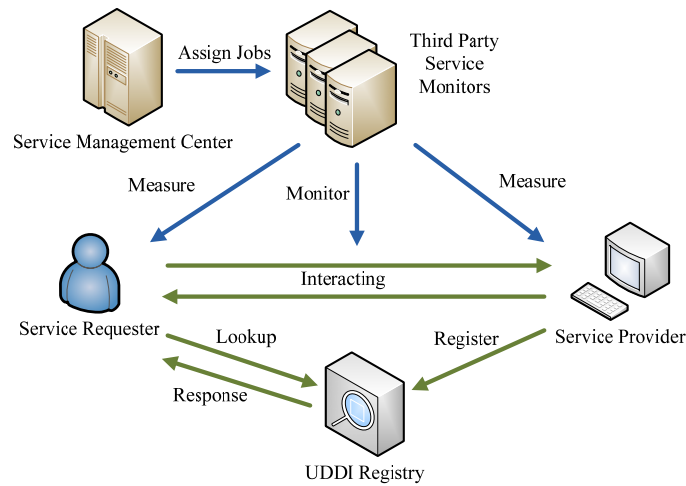


Fig. 1. Service monitor and service-oriented architecture.

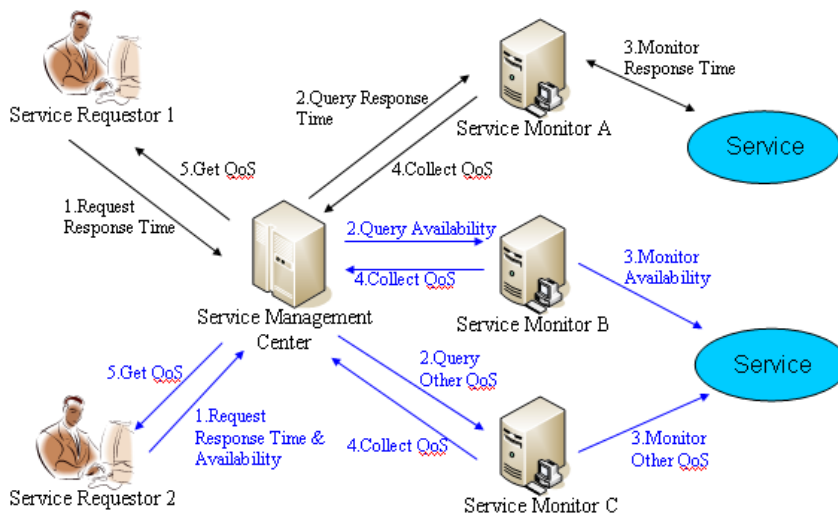


Fig. 2. The flow chart of the QoS query from service requester to SMC.

3.1 Policies Administration for Service Monitors

In order to manage service monitor's status, SMC has to customize the policies to manage criterions of assigning monitoring jobs. Administrators of SMC can determine policies they like and apply them to service monitors effectively. The design of policies considers factors influence the status of service monitors, and the policies define the threshold of influence by factors. It is determined the service level of service requester by the upper bound and lower bound, for example: The policy is defined as when the CPU usage of the system exceeds 60%, the priority of service monitor is set as low. If SMC assign the monitoring job next time, SMC will not assign the monitoring job to this

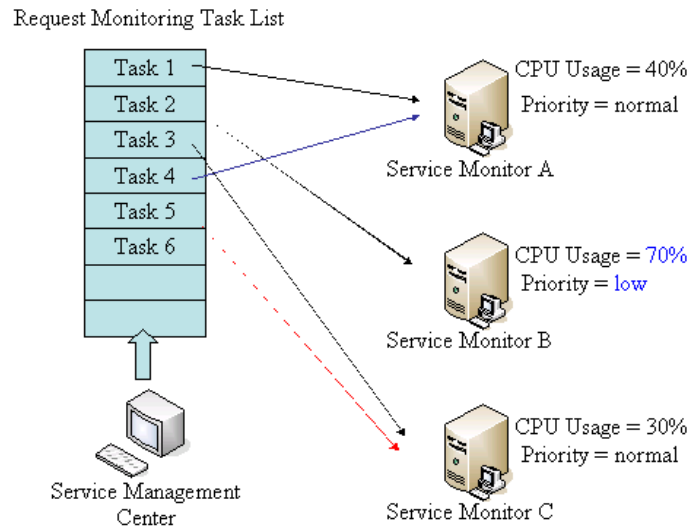


Fig. 3. The example of assigning jobs by SMC.

