Journal of Southeast University (English Edition)   Vol. 23, No. 3, pp. 413 – 417   Sept. 2007   ISSN 1003—7985

Business performance monitoring based on web service proxy

Chen Zaiben[1,2]   Xing Chunxiao[2]   Yang Jijiang[2]   Hu Qingcheng[1,2]

([1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

([2]Research Institute of Information Technology, Tsinghua University, Beijing 100084, China)

Abstract: To provide efficient monitoring of web service-based business processes, a web service proxy (WS-proxy) is developed to monitor business activities by monitoring the enactment of services. WS-proxy is deployed as an intermediary between internal business processes and external service providers, and it provides a single point of service access with the functions of message routing and content inspection. By using an XPath engine named WS-filter, performance indicators can be generated from service messages for assessing business performance. In the experiments, the feasibility of WS-proxy is verified and it achieves good performance in the monitoring work. The latency introduced by WS-proxy is only about 15% of the overall latency while extracting performance indicators just consumes less than one third of the processing time.

Key words: performance; process; monitoring; web service; proxy

In recent years, more and more organizations are deploying web service applications to provide standardized, programmatic application functionality over the Internet[1], and web services have been widely adopted as a mechanism for government and enterprise to construct their business processes. The invocations of web services are business activities, and how to acquire performance indicators from these activities for assessing business performance is what we try to solve. In this paper, we introduce an approach to monitor web services-based business process performance by using web service proxy (WS-proxy). WS-proxy insulates the internal business processes from external service providers, and provides a single point of service access for both business processes and external customers. Besides, WS-proxy can also extract target XML elements from the web service messages for generating performance indicators. To achieve efficient monitoring, WS-proxy uses an optimized XPath engine developed based on YFilter[2] for content-based message routing and data extracting.

1 Related Work

A lot of research has been done for monitoring business performance. These works focus on retrieving necessary data for performance evaluation. McGregor et al. developed an approach to monitor web services-based business processes by embedding log service into processes[3] and Fu et al. developed a mechanism for monitoring business performance by monitoring system events[4]. Thomas et al. introduced a semantic approach in which semantic expressiveness is added to the business process for describing business activities and their performance criteria, and agents are utilized for the monitoring work[5]. Our solution is similar to Fu' s event-based scheme in some ways. WS-proxy monitors business performance by monitoring the enactment of web services. Besides, our solution also refers to the web service router gateway[6] and another design of the web service proxy[7].

2 Web Service Proxy

In a common scenario, government and enterprise utilize web services to construct their own business processes, and provide services to external customers. The left side of Fig. 1 shows such a scenario, from which we can see that the service invocations are unorderly distributed and hard to be managed. Therefore, we intend to bring an intermediary node into the web service model to make the web service activities easier to be monitored, as shown in the right side of Fig. 1. We deploy WS-proxy on this intermediary as a service proxy, providing the capability of intercepting messages and extracting data for monitoring purposes. Besides, a central intermediary for message dispatching also makes it easier for service management. Here, we use SOAP/HTTP as the transfer protocol.

Received 2007-05-18.

Foundation item: The National Natural Science Foundation of China (No. 60473078).

Biographies: Chen Zaiben (1982—), male, graduate; Xing Chunxiao (corresponding author), male, doctor, professor, xingcx@ tsinghua. edu. cn.

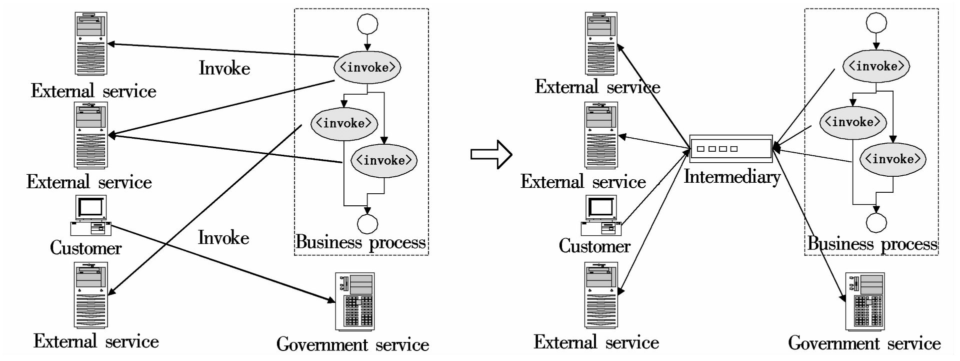

Fig. 1 Service invocations

2.1 WS-proxy features

To some extent, WS-proxy is a combination of web service router and gateway with the capacity of extracting and analyzing message content which is in XML format. It provides the following features:

● Service mapping WS-proxy maintains the service mapping between the existing web services and the virtual services offered by WS-proxy to others. An external service can be imported and made available as a virtual internal service to government systems. Likewise, an internal service provided by the government system can be exported as a virtual service for outside consumption. The virtual services do not exist physically, and WS-proxy just acts as a service proxy for both sides who will use WS-proxy as the service end-point by invoking virtual services.

