Journal of Computer Science 5 (7): 487-492, 2009
ISSN 1549-3636
© 2009 Science Publications

Parallel Performance Monitoring Service for Dynamically Composed Media Web Services

[1]G. Maria Kalavathy and [2]P. Seethalakshmi
[1]Sathyabama University, Chennai, Tamil Nadu, India
[2]Anna University Trichy, Tamil Nadu, India

Abstract: Problem statement: As distributed real-time multimedia services are accessed by users from different locations, the run-time performance must be managed and monitored to get a clear view of how media web services perform within their operational environments and to perform control actions to modify and adjust the their behavior. Otherwise it will provide negative effects on the reputation of service provider. Therefore the main objective of this study was to design and implement a media web service composed dynamically and to monitor the Dynamically Composed Media Web Services (DCMWS) through Parallel Performance Monitoring Service (PPMS). The PPMS is a web service that executed in parallel with media web service using multi threading technology. **Approach:** The PPMS monitors the run-time performance of Dynamically Composed Media Web Services (DCMWS) which were represented as BPEL (Business Process Execution Language) processes. The run-time performance of the media web services such as timeouts, external errors, percentage of successful completion of individual media web services and the occurrence of fault were monitored and using this result the corrective actions were taken by service providers. **Results:** The effectiveness of PPMS had been evaluated for media-on-demand composite web service and its results showed an improvement on the performance of run time monitoring of media web services. **Conclusion:** To facilitate management decisions and media web service behavior modifications, service providers need to monitor the run-time performance of DCMWS. In this context, the approach outlined in this study was most appropriate, convenient and efficient. The proposed PPMS herein played a crucial role in monitoring run-time performance of DCMWS.

Key words: Dynamic service composition, media-on-demand, multi threading, PPMS

INTRODUCTION

Emerging advances in distributed media services, such as video conferencing, media-on-demand and ubiquitous multimedia streaming, demands a scalable, robust and adaptive media service infrastructure. The media service composition concepts are main approaches to advance construction of large scale distributed media services in a scalable, easy-programmable and efficient manner[16]. Since the media service compositions are done dynamically the name for the composed service is Dynamically Composed Media Web Service (DCMWS). The DCMWS and its monitoring at run time are necessary to provide adaptive media web services. The dynamic service composition to media service allows the media services to be composed dynamically from components of distributed web services according to the requirements from different users. The components of distributed media services were developed and hosted in heterogeneous environments. This Composite Media web service provides streaming of real-time media that allows the user to choose the media services according to their choice in a user friendly way that provides the flexibility in choosing the kinds of information they would like to receive. One of the challenges is, ensuring the high performance of multiple media services that are requested. The demanded services are to be readily available to the end users; else it will create negative effects on the reputation of service provider or result in loss of business opportunities. Hence the runtime performance of the web services is to be monitored and is informed to the service provider to take corrective action. The performance of the web services are influenced by many factors such as network traffics, host workloads and host running environments. It is very important to monitor the run time performance of dynamically composed media web services during the process of their invocation in client side. There are few reported researches on performance monitoring of

Corresponding Author: G. Maria Kalavathy, Sathyabama University, Chennai, Tamil Nadu, India

media web services especially video-on-demand services after it is deployed.

The DCMWS are designed and implemented as BPEL processes and using multithreading technology the Parallel Performance Monitoring Service (PPMS) monitors the performance of each individual media web services that are requested. The runtime performance characteristics to be monitored are response time, timeout, external errors, percentage of successful completion of individual web services and the occurrence of fault. The results of this run time performance monitoring are used by service providers to take corrective actions.

The response time is calculated using software wrapping technique at the server side. The timeout is the interrupt signal that is generated when the client does not get the response with in the specified time limit. PPMS monitors the timeout and executes exception handling procedure if the client does not get the response with in the time limit.