low-priority service monitor. As shown in Fig. 3, it is assumed that jobs 1 through 3 are assigned to Service Monitor A, Service Monitor B, and Service Monitor C. And jobs 4 through 6 are ready to be assigned. While Because CPU usage of the service monitor B reaches 70%, the job 4 is assigned to service monitor A and job 5 is assigned to service monitor C.

In addition, SMC also considers the multi-policies situation through an *AssignValue* table. In Table 1, *AssignValue* is defined by a method of exhaustion of the all combinations of the set $\{\{\text{CPU Usage, Memory Usage, Monitor Task Number}\}, \{\text{high, normal, low}\}\}$. CPU usage, memory usage, and monitor task number are factors considered for monitoring jobs assignment. And high, normal, and low are the level of these factors. And the assumption is made that the high level of CPU Usage degrades service monitors more than high level of Memory Usage, and high level of Memory Usage degrades service monitors more than high level of Monitor Task Number. There would be 27 possible combinations in Table 1. The design of *AssignValue* has two stages. In the first stage, high level, normal level and low level are assigned by a value, 3, 2 and 1, respectively. And *LevelValue* is used to group combinations of level of factors. *LevelValue* is calculated by summing up each value of factors. For example, if all the level of three factors is low, the *LevelValue* would be 3 in Table 1. In the second stage, *AssignValue* is obviously assigned 27 to combination whose *LevelValue* is 9. However, if *LevelValue* is equal, *AssignValue* is assigned incrementally according to the assumption. For an instance, *LevelValue* of the combination of $\{\{\text{CPU Usage, high}\}, \{\text{Memory Usage, high}\}, \{\text{Monitor Task Number, normal}\}\}$, $\{\{\text{CPU Usage, high}\}, \{\text{Memory Usage, normal}\}, \{\text{Monitor Task Number, high}\}\}$ and $\{\{\text{CPU Usage, normal}\}, \{\text{Memory Usage, high}\}, \{\text{Monitor Task Number, high}\}\}$ are both 8. Because CPU Usage of combination $\{\{\text{CPU Usage, normal}\}, \{\text{Memory Usage, high}\}, \{\text{Monitor Task Number, high}\}\}$ is normal, and the others are high. So *AssignValue* is assign 24 to the combination. And then Memory Usage of $\{\{\text{CPU Usage, high}\}, \{\text{Memory Usage, normal}\}, \{\text{Monitor Task Number, high}\}\}$ is normal, and

Memory Usage of {{CPU Usage, high}, {Memory Usage, high}, {Monitor Task Number, normal}} is high, so *AssignValue* is assigned 25 to the combination {{CPU Usage, high}, {Memory Usage, normal}, {Monitor Task Number, high}}. Thus, *AssignValue* is assigned 26 to {{CPU Usage, high}, {Memory Usage, high}, {Monitor Task Number, normal}}. And so forth, *AssignValue* of the other combinations of level of factor are assigned in Table 1.

Table 1. *AssignValue* table.

CPU Usage	○	○	○	⊙	○	○	○	⊙	⊙	●	○	○	⊙	⊙	⊙	●	●	○	⊙	⊙	●	●	●	⊙	●	●	●	
Memory Usage	○	○	⊙	○	○	⊙	●	○	⊙	○	⊙	●	○	⊙	●	○	⊙	●	○	⊙	●	●	⊙	●	●	●	●	
Monitor Task Number	○	⊙	○	○	●	⊙	○	⊙	○	○	●	⊙	●	⊙	○	⊙	○	○	●	●	⊙	●	○	○	●	●	⊙	●
<i>LevelValue</i>	9	8	8	8	7	7	7	7	7	7	6	6	6	6	6	6	6	5	5	5	5	5	5	5	4	4	4	3
<i>AssignValue</i>	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

● means that priority is high ⊙ means that priority is normal ○ means that priority is low

