Dynamic Agent Based Monitoring Mechanism for Web Services

WEN Junhao, ZHAO Li, HE Pan

Abstract

With more and more web services appearing in the Internet, quality of service (QoS) has aroused attentions from many researchers. Only if QoS is assured, services can be widely used and the needs of customer can be preferably met. QoS has been researched for a long time and there are many approaches proposed to monitor web services, but most of them need to involve with service provider. For example, aspect oriented programming (AOP) based monitoring mechanism needs to insert codes into service source codes at service runtime; Simple Network Management Protocol (SNMP) based monitoring mechanism requires service provider to support the SNMP protocol. To solve these issues, this paper proposes a dynamic agent based mechanism to monitor web services, which is then applied to service oriented architecture (SOA). Through extending register functions in SOA, monitoring and management system (MMS) could get service information to generate service-related agent that monitor the service states at runtime. According to SLA, MMS evaluates quality of service with data reported by agent. To implement our monitoring architecture, Java Management Extensions (JMX) is referenced. The monitoring mechanism proposed in this paper not only doesn’t need additional information from service provider, but also doesn’t increase loading burden on service provider. Comparing with others mechanism of web services monitoring, ours provides more flexibility and scalability.

Keywords: Dynamic Agent, Web Services, SLA, QoS

1. Introduction

Service oriented architecture (SOA) makes different services integrated by a standard interface, which improved usability of services, advanced interoperability and loose-coupling between services. SOA consists of service register (SR), service provider (SP) and service customer. At present, the main technology for implementing SOA is Web services, which has applications that contain themselves and are modularization, builds basic protocols including: Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI).

As a core part of SOA, service need not only meet the needs of customer, but also should have guarantee of quality of service (QoS). With more and more web services appearing in the Internet, QoS is a determinant factor to select services and affects usage of services. Without QoS assurance, creditability and utilization of services will be reduced. Therefore, not only services customer should pay attention to quality of service, but also services provider should do.

To assure QoS, monitoring web services is an important mechanism. Through obtaining services states at runtime, calculating QoS attributes value and comparing with specifications as SLA, the QoS can be evaluated, exceptional behaviors of services can be found, the monitored results for web services could help to improve QoS[10] and benefit customers to select proper services. Monitoring web services could be executed by service providers or independent third parties. In many researches, a lot of mechanisms to monitor web services are proposed, for example AOP based monitoring mechanism and SNMP based monitoring mechanism etc. However, most of them need additional information from service provider, install component into service provider or plug some codes for monitoring into service source codes, these methods take loading on service provider and make monitoring limited, meanwhile it violate the loose-coupling principle of SOA. To solve these issues,
we propose a dynamic agent based monitoring mechanism for web services. Dynamic agent means that the agent to monitor web services is dynamically generated by MMS according to service info that is submitted by service provider, the monitoring implementation need not to pre-know services running details, especially additional support of service provider. SNMP [11] gives us much help to design the dynamic agent based monitoring mechanism for web services. According to service info provided by service register, MMS generates an agent for monitoring services states, evaluates QoS through calculating QoS attributes value according to reports of agent and comparing with SLA; then obtains the “health” status of service. If the quality of service deviates from SLA, MMS sends messages about QoS deviation to register. Here, service register functions are extended in order to deliver service info for MMS.

The rest of this paper are organized as follows: Section 2 reviews related work, section 3 presents our monitoring framework, section 4 describes the agent design, section 5 discusses the implementation of our monitoring architecture, section 6 concludes this paper.

2. Related work

Most of methods to monitor web services evaluate QoS by capturing parameters related to QoS attributes at service runtime, calculating QoS attributes value and comparing evaluation results with some specifications such as SLA. According to the evaluated results, QoS violator can be measured for web services. Here, QoS attributes refer to availability, reliability, response time, performance and security of web services etc.

Natee. Artaiam and Twittie.Senviongse.G. Eason [1] propose a mechanism enhancing service-side QoS monitoring for web services, which emphases to improve monitoring by adding more QoS attributes on service side. In [2], proposes a trusted web services assessment model based on six dimensional QoS Framework and end-to-end monitoring, the monitoring method enforces service providers to support SNMP protocol. Paper [3] presents a novel mechanism, which is AOP based trustable SLA compliance monitoring for web services, implemented with plugged monitor logic codes which have been pre-defined in SLA manager into the service source codes at run-time of service. Paper [4] discusses a composite event-based monitoring for semantic web services, which defines the monitored events in semantic web service, but it isn’t flexible when new events appear. In [5, 6], they aim at monitoring composition process of web services, and are based model-driven monitoring. Paper [7] designs a monitoring model with mobility agent to extend SNMP protocol, but the mobility agent can’t work unless service provider have some special supports and this paper doesn’t define specific interaction between service and mobility agent. Paper [8] proposes a novel mechanism for monitoring web services which is different with the most researches. This approach implements QoS monitoring for web services by time series forecasting, tend is beneficial to reducing loading burden for monitoring. But it lacks reliability and real-time.