Calculating the percentage of successful completion of media web service for example video-on-demand service is useful to restart the service from the point it was failed if any fault is occurred. The calculation of percentage of successful completion is done by polling technique. The monitor thread of video-on-demand service polls the client at every specified interval of time and if it responds, stores the successful completion in terms of percentage to the log file.

Related work: Performance is one of the most important nonfunctional requirements of service based media applications. Because service-based software development for media applications is emerging new technology, there have been no reported performance assurance studies on media web service applications. Most of the performance assurance testing is performed before the deployment of the web services.

Compile-time analysis techniques to perform the white-box testing of exception handlers in Java web services are analyzed by Fu *et al*.[3]. Huang *et al*.[5] were designed a software tool to assess web application security which is based on software testing techniques such as dynamic analysis, black-box testing, fault injection and behavior monitoring. Offutt and Xu[12] proposed an approach to test web services based on data perturbation and interaction perturbation, which uses two types of communication mechanism such as RPC communication and data communication. Liu *et al*.[9] proposed a web test model, which considers each web application component as an object and generates test cases based on data flow between those objects. Gorton and Liu[4] designed a middleware infrastructure, the transaction and directory services and the load balancing to compare the performance of six different J2EE-based distributed applications. Avritzer *et al*.[8] compared the performance of different Object Request Broker (ORB) implementations that are related to the CORBA Component model. Liu *et al*.[10] evaluated the suitability of light-weight test cases on distributed applications.

McGregor and Schiefer[2] described a framework which uses process definition information to define web service to the solution manager service. They also introduced the concept of the Event Processing Container providing a robust, scalable and high-performance event processing environment able to handle a large number of process events in near real-time. Baresi *et al*.[11] proposed an approach to monitor timeouts, runtime errors and violations of functional contracts of service compositions defined by BPEL processes using assertions. Liguo Yu[7] proposed software wrapping technique that is used at client side. The clients interacts with the service through the wrapper which customize the messages exchanged between client and service and monitors the performance of the service by calculating the response time only. But here the response time is calculated using software wrapping technique from the service provider point of view to take immediate action if the performance is poor. Mahbub *et al*.[6] described the framework to monitor behavioral properties and assumptions at run time using event calculus. William N. Robinson[15] proposed REQMON monitoring system that raises only an alert by sending a failure message to the global monitor. It can’t recover web service from the point at which the failure occurs. Koschel and Astrova[1] designed a configurable event monitoring web service which is useful in the context of Event Driven Architectures (EDA) and Complex Event Processing (CEP). Ezenwoye and Sadjadi[13,14] presented an approach to transparently adapting BPEL processes to tolerate runtime and unexpected faults and to improve the performance of overly loaded web services. They presented an approach in which when one or more partner services do not provide satisfactory service the request for service is redirected to one of these static, dynamic and generic proxies, where the failed or slow services are replaced by substitute services[16]. But this approach is not used for recovering the web service from the point at which the fault is occurred.

There is no reported research on performance monitoring of media web services especially percentage of successful completion of media web service. Hence it has been chosen to design and implement DCMWS and its run time performance has been monitored by PPMS. It is very much useful in media web applications to recover from the point at which the fault is occurred.

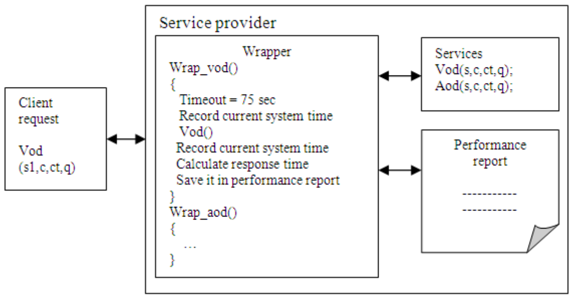
MATERIALS AND METHODS

Parallel Performance Monitoring Service: The PPMS monitors the run time performance of DCMWS using three mechanisms such as software wrapping in server side, setting and monitoring time limits for individual media web services and polling technique to monitor web service compositions defined by BPEL. These three mechanisms correspond to four classes of performance characteristics such as response time, timeouts, external errors and percentage of successful completion of the media web service.