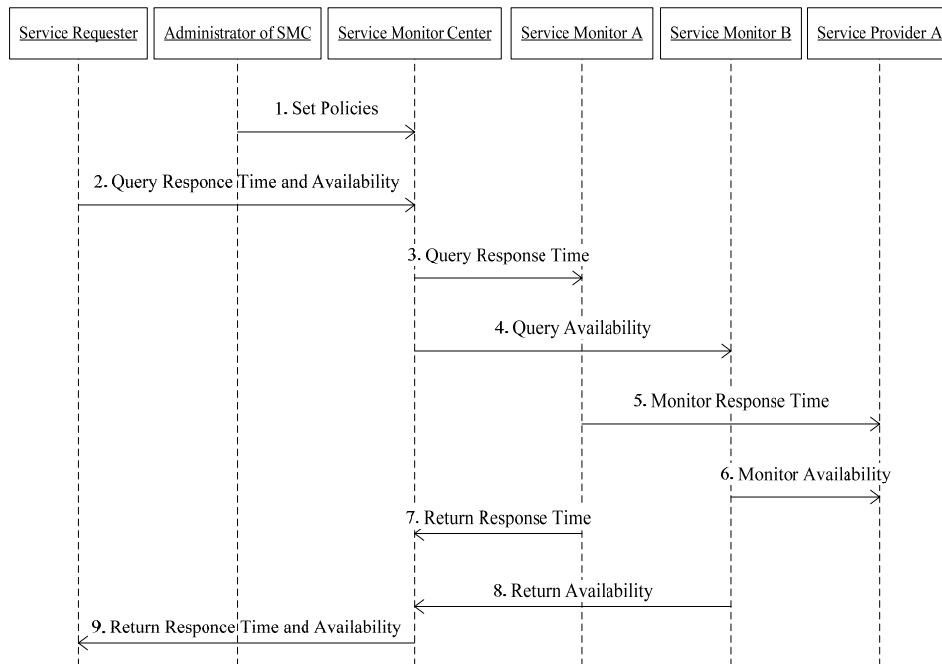


Fig. 4. Sequence diagram of monitoring management by SMC.

The sequence of monitoring management by SMC is shown in Fig. 4. Firstly, the administrator of SMC would set policies to manage service monitors. And the second step, a query for QoS parameters, such as availability or response time, was sent to SMC from the service requester. Then SMC would assign jobs to the corresponding service

monitors according to the policies in the third and fourth steps. Thus, from the fifth step to eighth step, the service monitors would monitor the service provider and return the queried QoS parameters to SMC. In case of Fig. 4, Service monitor A was responsible to monitor response time, and service monitor B was responsible to monitor availability. And then they both returned the monitoring result to SMC. In the ninth step, SMC would return the collected QoS parameters to the service requester.

3.2 Status for Service Monitors

Using service monitors efficiently to monitor service is one of the main functions of SMC, and the function is to determine the status of service monitors. The status for service monitors, such as the CPU utilization, memory usage, and number of monitoring jobs for a service monitor, can be monitored by SMC.

The number of monitoring jobs for a service monitor can be divided into two parts, that is, one is the real-time current number, and another is the real-time maximum number. This type will be defined by the deadline of Service Level Agreement (SLA), and service providers and service requesters have to make a deal before the deadline of SLA. There is another long-term number of monitoring jobs, including the LongTermMaxNumber which is the maximum number of monitoring jobs to be executed, and the LongTermCurrentNumber which is the maximum number of jobs being executed. Through the three above factors, SMC will determine whether the service monitor is busy. If the service monitor is busy now, the monitoring job won't be assigned to service monitor next turn.

4. IMPLEMENTATION OF POLICY-BASED MONITORING SYSTEMS FOR WEB SERVICES

In this paper, the implementation is to build the architecture which aims to monitor status of service monitors and customize related policies to assign monitoring jobs. In Fig. 5, the panel of SM Management reveals status of all the service monitors and the structure arranged as a tree. And the descriptions for each row of SM Status Table in order are: index of SM, CPU utilization, memory usage, LongTermMonitorMaxNumber, RealTimeMaxNumber, LongTermMonitorCurrentNumber and RealTimeCurrentNumber. Besides, this implementation shows CPU utilization, memory usage, and Monitoring-Load by chart. In Fig. 6, the administrator can customize the policies to manage service monitors and determine how to assign monitoring jobs according to the policies. The part fenced in by purple circle can be used to add or edit contents of a new or existed policy. After submitting a new policy, the policy you edited will be appear in the part fenced in by blue circle and the Policy Storage area. Also, the administrators can select the policy they want in Policy Storage, and then the selected policies will appear in the Policy Cache (The part fenced in by red circle). Furthermore, the content of the policy will be presented in XML form.

