Presentations

This page contains slides from stand-alone presentations. Some of the Publications also have presentations associated with them.

Jikes RVM

- Author: Ian Rogers
- Conference: Free and Open Source Developers' European Meeting
- Abstract:
  - A short presentation on the Jikes RVM and related research in the context of open source Java.
- Download: PowerPoint (5.8MB)

Dynamic Compilation and Adaptive Optimization in Virtual Machines

- Authors: "Matthew Arnold, Stephen Fink, David Grove, and Michael Hind"
- Instructor: Michael Hind
- Conference: ACACES'06, July 24-28, 2006 (This is an updated version of the PLDI'04 tutorial)
- Abstract:
  - The past decade has witnessed the widespread adoption of programming languages designed to execute on virtual machines, such as the Java and C# programming language. However, such languages face significant performance challenges beyond those confronted by traditional languages. First, portable program representations and dynamic language features, force the deferral of most optimizations until runtime, inducing runtime optimization overhead. Second, modular program representations preclude many forms of whole-program inter-procedural optimization. Third, virtual machines incur additional costs for runtime services such as security guarantees and automatic memory management. To address these challenges, mainstream virtual machine implementations include substantial infrastructure for online profiling, dynamic compilation, and feedback-directed optimization. As a result, adaptive optimization has begun to mature as a widespread production-level technology.
  - This course will survey the state-of-the-art in the areas of dynamic compilation and adaptive optimization in virtual machines. Dynamic compilation is the process of dynamically optimizing a portable representation, such as Java byte codes, into native code. Adaptive optimization is the online decision process that determines the overall strategy for profiling and employing dynamic compilation. In addition to surveying the state-of-the-art, and highlighting the many topics for open research in this area, this course will also debunk several misconceptions about these two topics, such as
    - Dynamic compilers must be blazingly fast.
    - Dynamic class loading is a fundamental roadblock to cross-method optimization.
    - A static compiler can always get better performance than a dynamic compiler.
    - Sophisticated profiling is too expensive to perform online.
    - Program optimization is a dead field.
    - Lack of a suitable infrastructure is a significant barrier to research in this area.
- Download: PDF (3.3MB)

MMTk: The Memory Management Toolkit

- Presentors: "Steve Blackburn and Perry Cheng"
- Conference: ISMM'04, October 23, 2004 (ISMM'04 Tutorial)
- Abstract: MMTk is a framework for construction of garbage collectors within the open-source Jikes Research Virtual Machine. The tutorial was given by two of its creators.
- Download: PDF (325KB)

Jikes RVM Optimizing Compiler Intermediate Code Representation

- Presentors: Shane Brewer
- Event: Group presentation at University of Alberta, March 20, 2003.
- Download: PowerPoint (280K) or PDF (247K)

The Design and Implementation of the Jikes RVM Optmizing Compiler

- Presentors: "David Grove and Michael Hind"
- Conference: OOPSLA '02, November 5, 2002 (OOPSLA '02 Tutorial)
- Abstract:
  - The Jikes Research Virtual Machine (RVM) is a software infrastructure designed to execute Java programs typically used in programming language implementation research. The Jikes RVM is available as an open source project. The Jikes RVM

provides the academic and research communities with a flexible open testbed to prototype new virtual machine technologies and experiment with various design alternatives. A large number of academic research groups have already adopted it. It runs on AIX, Linux, and OS X platforms and demonstrates industrial strength performance for many benchmark programs. The Jikes RVM includes state-of-the-art technologies for dynamic compilation, adaptive optimization, garbage collection, thread scheduling, and synchronization.
    - This tutorial presents an overview of the Jikes RVM optimizing compiler. The first part of the tutorial covers the structure of the compiler, focusing on the requirements, goals, and design of its intermediate representation (IR). The second part of the tutorial covers some of the compiler's extensive set of analyses and optimizations, ranging from simple local analyzes to SSA-based flow-sensitive optimizations with type-based heap analysis. The last part covers the integration of the optimizing compiler into the adaptive optimization system, focusing on instrumentation compilation and feedback-directed optimizations.
    - Specific issues to be covered include: optimizing the memory model, precise exceptions, and dynamic class loading; compiler requirements for runtime support of garbage collection maps, scheduling (yield points), exception tables, line number information; compiler/runtime system cooperation for fast object allocation and runtime services (for example dynamic type checking or invokeinterface); compiler structure for multiple platforms; tracing an interesting method (for example, one of our enumeration loops from the compiler) through key optimizations to illustrate how type analysis, inlining, scalar replacement, plus a set of traditional optimizations work together; etc.
- Download: PDF (4,250KB)