For each media service requested, PPMS thread is created that checks the performance in terms of response time, timeout, external error and percentage of successful completion. When such undesirable condition is detected by the PPMS, it returns status such as Service Unavailable Fault, Timeout fault and execution fault during VoD services is delivered to service providers for handling them.

Monitoring timeouts and response time: The timeout is the interrupt signal that is generated when the client does not get the response with in the specified time limit. ActiveBPEL allows the designer to set the timeout for scopes and all the operations in the scope must finish their execution within the time limit set by timeout. But PPMS set the time limit and exception handling procedures for each and every individual web services and monitors this time limit at runtime. The time limit is chosen as 75 sec using the default value of the parameter tcp_ip_abort_cinterval which is the second threshold timer used during connection establishment.

The response time is the delay between a request and the completion of an operation which is monitored by software wrapping technique in the server side. Software wrapping refers to a reengineering technique that surrounds a software component or system with a new software layer to hide the internal code and the logic of the component or system and to supply modern interfaces. The server receives the client request through wrapper. Here the wrapper provides the customization of messages exchanged between the client and the service and to monitor the performance of the service i.e., calculating response time.



```
Wrap_vod()
{
  Timeout = 75 sec
  Record current system time
  Vod()
 Record current system time
 Calculate response time
 Save it in performance report
}
Wrap_aod()
{
   ...
}
```

```
Vod(s,c,ct,q);
Aod(s,c,ct,q);
```

Fig. 1: An example of wrapping media web services

Figure 1 shows an example of a wrapper program that monitors the response time of the video-on-demand service. The client request for the video service comprises of service name, cost, completion time and quality (vod(s1,c,ct,q)) which is given to server through wrapper. The wrapper program calculates the response time, checks the timeout and records the results in the performance report i.e., log file. In this approach software wrapping is implemented at the service provider side, because the PPMS which is used in this approach is used to monitor the performance of the media web service from the server point of view to take the immediate action if the performance is poor.

Monitoring external errors: When the individual web service in the dynamic composition fails because of an unforeseen internal bug and external errors, the whole composition i.e., process is failed. To recover from this situation, PPMS defines faultHandler that can take care of the failure of the invoked service using a catchall clause. A part of sample code of PPMS to handle the faults occurred in each individual web service in the composition is shown below:

```
<faultHandlers>
<catchall>
<sequence>
    …logic to handle exceptions by communicating the error to the service provider to take necessary steps and terminating….
</sequence>
</catchall>
</faultHandlers>
```

Monitoring the percentage of successful completion: When the Video-on-Demand (VoD) service is requested, a thread is created for PPMS to monitor the performance of media web service and synchronization is achieved between them to exchange data. Parallelizations of web services with communication construct are provided by Multithreading technology with synchronization. The parallelism with communication construct is used to execute concurrent web services and to synchronize or exchange certain data between them during execution. The Fig. 2 explains the parallel execution of VoD web service and PPMS with communication among the services. The special processing element facilitates data sending and retrieval and formalizes it as process related instances.

Figure 2 shows the simultaneous operation of PPMS and VoD web service. The information about the status of the VoD service is monitored by PPMS at different times using polling technique and stores the status such as polling time, correctness of the service, polling time interval in the polling table. The correctness of the service is indicated by the flag variable called response. The sample polling table is shown in Table 1. At run time the percentage of completion is calculated by the following equation using the information given by PPMS:

$$\text{Percentage of successful completion (P)} = \frac{n \times Pt}{N} * 100 \qquad (1)$$