In Fig. 8, the administrator select the Policy 3, which means if the memory usage of service monitors exceeds 70%, the priority of the service monitor is set as low. We assume QoWSME as a service requester, and it requests SMC to assign jobs to service

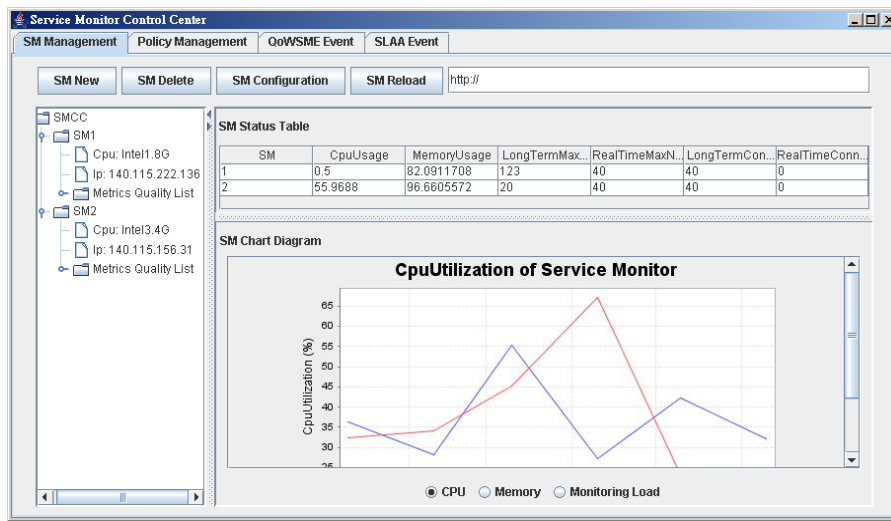


Fig. 5. CPU Usage of service monitors.

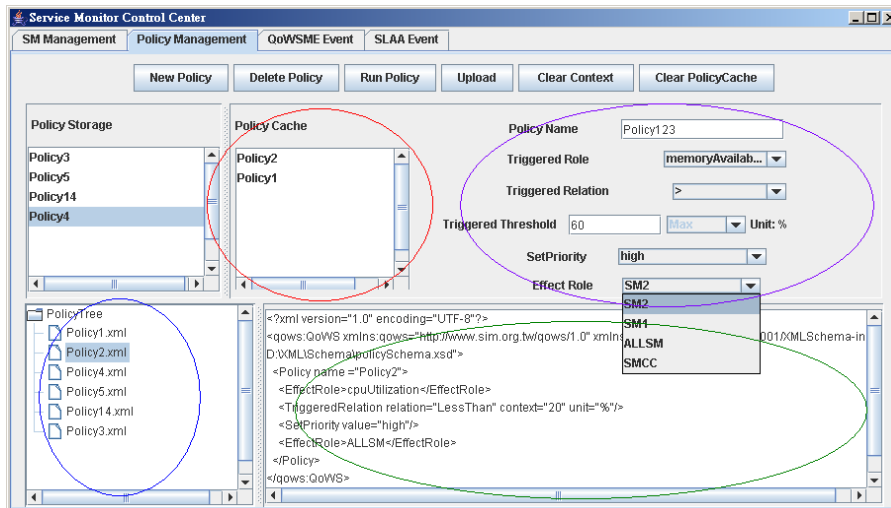


Fig. 6. The SMC management interface for administrators.

monitors for its service. In Fig. 9, each record in Request List includes the TaskID, the service name be monitored, its domain and the time needed to be monitored.