● Message routing WS-proxy can dispatch service requests/responses to which the real service is physically deployed, since both internal government systems and external customers invoke needed web services through WS-proxy. When receiving a SOAP message, it checks the SOAP header or HTTP header for target URI, and forwards the message to target servers according to the real service URI obtained from the service mapping table.

● Data extracting Our purpose is to extract data from messages for generating performance indicators. WS-proxy uses an XPath engine to process the incoming messages and extracts target XML elements according to the monitoring plan that specifies the target XML elements and their locations in the messages.

Compared with solutions in Refs. [3, 5], WS-proxy does not need to embed any collecting mechanism into the business process. It just monitors the enactment of web services and the monitoring work is transparent to both the internal and the external sides.

2.2 Framework and implementation

Fig. 2 shows the detailed framework of WS-proxy. WS-proxy relays messages sent between the business process and external services, and extracts target XML elements from these messages. WS-proxy is composed of four functional components that work together to carry out the service mapping, message routing and data collecting work.

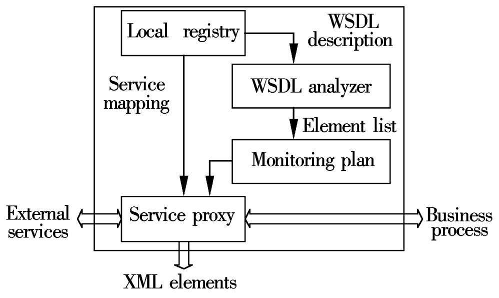

Fig. 2 WS-proxy framework

● Local registry This component maintains a service mapping table. When a new web service is imported, it generates a new virtual service, and adds a new entry to the table indicating the mapping between the existing web service and the virtual service. Both services have the same portType except the service address specified in 〈soap: address location = "URI"〉. All virtual services will have the WS-proxy's address as the service URL, so all service requests will be directed to WS-proxy. To maintain that service interface information, a UDDI component is also incorporated to store all WSDL documents of the existing services and virtual services.

● WSDL analyzer Before making a monitoring plan, the user should know what kind of data can be extracted from which service messages. The WSDL analyzer is used to parse the WSDL documents and find out what XML elements will be contained in which web service operations. From this component, users can get a list of XML elements indexed by a service URI.

● Monitoring plan The monitoring plan is a list of web service operation names along with target elements. From the XML element list acquired from the WSDL analyzer, users can add operations and target elements that will used for generating performance indicators to the monitoring plan. According to the monitoring plan, WS-proxy knows which messages should be checked and which elements should be extracted.

- Service proxy Service proxy is the key component of WS-proxy that takes on the work of message dispatching and inspecting. When a message arrives, service proxy first checks the monitoring plan. If the service operation indicated in the message is on the monitoring plan, service proxy will then further parse the incoming message for target elements, and store the extracted elements in a database. After that, the service proxy sends the message to the real service providers whose URI can be acquired from the service mapping table.

Since all messages are processed in this component, performance issues are particularly important. To avoid message congestion and high latency, we implement an optimized streaming XPath engine named WS-filter based on YFilter for extracting target elements. All target elements' XPath are organized in a deterministic finite automaton (DFA), and evaluated simultaneously when parsing the incoming message. Since from the WSDL document, we can know all elements' exact locations described by XPath, we add stop information generated from WSDL documents to the YFilter's algorithm to reduce unnecessary processing time.

Fig. 3 shows our implementation of the service proxy. We use axis (http://ws. apache. org/axis/) for sending and receiving service requests/responses. The service URI and operation names can be extracted by the WS-filter, and the dispatcher is for relaying messages. If the dispatcher finds that the message is on the monitoring plan, it will send a copy of the message back to the WS-filter to extract target elements.