Where:

n = Number of polls
Pt = Polling time interval
N = Total time required

Table 1: Polling table

	Polling time (Pt) in sec	Response
Poll 1	120	YES
Poll 2	120	YES
Poll 3	120	NO

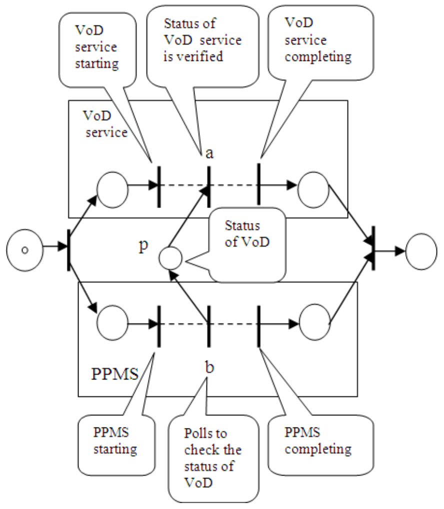


Fig. 2: Parallel execution of VoD Web service and PPMS

Let the polling time interval (pt) is 2 min and the video clip can run for 10 min. Then PPMS polls for 5 times. If the third poll does not get the response, then it is intimated immediately to take necessary action. The action to be taken is out of scope of this study and will be done in future using corrective adaptation techniques.

RESULTS

To evaluate the proposed PPMS, a series of experiments on dynamically composed media web services such VoD service executions were carried out. In this experiment BPEL process was created for media web services. This process includes the individual web services such as authentication service, new user service, Search service, SLA service, Selection service, PPMS, VoD service, NoD (News-on-Demand) service and MoD (Music-on-Demand) services. The Fig. 3 is the BPMN (Business Process Modeling Notation) diagram used to explain the composition of these web services.

The objective of this experiment was to measure (a) Monitored response time for multiple clients (b) Number of timeouts detected at run time (c) Percentage of successful completion of VoD service and (d) Number of exceptions raised due to external errors. The performance parameter response time of the requested media web service is the sum of the response time of the individual services in the composition described in the Fig. 3. The Response Time (RT) of the total composition for the new user request is calculated using software wrapping technique as below:

$$RY = rt1 + \sum_{i=3}^{6} rti + rt7 \qquad (2)$$

where, rt_i is the response time of the i^{th} web service.

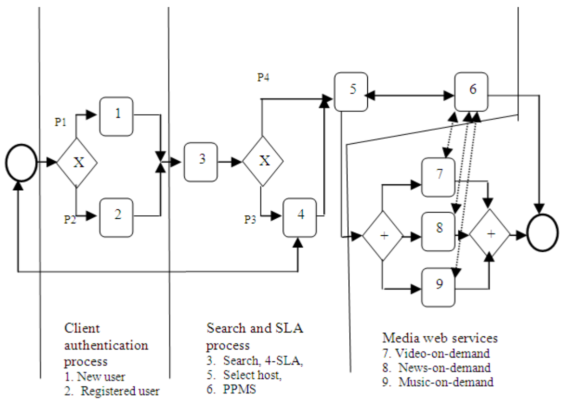
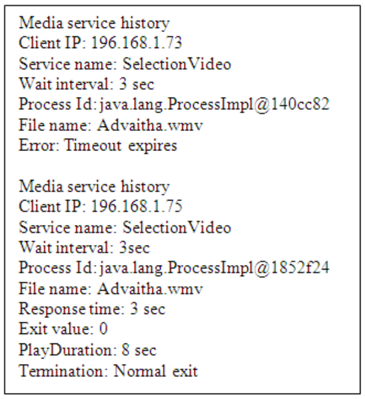


Fig. 3: BPMN diagram for dynamic composition of media web services



```
Media service history
Client IP: 196.168.1.73
Service name: SelectionVideo
Wait interval: 3 sec
Process Id: java.lang.ProcessImpl@140cc82
File name: Advaitha.wmv
Error: Timeout expires

Media service history
Client IP: 196.168.1.75
Service name: SelectionVideo
Wait interval: 3sec
Process Id: java.lang.ProcessImpl@1852f24
File name: Advaitha.wmv
Response time: 3 sec
Exit value: 0
PlayDuration: 8 sec
Termination: Normal exit
```

Fig. 4: Sample output of PPMS

The sample output of log file which is created at run time is shown in Fig. 4. The parameters that are monitored and stored in this log file are client ID, service name, wait interval, process ID, media file name, response time and errors if occurred.