In Fig. 10, when a monitoring job is assigned, the status of service monitor will be examined with the policies. When the status exceeds the defined threshold, for example, the policy 3 restricts the memory usage not to exceed 70%; otherwise the priority of service monitor will be set as low. Certainly if policy and monitoring jobs are getting more and more, the AssignValue of service monitor may arrive 8 quickly. So the Task 69 in Fig. 8 is first assign to service monitor 2 not to service monitor 1. Furthermore, SMC has to response the services it monitored when service requesters request.

In order to verify the effectiveness of the SMC, the experiment of using a specific management policy has been made. The policy of memory usage will be set, and then memory usage is monitored whether the actual usage of system memory exceeds the threshold defined in policies. In Fig. 7, the Policy 3 is selected, which means if the memory usage of service monitors exceeds 70%, the priority of the service monitor is set as low. In Fig. 8, it is observed that when the memory usage of service monitor 1 exceeds

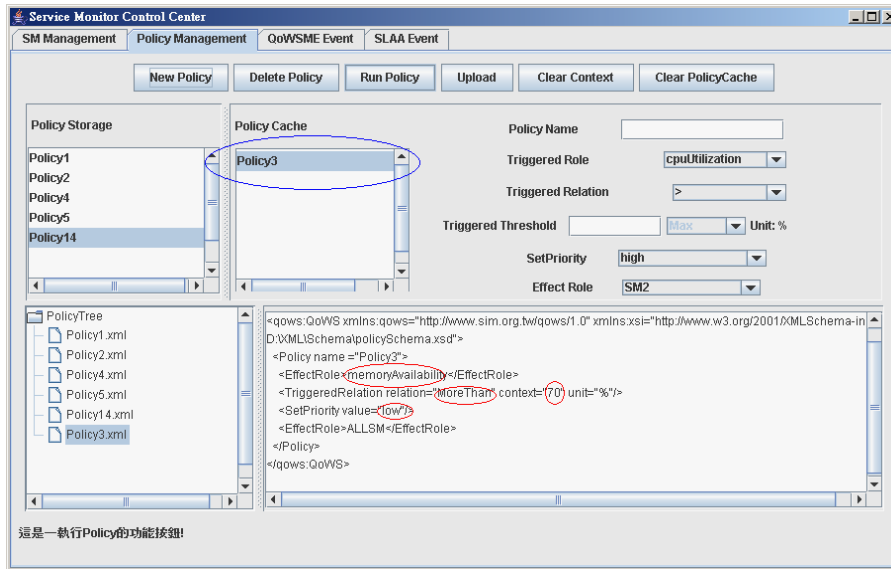


Fig. 7. The SMC policy editor for administrators.

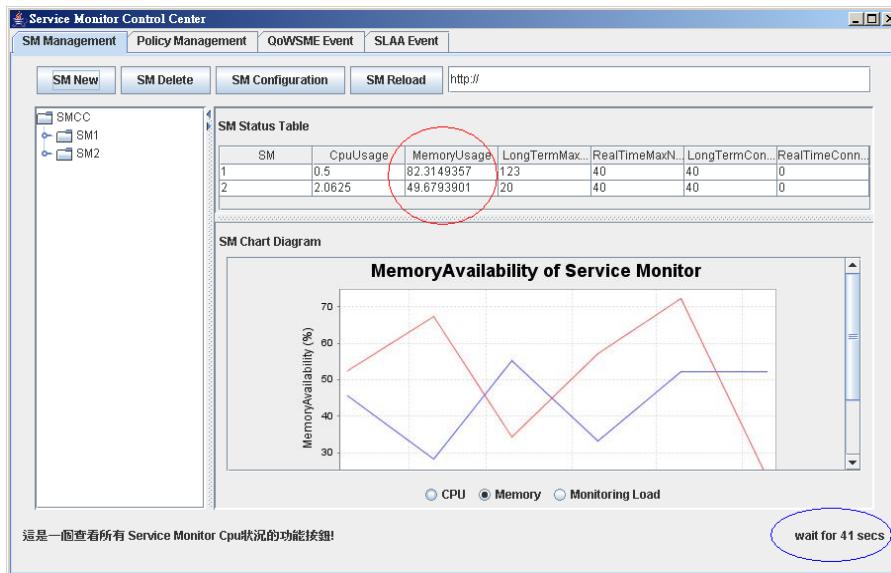


Fig. 8. The memory usage of service monitors.

