Probabilistic Critical Path Identification for Cost-Effective Monitoring of Service-based Web Applications

Qiang He, Jun Han, Yun Yang and Jean-Guy Schneider
Faculty of Information and Communication Technologies
Swinburne University of Technology
Melbourne, Australia 3122
{qhe, jhan, yyang, jschneider}@swin.edu.au

Hai Jin
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, China 430074
hjin@hust.edu.cn

Steve Versteeg
CA Labs
Melbourne, Australia 3122
steve.versteeg@ca.com

ABSTRACT

The critical path of a composite Web application operating in volatile environments, i.e., the execution path in the service composition with the maximum execution time, should be prioritised in cost-effective monitoring as it determines the response time of the Web application. In volatile operating environments, the critical path of a Web application is probabilistic. As such, it is important to estimate the criticalities of the execution paths, i.e., the probabilities that they are critical, to decide which parts of the system to monitor. We propose a novel approach to the identification of Probabilistic Critical Path for Service-based Web Applications (PCP-SWA), which calculates the criticalities of different execution paths in the context of service composition. We evaluate PCP-SWA experimentally using an example Web application. Compared to random monitoring, PCP-SWA based monitoring is 55.67% more cost-effective on average.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; H.3.4 [**Systems and Software**]: Distributed systems

Keywords

Web application, service composition, Web service, monitoring, critical path.

1. INTRODUCTION

The response time, among various QoS properties, is of particular significance and challenge in QoS management for service-based Web applications built through dynamic compositions of loosely coupled component services. During the execution of a Web application, runtime anomalies may occur and jeopardise the quality of the Web application [2]. In order to detect and predict runtime anomalies timely, we need to monitor the execution of the basic components (BCs) that compose a Web application, i.e., the component services and the data transmissions between the component services. However, monitoring consumes resources, including software, hardware and sometimes human resources. In large-scale environments, the issue of *monitoring cost*, i.e., the cost of monitoring in terms of monitoring resources, is particularly critical. For example, in the cloud environments, a cloud service provider may maintain up to hundreds of thousands of services for their clients [3]. It is to the cloud service providers' best benefits to maintain the monitoring cost at a reasonable and affordable level while being able to guarantee the response time of their applications.

The key to cost-effective monitoring for a service-based Web application is the identification and monitoring of its critical path (i.e., the execution path with the maximum execution time) because the runtime anomalies that occur on the critical path will cause delays that directly impact the response time of the Web application. However, the volatility of the operating environments makes the critical path of a Web application probabilistic - every execution path can be critical with certain probabilities. Thus, identifying the critical path turns into calculating those probabilities which represent their *criticalities* in the service composition.

Our approach, namely Probabilistic Critical Path for Service-based Web Applications (PCP-SWA), uses a novel timing model to capture the probabilistic nature of the service-based Web applications. This model allows developers to evaluate and analyse service-based Web applications in a more realistic way. PCP-SWA also provides a method that helps developers calculate the criticalities of different execution paths of service-based Web applications.

2. CRITICALITY EVALUATION

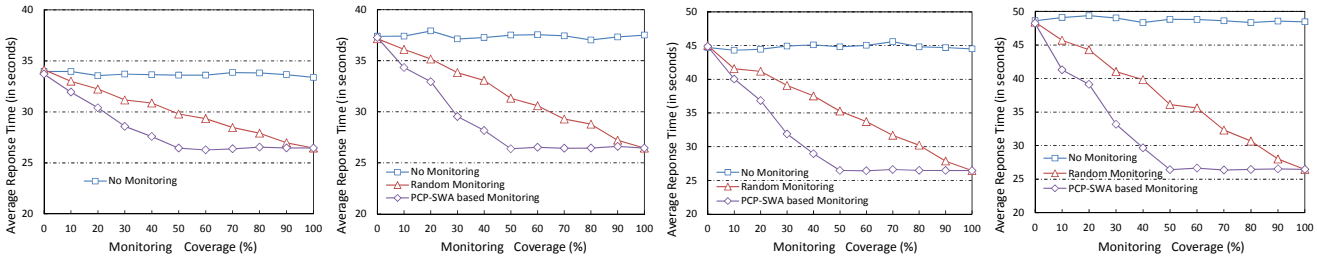
We model the response time of a BC S_i in the standard form as:

$$T_R(S_i) = t_0 + \sum_{i=1}^{n} w_i \cdot \Delta X_i \quad (1)$$

where t_0 is the mean value of $T_R(S_i)$; ΔX_i, i=1, 2, …, n, represent the variation of n sources of anomaly X_i, i=1, 2, …, n, from their mean values; w_i, i=1, 2, ..., n, represent the sensitivities of $T_R(S_i)$ to each of the sources of anomaly. t_0, w_i and the distributions of ΔX_i can be evaluated by inspecting S_i's past executions, service consumers' feedbacks, service providers' profiles, etc.