The graph for dynamic requests is shown in Fig. 5 which describes the variation in response time based on the number of simultaneous requests handled by the service provider. As shown in the graph, the response time increases almost linearly until about 10 concurrent requests. After 10 requests, the performance in terms of response time degrades considerably.

The next performance parameter is the timeout which is the interrupt signal that is generated when the client does not get the response with in the specified time limit. The network traffic conditions and server workloads leads to timeout of the service that is it can not respond with in the specified time interval. Active BPEL allows the designer to set the timeout for scopes and all the operations in the scope must finish their execution within the time limit set by timeout. But PPMS sets the time limit as 60 sec and exception handling procedures for each and every individual web services and monitors this time limit at runtime.

The next performance parameter the percentage of successful completion of the media service was found using polling technique. The PPMS thread parallely monitors the media web service by sending polling message and checks the response. If the response is received, it is entered in the polling table. Otherwise the service provider checks the polling table and calculates the percentage of successful completion of media service using the Eq. 1.

The external errors are monitored by PPMS using passive testing and the corresponding exception handler is invoked to give the information about the external errors to the service providers. The passive testing is the testing or observing the interaction between a service provider and clients.

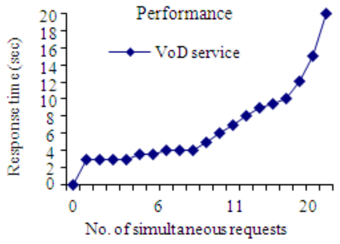


Fig. 5: No. of simultaneous requests Vs response time

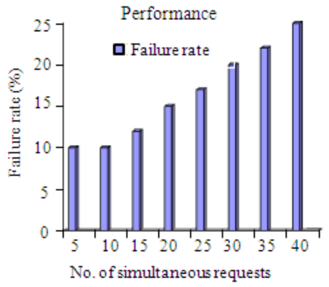


Fig. 6: No. of simultaneous requests Vs failure rate

The external errors are simulated that is restart the client or pause the operation in between. All these errors are monitored and the Fig. 6 shows the failure rate of VoD service due to external errors. It is measured as the ratio of the number of failure requests over total number of requests over a period of time.

DISCUSSION

It was observed that the deigned PPMS monitor the run-time performance of the DCMWS effectively. The monitoring of percentage of successful completion of media web service execution is very much necessary for the media service providers to take necessary action in case of fault occurrence. This will increase the profit and reputation of their business. The findings of failure rate and response time of number of simultaneous requests will be very useful to perform control actions to modify and adjust the behavior of media web services.

CONCLUSION

The Dynamically Composed Media Web Services (DCMWS) was designed and implemented and through Parallel Performance Monitoring Service (PPMS) its runtime performance was monitored. Software performance assurance of web services has not been thoroughly investigated because of it its dynamic binding and independent service implementation. But the PPMS provides three mechanisms such as software wrapping in server side, setting and monitoring time limits for individual media web services and polling technique to monitor the successful completion of media web service compositions defined by BPEL. These three mechanisms were implemented to monitor the performance characteristics such as response time, timeouts, external errors and percentage of successful completion of the media web service. These findings support the importance of monitoring the dynamically composed media web services and also show the improvements on run-time performance monitoring of media web services.

