Deep Root-Cause Performance Analysis: Integration and Navigation from a Load Testing Tool

Yoonki Song   Tao Xie
Department of Computer Science
North Carolina State University, Raleigh, USA
{ysong2, txie}@ncsu.edu

ABSTRACT

Many performance-testing tools can measure client-side response time and correlated it with collected application transaction monitoring data. Such correlation can break down client-side response time into its constituent parts on the various tiers of the server(s) under test. To avoid high overhead, some other performance tools for Java applications perform runtime profiling by using JVM profiling with statistical sampling. However, such an imprecise profiling mechanism poses challenges for performance analysis. It is challenging to identify the correlation between a transaction or time-range on the client side and the profiling information at a server's JVM. Therefore, it is difficult to provide user-friendly drill-down navigation from client-side response time or time-range to determine problem spots and bottlenecks in the JVM(s) on the server sides because of a lack of correlation between the transactions or time-range driven by a load testing tool at the client side and the runtime sample data collected by the JVM profiling tools at the server sides.

In the paper, we have developed a new technique and tool to help to address the preceding issues in deep root-cause performance analysis. We evaluate the accuracy, efficiency, and effectiveness of our approach by conducting experiments with several real-world web applications.

1. INTRODUCTION

Performance testing and analysis tools help testers to identify and diagnose the presence and cause of application or system bottlenecks. In performance testing, testers create loads to expose problems and observe failures. In the analysis and diagnosis against the testing results, they try to get the location of the root causes of the observed failures. Usually, deep root-cause performance analysis requires correlation of test results from the client side and traces/events information from the server side.

Many performance-testing tools [4] [3] can measure client-side response time and correlated it with collected application transaction monitoring data from the server side. Application Response Measurement (ARM) [1] used by the tools provides a common way to get profiling information and allows to measure application availability, application performance, and end-to-end transaction response time.

For non ARM-instrumented servers running Java applications, Java Virtual Machine Tool Interface (JVMTI) [2] feature of Eclipse Test and Performance Tools Platform (TPTP) [5] allows performance-testing tool to collect full traces from the non ARM-instrumented servers. One problem is here is that the collection of full traces is expensive and slow. To reduce the cost of the collection of full traces from the servers, a selective instrumentation should be conducted. However, such an imprecise profiling mechanism poses challenges for performance analysis because it is difficult to identify the correlation between a transaction (or time-range) on the client and the profiling information at a server's JVM.

In our approach, we show that a technique to recover information that is available in full traces and necessary for performance analysis from sampled traces. With the sampled traces, we show a robust technique for matching the sampled method calls with the start and end points identified

We have developed a new tool, an Eclipse plug-in, for reading in performance tests and traces in the XML format and identifying the start and end point of a transaction or a request among the method execution on the server side. The tool helps testers to perform deep root-cause analysis by integrating with a load-testing tool and navigating the performance results.

In the evaluation, we compared the precision and recall of the set of we compared the runtime cost of the full traces with the cost of the sampled traces. The result is.

Contributions of our work are the followings:

- We provide an approach for supporting performance analysis in the absence of ARM.

- We suggest a technique for supporting performance analysis with sampled traces rather than fully collected traces.

The remainder of this paper is organized as follows. Section 2 presents our approach.

2. APPROACH

Our approach is ...

3. REFERENCES

[1] Application Response Measurement. http://www.opengroup.org/management/arm/.
[2] Java virtual machine tool interface. http://www.j2ee.me/j2se/1.5.0/docs/guide/jvmti/.
[3] HP LoadRunner. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17^8_4000_100__.

[4] IBM Rational Performance Tester. `http://www-01.ibm.com/software/awdtools/tester/performance`.

[5] Test and Performance Tools Platform.
`www.eclipse.org/tptp/`.