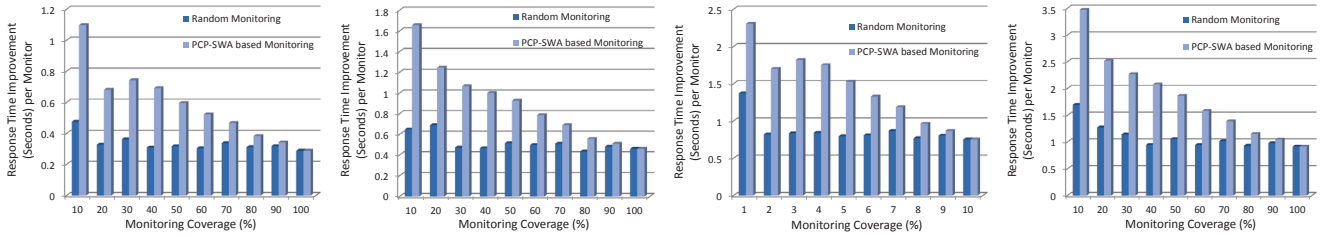
Given the response times of the BCs in a service composition, we can calculate their start times, denoted by T_S, and finish times, denoted by T_F. Next, we analyse the timing dependencies between those BCs by calculating their dominance probabilities, i.e., the probability that they solely determine the start time of their succeeding BC(s). For example, given two BCs S_i and S_j in a *sequence* structure where S_j is activated when S_i is finished, the dominance probability of S_i, denoted by $D(S_i)$, is 1.0 because the start time of S_j is always solely determined by the finish time of S_i. In a *parallel* structure where edges E_1, …, E_n merge into an

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.



(a) fault rate=0.1 **(b) fault rate=0.2** **(c) fault rate=0.3** **(d) fault rate=0.4**

Figure 1. Average response time.



(a) fault rate=0.1 **(b) fault rate=0.2** **(c) fault rate=0.3** **(d) fault rate=0.4**

Figure 2. Response time improvement per monitor.

succeeding edge E_s, let $Z \square \max(T_F(E_1), \ldots, T_F(E_{i-1}), T_F(E_{i+1}), \ldots, T_F(E_n))$, the dominance probability of E_i $(1 \le i \le n)$ is calculated as:

$$D(E_i)=P(T_F(E_i) \ge Z)=P(Z-T_F(E_i) \le 0)= F_{Z-T_F(E_i)}(0) \qquad (2)$$

where $F_{Z-T_F(E_i)}$ is the cumulative probability function of Z-$T_F(E_i)$.

In order to perform criticality calculation, we need to identify all possible execution scenarios that do not contain branch or loop structures from the service composition. Based on the BCs' dominance probabilities, the criticalities of different execution paths can be evaluated in each of the identified execution scenario following a certain rule: *the criticality of an execution path in an execution scenario is the product of the dominance probabilities of all the edges that belong to the execution path*. Then, the criticality of an execution path in the service composition can be computed by a weighted average over its criticalities obtained in all the execution scenarios using the execution scenarios' execution probabilities as weights.

3. EXPERIMENTS

We developed a prototype of PCP-SWA and experimented on *OnlineLive*, a live-on-demand Web application that converts, subtitles and transmit various live video streams. To simulate a distributed operating environment, we generated the response times of the BCs of *OnlineLive* based on a publicly available Web service dataset QWS [1], which comprises measurements of nine QoS parameters (including response time) of over 2500 real-world Web services. During the execution of *OnlineLive*, a certain number of anomalies were generated based on the fault rate and randomly introduced to the BCs. We increased the fault rate from 10% to 40% in steps of 10% to simulate increasing levels of volatility in the operating environment. When anomalies occurred to unmonitored BCs, randomly generated delays were applied to corresponding BCs. If a BC was being monitored, the delay was avoided (representing the fact that the anomalies were detected or predicted and adaptation actions were taken on time to fix the anomalies). Three sets of experiments were conducted in each volatile environment. In set #1, no monitors were allocated. In set #2, the monitors were randomly allocated to the BCs. In set #3, the monitors were allocated according to the criticalities of the execution paths from high to low.

Figure 1 demonstrates the average response time of *OnlineLive* obtained in different volatile environments. As the fault rate increases, the average response time increases because more anomalies cause longer total delay. Random monitoring and PCP-SWA based monitoring improved the response time of *OnlineLive* by an average of 17.87% and 27.80% respectively across all experimental cases. Figure 1 also shows how much monitoring resource is needed to meet different levels of response time requirements. Take Figure 1(d) for example, to guarantee that the average response time is below 30 seconds, random monitoring requires at least 80% monitoring coverage while the number for PCP-SWA based monitoring is only 40%.

Figure 2 compares the *response time improvement per monitor* obtained by random monitoring and PCP-SWA based monitoring. As illustrated, PCP-SWA based monitoring demonstrates significant advantage over random monitoring by an average margin of 55.67%. This observation indicates much higher cost-effectiveness of PCP-SWA than random monitoring. Moreover, the cost-effectiveness of PCP-SWA increases as the monitoring coverage decreases, which shows that PCP-SWA is particularly cost-effective when the monitoring resources are relatively limited.

ACKNOWLEDGMENTS

This work is partly funded by the Australian Research Council in collaboration with CA Labs.

4. REFERENCES

[1] Al-Masri, E. and Mahmoud, Q. H. Investigating Web Services on the World Wide Web. In *Proceedings* of *the 17th International Conference on World Wide Web (WWW2008).* pages 795-804, 2008.

[2] Baresi, L. and Guinea, S. Self-Supervising BPEL Processes. *IEEE Transactions on Software Engineering,* 37, 2, 2011, 247-263.

[3] Candan, K. S., Li, W.-S., Phan, T., and Zhou, M. Frontiers in Information and Software as Services. In *Proceedings* of *the 25th International Conference on Data Engineering (ICDE2009).* pages 1761-1768, 2009.