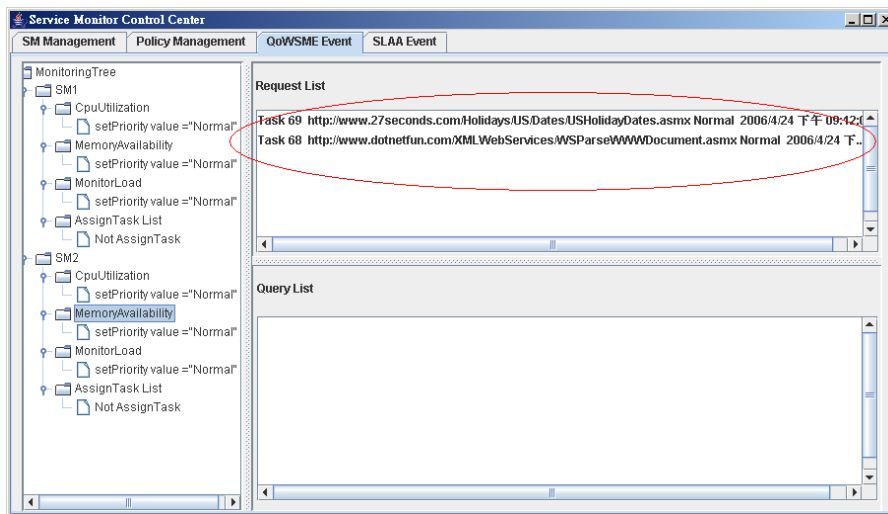


Fig. 9. The monitoring event of service requesters.

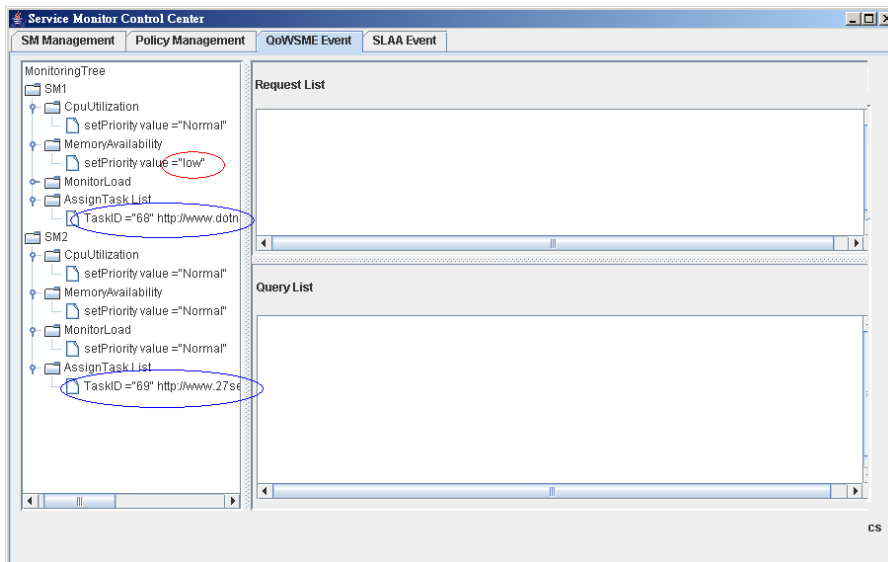


Fig. 10. The job assignment according to the policy.

the defined threshold, the priority of service monitor will be set as low and the usage of system memory will go down. The monitoring job is distributed to service monitor 2 because the priority of service monitor is still normal. Thus, it is found that the memory usage of service monitor 1 is bound within 70%. Otherwise, the memory usage of service monitor 1 will not go down even though the memory usage of service monitor 1 exceeds 70%. If the memory usage of service monitor 1 reaches 100%, performance of service monitor 1 will degrade greatly. Thus, the SMC has prevented this situation from carried

out. It shows that the memory usage of both service monitors can be utilized efficiently by the SMC.

5. CONCLUSIONS

In this paper, we implemented a policy-based monitoring system for Web Services. This system can be used to handle the monitoring jobs and the QoS service monitors monitored. Furthermore, the policy-based management is conducted to improve the performance of service monitors. Through the customized policies, the monitoring jobs can be assigned to service monitors more efficiently. And the loading of Service Monitors is reduced and distributed by predefined CPU-based and Memory-based policies. This paper provided administrators the interfaces to manage status of service monitors and monitoring jobs. Furthermore, we will examine the efficiency and effectiveness of our system using the benchmarking devices, such as SmartBits.