REFERENCES

1. Koschel, A. and I. Astrova. 2008. Event monitoring web services for heterogeneous information systems. Proceedings of the World Academy of Science Engineering and Technology, Sept. 2008, pp: 50-52. http://www.waset.org/pwaset/v33/v33-10.pdf
2. McGregor, C. and J. Schiefer, 2003. A framework for analyzing and measuring business performance with web services. Proceedings of the IEEE International Conference on E-Commerce, June 24-27, IEEE Xplore Press, USA., pp: 405-412. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1210277
3. Fu Ryder, C., B.G. Milanova and D. Wonnacott, 2004. Testing of java web services for robustness. Proceedings of 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis, July 11-14, ACM Press, Boston, Massachusetts, USA., pp: 23-24. http://doi.acm.org/10.1145/1007512.1007516
4. Gorton, I. and A. Liu, 2002. Software component quality assessment in practice: Successes and practical impediments. Proceedings of the 24th International Conference on Software Engineering, May 19-25, ACM Press, Orlando, Florida, pp: 555-558. http://doi.acm.org/10.1145/581339.581408
5. Huang, Y., S. Huang, T. Lin and C. Tsai, 2003. Web application security assessment by fault injection and behavior monitoring. Proceedings of 12th International World Wide Web Conference, Budapest, May 20-24, ACM Press, Budapest, Hungary, pp: 148-159. DOI: 10.1145/775152.775174
6. Mahbub, K. and G. Spanoudakis, 2005. Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. Proceedings of the IEEE International Conference on Web Service, July 11-15, IEEE Xplore Press, USA., pp: 257-265. DOI: 10.1109/ICWS.2005.100
7. Liguo Yu, 2007. Applying software wrapping on performance monitoring of web services. J. Comput. Sci., 6: 1-6. http://www.dcc.ufla.br/infocomp/artigos/v6.3/art01.pdf
8. Lin, C., A. Avritzer, E. Weyuker and L. Sai-Lai, 2000. Issues in interoperability and performance verification in a multi-ORB telecommunications environment. Proceeding of the International Conference on Dependable Systems and Networks, June 25-28, IEEE Computer Society, Washington, DC., USA., pp: 567-575. http://portal.acm.org/citation.cfm?id=737949
9. Liu, C., D. Kung and P. Hsia, 2000. Structural testing of web applications. Proceedings of 11th International Symposium on Software Reliability Engineering, Oct. 8-11, IEEE Xplore Press, San Jose, CA., USA., pp: 84-96. DOI: 10.1109/ISSRE.2000.885863
10. Liu, Y., I. Gorton, A. Liu, N. Jiang and S. Chen, 2002. Designing a test suite for empirically-based middleware performance prediction. Proceedings of the 14th International Conference on Tools Pacific: Objects for Internet, Mobile and Embedded Applications, (ICTPOIMEA'02), ACM Press, Sydney, Australia, pp: 123-130. http://portal.acm.org/citation.cfm?id=564110
11. Baresi, L., C. Ghezzi and S, Guinea, 2004. Smart monitors for composed services. Proceedings of the 2nd International Conference on Service Oriented Computing, Nov. 15-19, ACM Press, New York, USA., pp: 193-202. DOI: 10.1145/1035167.1035195
12. Offutt, J. and W. Xu, 2004. Generating test cases for web services using data perturbation. ACM. SIGSOFT. Software Eng. Note., 29: 1-10. http://doi.acm.org/10.1145/1022494.1022529
13. Ezenwoye, O. and S.M. Sadjadi, 2008. Proxy-based approach to enhancing the autonomic behavior in composite services. J. Network., 3: 42-53. http://www.bibsonomy.org/bibtex/2e42854cca027fb15459f277c75044f10/dblp
14. Ezenwoye, O. and S.M. Sadjadi, 2006. Robust-BPEL: Transparent autonomization in aggregate web services using dynamic proxies. Technical Report: FIU-SCIS-2006-06-01. http://www.cs.fiu.edu/~sadjadi/Publications/Technical%20Report%20FIU-SCIS-2006-02-01-RobustBPEL-DynamicProxies.pdf
15. William, N., Robinson. 2003. Monitoring web service requirements, Proceedings of the 11th IEEE International Requirements Engineering Conference, Sept. 8-12, IEEE Xplore Press, USA., pp: 65-74. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1232738
16. Balke, W.T. and K. Nahrstedt, 2004. Multimedia service composition: A brave new topic. Proceedings of ACM Multimedia Conference, Oct 10-15.New York, USA., pp: 1-2. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.6927&rep=rep1&type=pdf