We propose a dynamic agent based monitoring mechanism for web services. In our monitoring framework, we design the monitoring management system (MMS) and agent, extend register functions. The agent is dynamically generated by MMS according to registered service information, and run at service side, with the lightweight feature agent would hardly affect service running. Our monitor model is similar to SNMP protocol, using the agent and manager architecture. Agent is responsible to send reports about services states at runtime for MMS, and MMS evaluates QoS according to the reports. In order to dynamically create agent, we extend service register functions, the extension is not only for creating agent, but also for communication between MMS and register. In this paper, the monitoring process is completed through collaboration between register, MMS and agent.

3. Monitoring mechanism framework for web services

This section presents the design of monitoring framework for web services, which includes monitoring model and process. The monitoring framework is depicted as Figure 1.
3.1. Monitoring model

The SOA environment includes three parts: service provider, service customer and service register. Service register only provides the registration and query of services for service provider and service customers, which doesn’t refer to monitoring web services in most researches. Meanwhile, as a catalogue for services, register is not fit to provide too many functions. Therefore, to avoid increasing more burdens at service register, we only add a few new functions into register, which could make register communicate with MMS. In the monitoring framework, the MMS is the most important part, functions of which mainly include communication with service register, agent management and QoS evaluation, except them which is still responsibility for initialization and analyzing reports generated by agent. Agent running at service side provides the reports related to parameters of QoS attributes, which are captured from SOAP messages at service runtime. The monitoring model is shown in Figure 2. In our monitoring mechanism, we don’t consider the objective factors affected QoS such as network.

3.2. Monitoring process

In the design of this paper, monitoring process begins with that service provider registers his services with SLA, service register sends service information to MMS after receiving the service, MMS analyzes it and completes initialization work, then extracts parameters related to QoS attributes
to generate agent for this service, agent is sent to service provider and runs to monitor service states at runtime, the agent monitoring includes capturing the interaction messages between services and customers and recording the executed status of services. The agent periodically sends reports with monitoring data to MMS. MMS analyzes the reports and evaluates QoS by calculating the QoS attributes value and comparing with SLA. If the evaluated result appears deviation from SLA, MMS will notify service register with messages about QoS deviation, and register modifies service information according to the messages. To real-time monitor web services, MMS could send request to agent whenever. Figure 3 shows the monitoring process.

![Monitoring Process](image_url)

**Figure 3. Monitoring Process**

### 4. Design of dynamic agent

Dynamic agent proposed in this paper is dynamically generated by MMS according to service info, and will be canceled by MMS if QoS seriously deviates from SLA or monitoring service is failure. The agent is running at service provider to monitor service states at runtime, records parameters related to QoS attributes through analyzing the captured SOAP messages and generates reports. In order preventing to increase loading burden on service provider for monitoring, we employ the lightweight method to design agent. All functions about calculation are implemented by MMS; agent’s functions are monitoring service states, sending reports to MMS and responding to request from MMS.

During a monitoring period, agent is responsible to capture the SOAP messages at interaction between service provider and customer, analyze the messages and record messages id, time and other parameters related QoS attributes. We make an assumption that a response message responds to a request, request message id is the same as response id and each request message has a different id. To assure monitoring service is effective, agent needs to continually read service states; if service is failure, agent sends message to inform MMS about it. After completed monitoring for a period, agent generates a report with XML format and sends it to MMS, which clears the saved record on service provider and start to monitor for next period after received response from MMS. In order that MMS can real-time get the data that is monitored by agent, agent could support to respond to request from MMS. Figure 4 shows the monitoring architecture of agent.
To make agent synchronized with MMS, we design a clock between MMS and agent, which is set by MMS. When generating agent, MMS allocates a clock for the agent. Agent sends reports according to the clock, then when enters the next period for monitoring agent will reset the clock.

5. Implementation

To implement our proposed monitoring mechanism, the Java Management Extensions (JMX) is involved, which defines the architecture, the design patterns, the APIs and the services for application and network management and monitoring in the Java programming language [12]. For these characters of JMX we choose it to implement our architecture, but as it can’t fully meet our proposed design we modify it and choose JAVA programming language with independent on platform and My Eclipse development platform. This section mainly presents our monitoring mechanism implementation.