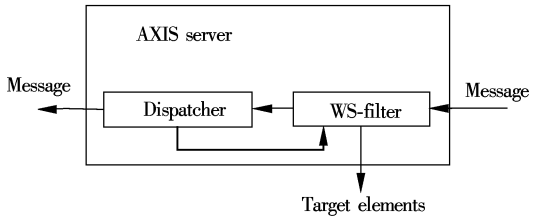

Fig. 3 Service proxy

2.3 Performance monitoring

Using the extracted elements to construct performance indicators for assessing business performance is the purpose of our work. In the following, we use an example to present the process of business performance monitoring using WS-proxy. Assume that an organization provides a business process for customers requesting loans[8]. The process is shown in Fig. 4. If the requested loan amount is high (e. g. $\geqslant 1\ 000$), the request is always sent to the loan approver for review. If the amount is low, a web service, called loan assessor, is invoked to determine the risk. If the assessor determines there is low risk giving the applicant the loan, then it can be approved. Otherwise, the request is sent to the loan approver for a full review.

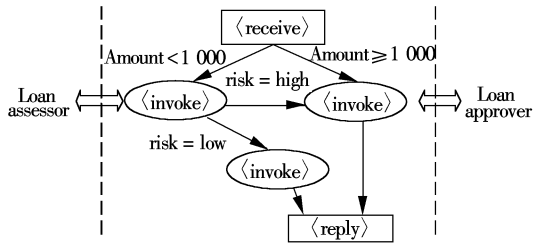

Fig. 4 Loan approval process

Assume that the organization intends to use the average processing time and the amount of a loan with high risk as performance indicators to evaluate this business process. The formula for calculating the performance in the i-th month is as follows:

$$P(i) = \frac{\text{Loan}(i) - \text{Loan}'}{\text{SD}(\text{Loan})} \times 60\% - \frac{\text{Time}(i) - \text{Time}'}{\text{SD}(\text{Time})} \times 40\% \tag{1}$$

where Loan(i) is the amount of a loan with high risk in the i-th month; Loan$'$ is the average of Loan(j), $j = 1, 2, \ldots, i$; Time(i) is the average processing time in the i-th month; Time$'$ is the average of Time(j), $j = 1, 2, \ldots, i$; SD( ) is the standard deviation function. The processing time can be recorded by WS-proxy automatically. To acquire the amount of a loan with high risk, WS-proxy needs to extract the "amount" element contained in the "approve" operation of the request message, which is shown as follows:

```
〈soapenv: Envelope...〉
  〈soapenv: Body〉
    〈approve...〉
      〈name xsi: type = "xsd: string"〉J. Cole〈/name〉
      〈amount xsi: type = "xsd: integer"〉2000〈/amount〉
    〈/approve〉
  〈/soapenv: Body〉
〈/soapenv: Envelope〉
```

Therefore, the "approve" operation and "amount" element are on the monitoring plan. Each time when the loan approver is invoked, the request message will be checked and the content of the "amount" element will be extracted. By applying formula (1), the business performance $P(i)$ can be calculated.

3 Experiments

In this section, we evaluate the feasibility and performance of WS-proxy. WS-proxy is deployed on a PC with Intel PM CPU (1.86 GHz) and 1 GB memory, and another two PCs with AMD Athlon CPU (1.67 GHz) and 1 GB memory act as client and web service server respectively. These three PCs all reside in the same 100 M Ethernet LAN.

3.1 Latency

First of all, we measure the latency introduced by WS-proxy, and the result is shown in Fig. 5. We use Java threads to simulate clients. With the number of threads increasing from 1 to 100, the WS-proxy processing time for a message (denoted as *W*) increases from 141 to 6 886 ms, which is nearly 30% of the overall latency (denoted as *C*) when the client number is 100. By deploying a more powerful server, this percentage can be maintained at around 15%. It can also be concluded that WS-proxy is not the major factor affecting the performance although it indeed brings in some latency.

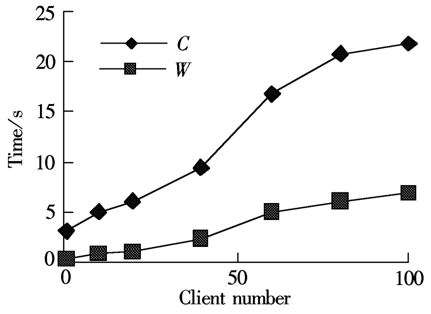

Fig. 5 Latency

3.2 Composition of processing time

The composition of WS-proxy processing time is also studied. Compared with the overall processing time, the time for checking the service mapping and the monitoring plan can be totally ignored. Those times are approximately 0 and 60 ms, respectively when there are 100 clients. From Fig. 6, we can see that extracting target elements (denoted as *E*) consumes less than one third of the processing time, which is about 2 s/message when there are 100 clients. WS-proxy is effective in processing web service messages. Here, *O* represents other processing time introduced by the axis server internally.

