Bill Pugh's Java Research and Resources

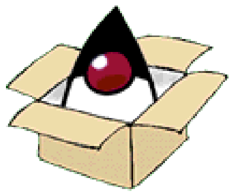
- Information on Java
- Compressing Java classfiles
- Fixing the Java Memory Model
- Secure Inner Classes

Information on Java

I have collected a number of resources on Java:

- Learning Java
- Research papers on Java, particularly on compiler, virtual machine and run-time issues.
- tools (such as bytecode instrumentation packages) that are useful to Java research projects.

Compressing Java class files



Java class files are often distributed as `jar` files, which are collections of individually compressed class files (and possibility other files). Jar files are typically about 1/2 the size of the original class files due to compression. I have developed a wire-code format for collections of Java class files. This format is typically 1/2 to 1/5 of the size of the corresponding compressed jar file (1/4 to 1/10 the size of the original class files).

A paper describing this work appeared at the ACM Conference on Programming Language Design and Implementation 1999.

Slides and handouts from my talk are available, as is an HTML version of the paper (although the PDF version is the definitive version).

An pre-alpha version of the software (version 0.8.0) *is now available for download*. You can read the (short) README file

Fixing the Java Memory Model

The Java memory model described in Chapter 17 of the Java Language Specification gives constraints on how threads interact through memory. The Java memory model is hard to interpret and poorly understood; it imposes constraints that prohibit common compiler optimizations and are expensive to implement on existing hardware. These issues are particularly important for high-performance Java applications, since they are more likely to use and need aggressive optimizing compilers and parallel processors.

In addition, programming idioms used by some programmers and used within Sun's Java Development Kit is not guaranteed to be valid according the existing Java memory model.

A paper I've written reviews these issues and suggests replacement memory models for Java. The paper appeared at ACM Java Grande conference.

The slides from my talk are available.

I've written a program that tests if a JVM violates Coherence, a property required by the Java Memory Model. So far, I've found that Coherence is violated by Sun's 1.1.8, 1.2.1 Classic and 1.2.1 Hotspot VM's for Wintel, and Sun's 1.2.1 production VM for Sparc Solaris and Microsoft's VM. Other VM's I've tested (Sun's 1.2.1 reference and 1.1.8 for Sparc Solaris) don't do the optimization (eliminating redundant loads) even when it is legal. I've not yet found any compilers that eliminate loads only in the cases allowed by the Java memory model. This problem is now listed as Bug 4242244 in Sun's Java Bug Parade.

Dan Scales, of Digital Western Research Laboratories, has looked at the impact of Coherence on compiler optimizations. His initial examination showed that while many benchmarks were essentially uneffected, other benchmarks were slowed down by factors of 1.11 to 1.43.

A separate page collects pointers to resources discussing issues related to the memory model.

Fixing the inner class security hole

Innerclasses were added to Java in version 1.1, in a way that was compatible with version 1.0 Java virtual machines. Given two classes A and B, if one is declared inside the other, or if both are declared inside another class, then A and B should have access to each other's private fields and methods. Since the Java virtual machine doesn't understand innerclasses, the compiler produces a modified program in which the *private* protection of the fields and methods has been essentially eliminated.

Working with Anasua Bhowmik, I have developed a technique for closing this security hole. The technique we have developed could be implemented within a compiler, but we have implemented a tool that takes java classfiles compiled using the standard, insecure technique and automatically closes the innerclass security hole.

This work was described at a PLDI99 poster session. Slides and a handout are available in PDF format.

Research described here has been supported by the National Science Foundation under grants CCR9619808 and ASC9720199. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).