ACKNOWLEDGEMENT

The authors would like to deeply thank the anonymous referees for their constructive suggestions.

REFERENCES

1. H. I. Liu and M. N. Yang, "QoL guaranteed adaptation and personalization in e-learning systems," *IEEE Transactions on Educations*, Vol. 48, 2005, pp. 676-687.
2. "Simple object access protocol (SOAP) 1.1," <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>, 2000.
3. "Web services description language (WSDL) 1.1," <http://www.w3.org/TR/wsdl>.
4. I. Y. Chen and C. C. Huang, "A service-oriented agent architecture to support telecardiology services on demand," *Journal of Medical and Biological Engineering*, Vol. 25, 2005, pp. 73-79.
5. L. D. Chou, K. C. Shen, K. C. Tang, and C. C. Kao, "Implementation of mobile-agent-based network management systems for national broadband experimental networks in Taiwan," *Lecture Notes in Computer Science*, Vol. 2744, 2003, pp. 280-289.
6. Y. J. Chang, T. Y. Wang, Y. S. Chang, and L. D. Chou, "A qualitative study of web 2.0 services for NGP – A case study of employment services for individuals with mental illness," *Journal of Advanced Engineering*, Vol. 3, 2008, pp. 161-170.
7. C. N. Hsu, C. H. Chang, C. H. Hsieh, J. J. Lu, and C. C. Chang, "Reconfigurable web wrapper agents for biological information integration," *Journal of the American Society for Information Science and Technology*, Vol. 56, 2005, pp. 505-517.
8. C. L. Fang, D. R. Liang, F. Y. Lin, and C. C. Lin, "Fault tolerant web services," *Journal of Systems Architecture*, Vol. 53, 2007, pp. 21-38.
9. L. D. Chou, C. H. Lin, Y. C. Lin, and C. M. Huang, "Seamless handover in WLAN and cellular networks through intelligent agents," *Journal of Information Science and Engineering*, Vol. 23, 2007, pp. 1087-1101.

10. L. D. Chou, J. M. Chen, H. S. Kao, S. F. Wu, and W. Lai, "Seamless streaming media for heterogeneous mobile networks" *ACM/Springer Mobile Networks and Applications*, Vol. 11, 2006, pp. 873-887.
11. H. Y. Shin, J. L. C. Wu, and Y. C. Hsu, "The study of QoS guarantee in the optical burst switching internet backbone," *Optical Switching and Networking*, Vol. 3, 2006, pp. 50-63.
12. R. T. Sheu and J. L. C. Wu, "A prioritized resource reservation scheme with QoS-violation assessment," *Journal of Information Science and Engineering*, Vol. 21, 2005, pp. 23-37.
13. L. D. Chou, C. C. Chen, Y. S. Wang, and P. C. Tseng, "Design of an SLA-based service monitor system using agents," *System and Information Sciences Notes*, Vol. 1, 2007, pp. 365-368.
14. X. T. Nguyen and R. Kowalczyk, "Enabling agent-based management of web services with WS2JADE," in *Proceedings of the 5th International Conference on Quality Software*, 2005, pp. 407-412.
15. Y. D. Lin, C. M. Tien, S. C. Tsao, R. H. Feng, and Y. C. Lai, "Multiple-resource request scheduling for differentiated QoS at website gateway," *Computer Communications*, Vol. 31, 2008, pp. 1993-2004.
16. G. Yeom and D. Min, "Design and implementation of web services QoS broker," in *Proceedings of the International Conference on Next Generation Web Services Practices*, 2005, pp. 454-455.
17. "Web Services Monitor," <http://sitescope.tellurian.net/SiteScope/docs/WebServiceMonitor.htm>.
18. "An introduction to task dispatch manager," <http://www.vpsoft.com/tdm.htm>.
19. L. Cherkasova and M. Karlsson, "Scalable web server cluster design with workload-aware request distribution strategy WARD," in *Proceedings of the 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, 2001, pp. 0212.
20. Y. W. Chen and S. H. Hu, "Study of the traffic scheduler by using correlation heuristics," *IEICE Transactions on Communications*, Vol. E87-B, 2004, pp. 2273-2280.
21. "QoS for web services: Requirements and possible approaches," <http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/>.
22. H. Y. Shin, J. L. C. Wu, and H. H. Liu, "Performance analysis of dynamic multichannel scheme with channel de-allocation in integrated wireless networks," *IEICE Transactions on Fundamentals*, Vol. E87-A, 2004, pp. 1681-1691.
23. J. H. Chen and S. T. Sheu, "Distributed multichannel MAC protocol for IEEE 802.11 ad hoc wireless LANs," *Computer Communications*, Vol. 28, 2005, pp. 1000-1013.
24. S. D. Wu, C. W. Chen, and X. W. Lu, "Adaptive context-aware web services monitoring: QoS guaranteed web services monitoring," in *Proceedings of the 16th Workshop on Object-Oriented Technology and Applications*, 2005.
25. J. Lee, Y. C. Wang, C. L. Wu, S. J. Lee, S. P. Ma, and W. Y. Deng, "A possibilistic petri-nets-based service matchmaker for multi-agent systems," *International Journal of Fuzzy Systems*, Vol. 7, 2005, pp. 199-213.
26. J. M. Chen, S. P. Ho, Y. C. Lin, and L. D. Chou, "QoS routing with link stability in mobile ad hoc networks," *Lecture Notes in Computer Science*, Vol. 3824, 2005, pp. 683-692.