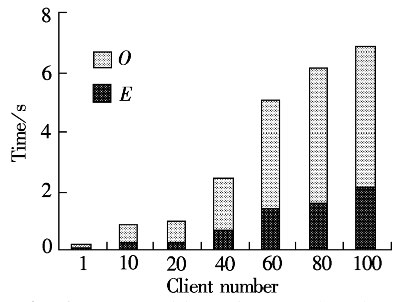

Fig. 6 Composition of processing time

3.3 Performance of extracting algorithm

This experiment is designed to examine purely the WS-filter's performance, so we run the algorithm independently for processing locally pre-stored messages. As shown in Fig. 7, the processing time of a message just increases by about 6% when the number of target elements increases by 880%, which shows that the WS-filter is very efficient when there are multiple target elements, and the latency introduced by extracting target elements is mainly caused by other factors such as network latency and axis processing load.

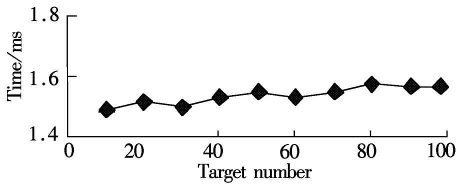

Fig. 7 Performance of WS-filter

4 Conclusion

This paper describes an approach to monitor business performance using WS-proxy that provides service mapping, message routing and data extracting features. WS-proxy provides a single point of service access for government systems and external customers, while incorporating the function of message monitoring for assessing business performance. We discuss the detailed design of WS-proxy and how it can be applied. Finally, experiments are conducted for verifying the feasibility of our solution and WS-proxy achieves good performance.

References

[1] Felber P, Chan C Y, Garofalakis M, et al. Scalable filtering of XML data for web services [J]. *IEEE Internet Computing*, 2003, **7**(1): 49 – 57.

[2] Diao Y, Fischer P, Franklin M J, et al. YFilter: efficient and scalable filtering of XML documents [C]// *Proc of the 18th International Conference on Data Engineering*. Los Alamitos: IEEE Computer Society, 2002: 341 – 345.

[3] McGregor C, Schiefer J. A Web-Service based framework for analyzing and measuring business performance [J]. *Information Systems and e-Business Management*, 2004, **2**(1): 89 – 110.

[4] Fu S S, Chieu T C, Yih J, et al. An intelligent event adaptation mechanism for business performance monitoring [C]// *Proc of the IEEE International Conference on e-Business Engineering*. Los Alamitos: IEEE Computer Society, 2005: 558 – 563.

[5] Thomas M, Redmond R, Yoon V, et al. A semantic ap-

proach to monitor business process performance [J]. *Communications of the ACM*, 2005, **48**(12): 55 – 59.

[6] Liu F, Wang G, Chou W, et al. Target: Two-way web service router gateway [C]//*Proc of the IEEE International Conference on Web Services*. Los Alamitos: IEEE Computer Society, 2006: 629 – 636.

[7] Koyama K, Iguchi K, Hosono S, et al. Web services proxy: an extensible platform for intermediaries of XML networks [C]//*Proc of the IEEE International Conference on Web Services*. Los Alamitos: IEEE Computer Society, 2005: 246 – 253.

[8] Duftler M, Curbera F, Khalaf R. Business process with BPEL4WS [EB/OL]. (2003-03-11) [2007-04-30]. www.ibm.com/developerworks/webservices/library/ws-bpelcol5/.

基于 web service 代理的业务绩效监控

陈再本[1,2]　邢春晓[2]　杨吉江[2]　胡庆成[1,2]

([1] 清华大学计算机科学与技术系,北京 100084)
([2] 清华大学信息技术研究院,北京 100084)

摘要:为了提供对基于 web service 业务流程的有效监控,开发了 web service 代理(WS-proxy)用于监控服务的交互进而监控业务活动的绩效. WS-proxy 以中间结点的形式部署在内部的业务流程和外部的服务提供者之间,同时它提供了消息路由和消息内容检查功能. 通过利用一个 XPath 引擎(WS-filter),它可以将绩效指标数据从服务消息中抽取出来进而评估业务绩效. 在实验中,WS-proxy 的可用性和监控效率得到了验证. WS-proxy 带来的延迟大约只占到整体延迟的 15%,并且抽取绩效指标数据仅消耗了不到 1/3 的处理时间.

关键词:绩效;流程;监控;web 服务;代理

中图分类号:TP311