DejaVu: Deterministic Java Replay Debugger for Jalapeño Java Virtual Machine

Bowen Alpern
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
alpern@watson.ibm.com

Jong-Deok Choi
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
jdchoi@watson.ibm.com

Ton Ngo
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
ton@us.ibm.com

Manu Sridharan
Dept. of Electrical Engineering and Computer Science
MIT
Cambridge, MA 02139
msridhar@mit.edu

ABSTRACT

The execution behavior of a Java application can be non-deterministic due to multithreading. This non-determinism makes understanding and debugging multithreaded Java applications a difficult and laborious process. DejaVu (Deterministic Java Replay Utility) helps the user in understanding and debugging non-deterministic Java applications by deterministically replaying the execution behavior of a non-deterministic execution. In this demo, we will present a debugger for the Jalapeño Java Virtual Machine that utilizes the replay capability provided by DejaVu. The debugger helps in isolating non-deterministic failure(s) by faithfully reproducing the same execution behavior that led to the observed failure(s). Jalapeno is a JVM being developed at IBM T. J. Watson Research Center. The debugger provides the following features: (1) DejaVu deterministically replays Java programs; (2) Remote Reflection supports general debugging functionalities such as setting breakpoints, examining the program state, and detecting deadlocks; and (3) a GUI provides an intuitive and easy to use interface.

1. INTRODUCTION

Servers differ markedly from client applications in several respects. Servers are usually longer-running, more highly threaded, and have greater scalability concerns. Servers must respond to requests quickly, often within explicit time limits, even though the number of clients and requests may be impossible to predict. These properties conspire to make server programming difficult.

Several features of Java make it attractive for server development, including garbage collection and a simple but effective threading model. Yet server development remains troublesome because of the challenges of debugging multithreaded code in general and server code in particular. Servers are difficult to implement correctly because of the sheer scale of multithreading compared to client applications. For instance, the average GUI employs just a few threads, while a server can spawn dozens under even moderate load. Add to this the comparatively low observability of server behavior versus a GUI, plus the nondeterminism that usually comes with multithreading, and the debugging task becomes daunting. Nondeterminism is arguably the most challenging problem because it is difficult to even repeat the failure to debug.

We present the integration of several components developed at the IBM T. J. Research Center: the Jalapeño Java Virtual Machine, the DejaVu deterministic replay, and Remote Reflection for perturbation-free debugging. The result is a powerful platform for both developing multithreaded programs such as server applications and delivering the performance they require. The debugger includes a GUI based on Java Swing and provides both low level and high level symbolic debugging.

2. Jalapeño JAVA VIRTUAL MACHINE

At the foundation of this platform is Jalapeño [2, 1], a compile-only Java Virtual Machine (JVM) for high performance servers. Written in Java, Jalapeño brings the benefits of the Java object model to server design and implementation without sacrificing performance. Blurring the boundary between application code and runtime services leads to several advantages. First, application code can call runtime methods directly without any overhead to adapt to the stack and calling conventions of a different language. Secondly, Jalapeño optimizing compiler is free to inline runtime services directly into application methods. Such inlining eliminates a method call overhead and may expose opportunities for further optimization. Finally, the same dynamic optimizations that the optimizing compiler applies to applica-

OOPSLA 2000 Companion Minneapolis, Minnesota

tion code can be applied to the optimizing compiler itself.

3. DEJAVU

DejaVu [3, 4] offers a solution to the problem of debugging in the face of nondeterminism. It provides deterministic replay of a nondeterministic execution sequence, allowing a program to be reexecuted repeatably. Thus a bug may be isolated quickly, and its causes and effects observed reliably, without disappearing in one or more executions due to nondeterminism.

DejaVu runs in two modes: the *Record mode* captures key information for replay, while the *Replay mode* deterministically replays execution using the captured information. Because the current version runs on a uniprocessor where only one thread can run at a time, preserving execution order is a matter of capturing thread scheduling information during recording and enforcing the same thread schedule during replay.

Jalapeño employs a quasi-preemptive thread scheduling driven by yield-points inserted by the compiler and timer interrupt. DejaVu in Jalapeño utilizes this quasi-preemptive thread scheduling to efficiently capture and replay thread scheduling. For events that are not reproducible, such as reading a wall-clock value or keyboard inputs, DejaVu captures their values during record mode and reuses them during replay mode.

4. REMOTE REFLECTION

A debugger integrated with DejaVu must be able to start and stop both the application and the JVM as they replay their execution deterministically. In addition, the debugger must provide the usual debugging functions such as querying objects and program state, setting breakpoints, etc. Jalapeno provides an extensive reflection interface so that the system components can be integrated seamlessly and effectively. As a result, it is desirable for a debugger to exploit the same reflection interface instead of using a different ad hoc interface. Unfortunately, this approach leads to conflicting requirements between DejaVu and the debugger: running the debugger within Jalapeño would alter the system state and perturb the deterministic replay, yet running outside Jalapeño would prevent the debugger from using the Jalapeño internal reflection interface.

Remote reflection [5] solves this problem by allowing a program in a JVM, e.g. a debugger, to execute a reflection method that operates directly on an object residing in another JVM, e.g Jalapeño and the application being replayed. The key to remote reflection is a proxy object in the local JVM called the *Remote Object* which represents the real object in the remote JVM.

5. STATUS

We have completed an initial integrated system that includes all the components above. With this system, we have been able to record and replay PBob, a significant Java benchmark with a 20 minutes execution. During replay, the debugger can stop the program at arbitrary points to inspect the entire system, including the application and Jalapeño itself.

6. REFERENCES

[1] Bowen Alpern, Dick Attanasio, John J. Barton, Michael G. Burke, Perry Cheng, Jong-Deok Choi, Anthony Cocchi, Stephen Fink, David Grove, Michael Hind, Susan Flynn Hummel, Derek Lieber, Vassily Litvinov, Ton Ngo, Mark Mergen, Vivek Sarkar, Mauricio J. Serrano, Janice Shepherd, Stephen Smith, V. C. Sreedhar, Harini Srinivasan, and John Whaley. The Jalapeño Virtual Machine. *IBM Systems Journal*, 39(1), pages 211–238, 2000.

[2] Bowen Alpern, Dick Attanasio, John J. Barton, Anthony Cocchi, Susan Flynn Hummel, Derek Lieber, Ton Ngo, Mark Mergen, Janice Shepherd, and Stephen Smith. Implementing Jalapeño in Java. *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, pages 314–324, November 1999.

[3] Jong-Deok Choi and Harini Srinivasan. Deterministic Replay of Java Multithreaded Applications. *ACM SIGMETRICS Symposium on Parallel and Distributed Tools*, pages 48–59, August 1998.

[4] Ravi Konuru, Harini Srinivasan, and Jong-Deok Choi. Deterministic Replay of Distributed Java Applications. *14th International Parallel & Distributed Processing Symposium*, pages 219–228, May 2000.

[5] Ton Ngo and John Barton. Debugging by Remote Reflection. *European Conference on Parallel Computing*, August 2000.