5.1. Mapping between JMX and monitoring architecture

JMX consists of instrument level, agent level, distributed services level and additional management protocol APIs [12]. Only the first three levels relate to the proposed monitoring mechanism. Instrument level involves managed programmer that is wrapped by using Managed Bean (MBean) registered in MBean server, which is mapped to the services provider in our design; agent level includes agents which can manage resource and communicate with JMX manager, which is similar to our dynamic agent; in JMX architecture, the JMX Manager exists in distributed services level, which has a role related to MMS. Figure 5 shows the mapping between JMX and Monitoring Architecture.
5.2. Integrating JMX APIs into MMS

In the JMX architecture, manageable resource exists at instrument level and needs to register as MBean which is a Java object to implement a specific interface [12]. JMX agent also registers as an MBean to manage resource. Between manageable resource and JMX agent, notification and timer has been implemented. JMX manager communicates with JMX agent to manage resources. For these, JMX notification model might help to implement communication between register and MMS and agent and MMS, the timer mechanism is used to design the clock that is used when agent monitors service. In JMX, the notification and timer has been achieved by the way of APIs, we modify these APIs and integrate them in MMS. Due to the space limitation, the details cannot be displayed in this section.

5.3. Implementation of MMS

Except Notification and Timer, Monitor and Calculator in the design are important to implement MMS. Monitor is to dynamically generate agent and Calculator for evaluating QoS. There are no adaptors and connectors between agent and MMS. Agent is managed by MMS, which is generated by MMS according to service info by dynamically compiling pre-defining components as a part of Monitor. After completing agent compilation, the executable file is sent to service provider. To run agent, service provider only needs to install Java virtual machine. All calculation-related functions exist in MMS to light weight agent. Actually, Monitor, defined as a public interface in MMS and implemented according to specific services, is a process that an agent captures and analyzes SOAP messages at runtime of web services. In the MMS design, all functions implementation is inherited from an interface called Management, according to specific requirements the related interfaces are implemented. As the space limitation, Figure 6 shows the relationship of a part of important interfaces and classes defined in MMS. In the Figure 6, the solid line represents extension relationship, the dotted line expresses implementation relationship. Interface Monitor, Calculator, Notification and Timer extend from Management. Monitor defines all related to monitoring functions which are used to generate agent and implement monitoring by agent at service runtime; Calculator defines operators related to calculating QoS attributes value; Notification defines communication policy between service register and MMS and agent; Timer abstracts the clock that defines the monitoring period of agent. In the figure 6, all leaf nodes are class, which implement the parent interfaces to achieve specific functions.

![Figure 6. Relationship of the Important Interfaces and Classes in MMS](image-url)
5.4. Comparison of monitoring mechanism for web services

Our proposed mechanism could evaluate QoS attributes such as reliability, availability, performance and response time etc. Through extending the MMS, it is possible to implement more evaluation about QoS to advance monitoring web services. The agent is dynamically generated by MMS using lightweight way. Comparing with other methods, ours provides more flexible and hardly increase loading burden on service provider. The comparison of monitoring methods for web services is shown in the Table 1. Notes the symbol “˅” marks support.

<table>
<thead>
<tr>
<th>Monitoring approach</th>
<th>QoS monitoring</th>
<th>Flexibility</th>
<th>Scalability</th>
<th>Lightweight feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based AOP monitoring mechanism</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Based SNMP monitoring mechanism</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Our monitoring mechanism</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

6. Conclusion

With more and more web services appearing in the Internet, it is more difficulty to select proper services for customers. QoS is an important indicator to measure and select services, meanwhile, only if QoS is assured, the services utilization can be advanced as much as possible. The main method, with monitoring web services to obtain the actual QoS attributes value, is implemented through capturing parameters related to QoS attributes at runtime of web services and evaluating quality of service according to specifications such as SLA. Many researches about monitoring web services have been done, and proposed a lot of mechanism, but a few are about dynamically generated agent. We extend the register functions to communicate with MMS. Through capturing the SOAP messages to fetch parameters related to QoS attributes, agent running at service provider side could monitor the service states and periodically send reports to MMS. MMS implement to QoS evaluation according to SLA. If evaluated results deviate from SLA, MMS will send messages about QoS deviation to register, the register modify service information to meet customers’ needs. Finally, we utilize JMX to implement the monitoring architecture. Our proposed mechanism doesn’t need special support provided by service provider, not to plug monitoring codes into service sources codes and hardly increases loading burden on service providers because of the lightweight feature of agent. In the future, we plan to design distributed MMS to advance efficiency of monitoring web services.

Acknowledgment

This work is supported by Natural Science Foundation Project of CQ CSTC2010BB2244 and supported by the Fundamental Research Funds for the Central Universities Project No.CDJSX11181161 and supported by the Fundamental Research Funds for the Central Universities Project No. CDJZR10090001.

References


