Proactive Problem Determination in Transaction-Oriented Applications

Soila Pertet and Priya Narasimhan
Electrical & Computer Engineering Department
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
{spertet,priyan}@ece.cmu.edu

Anca Sailer and Gautam Kar
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
{gkar,ancas}@us.ibm.com

System performance gets a lot less publicity than "hard" downtime. However, recent studies [3] indicate that performance problems in e-commerce systems are about five times more likely than unavailability problems. Determining the root cause of a performance problem can be a daunting task for system administrators, especially when the system overwhelms them with application and performance metrics that provide little insight or no insight on the cause of the failure. We present ongoing work which investigates the minimal level of system monitoring required to adequately diagnose performance problems in transaction-oriented applications. We plan to exploit system monitoring to identify pre-fault *"symptoms"* and initiate proactive (rather than reactive) fault-recovery. Proactive fault-recovery could significantly lower the impact of application performance problems.

1. Background

This research extends work done by Sailer et al [1] that presents a 3-tier performance monitoring architecture for identifying the root cause of end-to-end response time Service Level Objective (SLO) violations in transaction-oriented applications. Their 3-tier architecture is described below:

- *Tier 1:* Monitoring agents specific to server platforms, *e.g.*, HTTP servers, Web Application Servers (WAS), messaging servers, and databases, extract component-specific monitoring data and send it to the second tier.
- *Tier 2:* A management service extracts dependencies between components, assigns weights to the extracted dependencies, and stores them in a repository. For example, the service may determine Transaction T1 depends fully on Servlet S1, and S1 depends on query Q1 for 30% of the time [1] (see Figure 1).
- *Tier 3:* This tier localizes the root cause of problems by decomposing Transaction-SLOs (T-SLOs) into component-SLOs (c-SLOs) [2]

1 S1 may call Q1 or Q2, or Q1 and Q2

When an SLO monitor observes an end-to-end response-time violation, the individual components associated with that transaction are automatically ranked by the degree to which they violate their constructed c-SLO. A system administrator can then scan the limited set of ranked components and quickly determine the actual root cause through more detailed examination.

Additionally, we will incorporate key research ideas from Pertet and Narasimhan [4], which describe proactive fault-recovery strategies for distributed middleware systems in the presence of resource-exhaustion faults. They analyze the effect of different proactive fault-recovery schemes on client/server response times, and demonstrate that proactive fault-recovery can significantly reduce jitter and the number of user-visible failures.

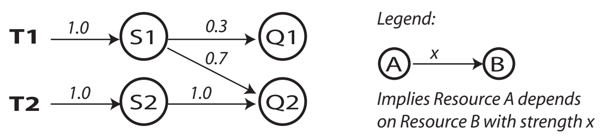

Figure 1. Dependency Graph.

2. Related Work

In the past, problem determination techniques have concentrated on network, and system-level fault management. Our work focuses on pinpointing application performance problems. Steidner and Sethi [6] review existing approaches to fault localization and highlight the key challenges facing them. Rish et al [5] describe an active probing technique for real-time problem determination which selects probes on-demand and reduces the time spent localizing problems. Chen et al [2] use eBay's Centralized Application Logging (CAL) and decision trees to diagnose application-level failures. Through our research, we plan to identify a near-optimal monitoring-granularity for problem determination in transaction-oriented applications.

2 The c-SLO is the threshold for the time spent by a transaction locally at a component out of the end-to-end T-SLO threshold allowed for that individual transaction.

Module	Metric	Granularity	Level of Monitoring
Web application	Number of servlets loaded	per Web application	Low
	Number of concurrent requests	per servlet	High
Database connection pool	Total number of faults, $e.g.$, timeouts	per connection pool	Low
	Average waiting time	per connection pool	Medium

Table 1. Sample Metrics from Websphere's Performance Monitoring Infrastructure(PMI).

3. Our Approach

Sailer et al [1] used the available middleware instrumentation and a single metric, $i.e.$, end-to-end response time violations, for problem determination. We are extending their system to to achieve more fine-grained problem determination and to incorporate proactive fault-tolerance by exploiting the additional metrics available in today's systems.

We investigate what level of monitoring will allow us to pinpoint a large class of application performance problems, and yet introduce minimal overhead. Finding the smallest set of performance metrics required is NP-hard, therefore, we will use approximation techniques to find near-optimal monitoring levels for problem determination. The key research challenges we seek to address are outlined below:

Decomposing transaction-SLOs (T-SLOs) to component-SLOs (c-SLOs): Sailer et al [1] manually profiled their system and found that groups of transactions exhibited similar behavior and could share the same c-SLO threshold. This simplified the task of managing c-SLOs in large, complex systems. We are investigating ways to automatically detect similarities between transaction behavior.

Identifying fault injection strategies: We need to identify the representative faults, which yield a wide coverage of the typical performance problems in experienced in transaction-oriented applications, $e.g.$, locking database tables, reducing the database buffer size, limiting thread pools on the application server, $etc.$

Localizing the most likely causes of the failure: When a transaction violates its SLO, the individual components are automatically ranked by the degree to which they violate their constructed c-SLO. This yields a set of likely suspects. We would like to assess the effectiveness of different fault-localization techniques, $e.g.$, decision trees, especially when faced with noisy data.

Drilling down on faulty component using system metrics: Most systems have monitors that collect information on typical metrics. Table 1 gives a sample of the metrics available in IBM's Websphere Application Server. We would like to correlate these metrics with constructed SLOs for more fine-grained problem determination. For partial failure models, we first identify the likely set of faulty components before examining the system metrics. However, when dealing with system-wide failures, it might be more efficient to examine the system metrics first. Since monitoring can introduce significant overhead, we need to determine the granularity and frequency of monitoring that is sufficient for problem determination.

System evaluation: We will measure the overheads associated with instrumentation in the context of realistic applications, $e.g.$, the TPC-W benchmark. We will also evaluate the effectiveness of the problem determination mechanisms that we develop in terms of (1) the number of failures correctly diagnosed, (2) the number of false positives contained in the candidate set of faulty componenents, and (3) the suitability of these mechanisms for online problem determination.

Proactive fault-tolerance: We plan to use either artificial intelligence or statistical techniques to identify patterns of abnormal behavior in the performance metrics. If these patterns exist for certain kinds of faults, we can apply proactive fault-recovery techniques to reduce the probability that a user experiences a performance slowdown.

References

[1] M. K. Agarwal, A. Neogi, K. Appleby, J. Faik, G. Kar, and A. Sailer. Threshold management for problem determination in transaction based e-commerce systems. In *To appear in the International Symposium on Integrated Network Management*, Nice, France, May 2005.

[2] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing*, pages 36–43, New York, NY, May 2004.

[3] T. Kuegler. The billion-dollar question: The impact of web site performance on e-commerce. *MarketingProfs.com*, February 2005.

[4] S. Pertet and P. Narasimhan. Proactive recovery in distributed CORBA applications. In *International Conference on Dependable Systems and Networks*, pages 357 – 366, Florence, Italy, June 2004.

[5] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik. Real-time problem determination in distributed systems using active probing. In *Network Operations and Management Symposium*, pages 133–146, Seoul, Korea, April 2004.

[6] M. Steidner and A. S. Sethi. The present and future of event correlation: A need for end-to-end service fault localization. In *World Multicongerence on Systemics, Cybernetics and Informatics*, pages 124–129, Orlando, FL, July 2001.