The Design and Implementation of the Jikes RVM Optmizing Compiler

- Presentors: "Stephen Fink and Michael Hind"
- Conference: PLDI '02, June 16, 2002 (PLDI '02 Tutorial)
- Abstract:
    - The Jikes Research Virtual Machine (RVM) is a software infrastructure designed to execute Java programs typically used in programming language implementation research. The system runs on the AIX™/PowerPC™ Linux ®/PowerPC, and Linux/IA-32 platforms and demonstrates industrial strength performance for many benchmark programs. Many academic groups have adopted the Jikes RVM as their primary research infrastructure, resulting in publications in leading conferences such as PLDI '02 and ISMM '02. In the first six months since its open source release the VM has been downloaded by over 1000 different IP addresses, including over 70 universities around the world.
    - This two-hour tutorial presents an overview of the Jikes RVM optimizing compiler. The first part of the tutorial covers the structure of the compiler, focusing on the requirements, goals, and design of its intermediate representation (IR). The second part of the tutorial covers some of the compiler's extensive set of analyses and optimizations, ranging from simple local analyses to SSA-based flow-sensitive optimizations with type-based heap analysis. Time permitting, we will highlight the integration of the optimizing compiler into the adaptive optimization system.
- Download: gzip-compressed PostScript (1,055 KB)

The Design and Implementation of the Jalapeño Research VM for Java

- Presentors: "Dick Attanasio and Michael Hind"
- Conference: International Conference on Parallel Architectures and Compilation Techniques, September 9, 2001 (PACT01 Tutorial)
- Abstract:
    - The design and implementation of a high-performing Java virtual machine is a formidable task. Over the past three years IBM researchers have developed the Jalapeno research virtual machine for Java as a high-performing server platform. To address the requirements of servers, performance and scalability in particular, Jalapeño was designed from scratch to be as self-sufficient as possible.
    - Key design features of Jalapeño include
        - the entire VM is implemented in the Java programming language,
        - a flexible online adaptive optimization infrastructure, implemented as concurrent Java threads, utilizes two compilers and no interpreter guided by a cost-benefit model
        - a family of parallel, type-exact garbage collectors,
        - a lightweight thread package with compiler-supported preemption, and
        - an aggressive optimizing compiler with multiplel optimization levels.
    - This full-day tutorial will share the lessons learned from the design and implementation of Jalapeño. Design decisions will be contrasted to other Java implementations.
- Download:
    - All 12 sections (ZIP-compressed PostScript) (2,319K)
    - Individual sections, all in gzip-compressed PostScript:
        - 1. Introduction (92K)
        - 2. VM (86K) "Corresponding papers in IBM Systems Journal and OOPSLA '99"
        - 3. Exception handling (51K)
        - 4. Dynamic Type Checking (126K) "Corresponding paper in JVM '01"
        - 5. Interface invocation (216K) "Corresponding paper in OOPSLA '01"
        - 6. Threading and Synchronization (35K) "Corresponding paper in IBM Systems Journal"
        - 7. JNI support (58K)
        - 8. Memory Management (126K) "Corresponding paper in LCPC '01"
        - 9. Optimizing Compiler Overview (240K) "Corresponding paper in Java&#153 Grande '99"
        - 10. Adaptive Optimization System (540K) "Corresponding paper in OOPSLA '00"
        - 11. DejaVu (6817K)

- 12. References (86K)

Optimizing Compilation of Java Programs

- (Revised version of PLDI tutorial)
- Presentor: "Vivek Sarkar"
- Conference: ACM Java Grande, June 2001 (Java Grande Tutorial)
- Download: gzip-compressed PostScript 186K

Optimizing Compilation of Java Programs

- Presentor: "Vivek Sarkar"
- Conference: ACM Conference on Programming Languages Design and Implementation, June 2000
- Download: gzip-compressed PostScript 258K

Issues in High-Performance Programming in Java

- Authors: "Bowen Alpern and Susan Flynn-Hummel"
- Conference: High Performance Computing System, June 13th, 1999 (HPCS '99 Tutorial)
- Abstract:
    - Java is a modern object-oriented type-safe programming language with automatic memory management. These features make it attractive to anyone responsible for developing and maintaining a large software project. Its multithreading support is a further inducement to those anxious to exploit parallel processing hardware. To date, however, there has been a performance penulty for using the language. For the vast majority of programming efforts, this is marginal concern at most. But, for performance programming, any degradation in performance is a potential show stopper.
    - Java performance has improved markedly in the last year or two. The advent of a second generation of Just-In-Time (JIT) compiler technology promises to further close the performance gap between Java and traditional languages for most applications. This raises the question: will Java ever be the language of choice for performance-critical applications?
    - Unfortunately, the jury is still out. The Java Grande forum has been established to address the problems with using Java for computationally intensive science and engineering applications. Under its auspices, a number of interesting projects are underway. Proposals for improving Java performance run a bewildering gamut from idioms to be used by the programmer, through bytecode and JIT optimization techniques, to major language changes.
    - This tutorial will systematically identify and classify the aspects of Java that are problematic from the performance point of view. It will then chart the terrain upon which such issues might be addressed. Finally, it will survey approaches to improving Java performance.
- Download: PostScript file Part 1 (764K), