Whole-Stack Analysis and Optimization of Commercial Workloads on Server Systems

C R Attanasio[1], Jong-Deok Choi[1]*, Niteesh Dubey[1], K Ekanadham[1], Manish Gupta[1], Tatsushi Inagaki[2], Kazuaki Ishizaki[2], Joefon Jann[1], Robert D Johnson[1], Toshio Nakatani[2], Il Park[1], Pratap Pattnaik[1], Mauricio Serrano[1], Stephen E Smith[1], Ian Steiner[1]**, and Yefim Shuf[1]

[1] IBM T. J. Watson Research Center
{dicka, jdchoi, niteesh, eknath, mgupa, joefon, robertdj, ilpark, pratap, mserrano, stesmith, isteine, yefim}@us.ibm.com
[2] IBM Tokyo Research Laboratory
{e29253, ishizaki, nakatani}@jp.ibm.com

Abstract. The evolution of the Web as an enabling tool for e-business introduces a challenge to understanding the execution behavior of large-scale middleware systems, such as J2EE [2], and their commercial workloads. This paper presents a brief description of the whole-stack analysis and optimization system – being developed at IBM Research – for commercial workloads on Websphere Application Server (WAS) [5] – IBM's implementation of J2EE – running on IBM's pSeries [4] and zSeries[3] server systems.

1 Introduction

Understanding the execution behavior of a software or hardware system is crucial for improving its execution performance (i.e., optimization or performance tuning). The evolution of the Web as an enabling tool for e-business introduces a challenge to understanding the execution behavior of large-scale middleware systems, such as J2EE [2], and their applications.

J2EE is a collection of Java interfaces and classes for business applications. J2EE implementations provide a container that hosts Enterprise JavaBeans (EJBs), from which J2EE applications are constructed. The J2EE container is like an operating system for EJBs, providing services such as database access and messaging, as well as managing resources like threads and memory.

J2EE is a Java application running on a Java Virtual Machine (JVM). The JVM in turn is like an operating system for J2EE and its applications, providing services such as synchronizations and memory management. JVM, typically written in C or C++, is an application of the underlying operating system,

* Contact author
** Also at Univ. of Illinois at Urbana-Champaign

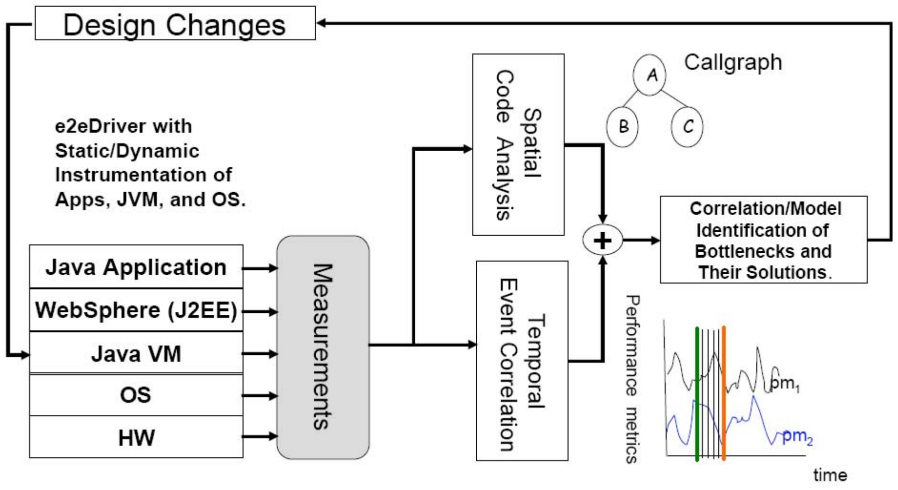
Design Changes
e2eDriver with Static/Dynamic Instrumentation of Apps, JVM, and OS.
Java Application
WebSphere (J2EE)
Java VM
OS
HW
Measurements
Spatial Code Analysis
Temporal Event Correlation
Callgraph
A
B
C
+
Correlation/Model Identification of Bottlenecks and Their Solutions.
Performance metrics
pm1
pm2
time

Fig. 1. Performance Analysis and Optimization Methodology

which itself is a client of the underlying hardware. The complicated interactions between the various layers of the software stack, from the J2EE applications to J2EE to JVM and to the operating system, and the underlying hardware layer, is a major source of the challenge to understanding and optimizing the performance of large-scale middleware systems and their applications.

This paper presents a brief description of the whole-stack analysis and optimization system – being developed at IBM Research – for commercial workloads on Websphere Application Server (WAS) running on IBM's pSeries [4] and zSeries[3] server systems. WAS is IBM's implementation of the J2EE middleware.

2 Whole-Stack Analysis and Optimization System

Figure 1 shows the performance analysis and optimization methodology of the system. The methodology provides means for:

1. instrumenting the software layers to collect statistics about software events at source level,
2. instrumenting hardware to collect statistics about various hardware events, and more importantly
3. correlating these two so that one can see the hardware events that correspond to a software event, and vice versa.

This enables the detection of hot-spots at either level and initiate generation of corresponding events at the other level.

The system employs static and dynamic instrumentation of the whole software stack, and generates trace of various software and hardware runtime events

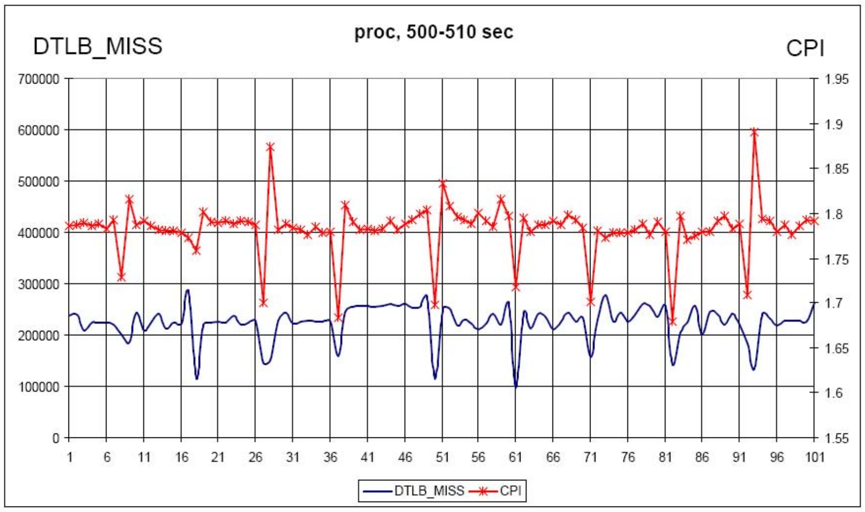

Fig. 2. Correlation between Data TLB (DTLB) Misses and CPI

both for online reduction and offline evaluation. Two major analyses are then applied to the trace: the *spatial code analysis* and the *temporal event correlation*.

The spatial code analysis identifies the static code body whose execution behavior is of interest; it identifies methods or basic blocks in a method that incur high execution overhead. A major tool of this analysis is the static and dynamic call graph and the call tree. The temporal event correlation correlates runtime events and performance metrics from the software and the hardware layers that are of interests; it identifies the relationship among various runtime events, such as Java garbage collection or thread synchronizations, and performance metrics such as *cycles per instruction* (CPI). Figure 2 shows an example performance metric trace that exhibits the correlation between CPI and the data TLB misses. The trace is generated by IBM's *hardware performance monitor (hpm)* tool kits [1]. Figure 3 shows the system layers, and examples of the performance metrics available for each layer.

The results from the spatial code analysis and the temporal event correlation are combined to identify static code bodies responsible for relatively poor performance, and potential remedies for unsatisfactory performance. Based on these findings, the system is redesigned and modified, and the cycle of performance analysis and optimization repeats until the performance becomes satisfactory.

3 Conclusion

We have presented a brief description of the whole-stack analysis and optimization system, being developed at IBM Research, for commercial workloads on

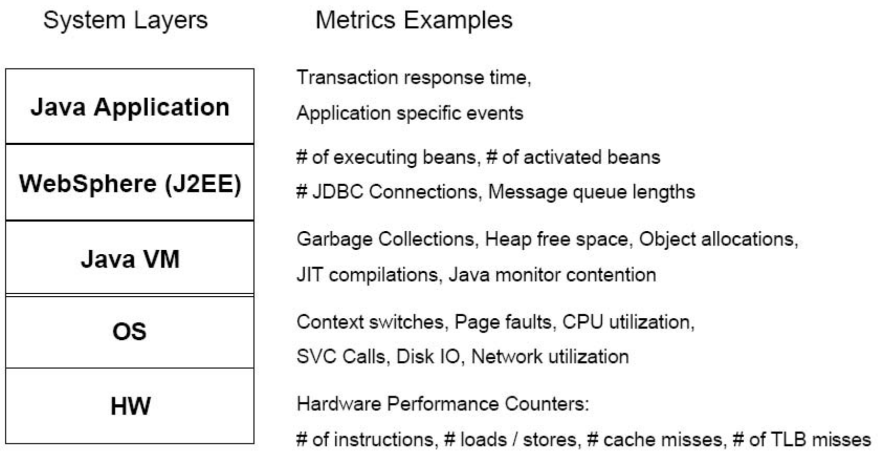
System Layers
Metrics Examples
Java Application
Transaction response time,
Application specific events
WebSphere (J2EE)
# of executing beans, # of activated beans
# JDBC Connections, Message queue lengths
Java VM
Garbage Collections, Heap free space, Object allocations,
JIT compilations, Java monitor contention
OS
Context switches, Page faults, CPU utilization,
SVC Calls, Disk IO, Network utilization
HW
Hardware Performance Counters:
# of instructions, # loads / stores, # cache misses, # of TLB misses

Fig. 3. System Layers and Performance Metrics

Websphere Application Server (WAS) running on IBM's pSeries [4] and zSeries[3] server systems. Based on the analysis results we have obtained so far, we are currently experimenting with several optimizations at the various software stack layers, from WAS to the operating system, and also at the hardware. We will publish the results of these optimizations in the future.

References

1. HPM Tool Kit. http://www.alphaworks.ibm.com/tech/hpmtoolkit.
2. Java 2 Platform, Enterprise Edition (J2EE). http://java.sun.com/j2ee.
3. Mainframe servers: zSeries. http://www-1.ibm.com/servers/eserver/zseries.
4. IBM pSeries Information Center. http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base.
5. WebSphere Application Server. http://www.ibm.com/websphere.