27. J. D. Moffett and M. S. Sloman, "Policy hierarchies for distributed systems management," *Special Issue on Network Management, IEEE Journal of Selected Areas in Communications*, Vol. 11, 1993, pp. 1404-1414.
28. H. H. Liu, K. J. Zjang, Y. J. Chang, T. Y. Wang, and L. D. Chou, "Testbed of web 2.0 services for individuals with mental illness," in *Proceedings of IEEE TENCON*, 2007, pp. 1-4.
29. L. D. Chou, H. J. Hsieh, and J. M. Chen, "Multicast with QoS support in heterogeneous wireless networks," *Lecture Notes in Computer Science*, Vol. 3207, 2004, pp. 581-590.
30. Y. W. Chen, Y. L. Lee, and I. H. Peng, "Connection control by virtual admission in wireless LAN environment," *Lecture Notes in Computer Science*, Vol. 4138, 2006, pp. 14-25.
31. Y. M. Wang, "Local cluster first load sharing policy for heterogeneous clusters," *Journal of Information Science and Engineering*, Vol. 23, 2007, pp.497-510.



Li-Der Chou (周立德) received a B.S. in Electronic Engineering from National Taiwan Ocean University, Taiwan, R.O.C. in 1988, a M.S. and a Ph.D. in Electronic Engineering from National Taiwan University of Science and Technology, Taiwan, R.O.C. in 1991 and in 1995, respectively. He has worked as an associate professor with Jin-Wen College of Business and Technology, Chung-Hua Polytechnic Institute, and Central Police University, respectively. From 1997 to 1999, he acted as the director of the Multimedia Communications and Services Division, Computer Center, National Central University, Taiwan, R.O.C.

From 1997 to 2005, he was an associate professor with the Department of Computer Science and Information Engineering, National Central University, Taiwan, R.O.C. Currently he is a professor with the same department. He has published more than 100 journal and conference papers in the area of computer networks. In addition, he owns one United State and eight Taiwan invention patents. Moreover, he won the Award of Excellence on 2005 IPv6 International Appli-Contest 2005 held in Japan. His current research interests include network management and planning, broadband wireless networks, sensor networks, and Internet applications and services.



Chien-Chi Chen (陳建麒) received his B.S. degree in Computer Science from Tamkang University, Taiwan, R.O.C. in 2004, M.S. degree in Computer Science and Information Engineering from National Central University, Taiwan, R.O.C. in 2006. His research interest includes XML, SOA and web services technologies.



Po-Chia Tseng (曾柏嘉) received his B.S. and M.S. degrees in Computer Science and Information Engineering from National Central University, Taiwan, in 2006 and 2008, respectively. His research interests include embedded system, networks management, and web service.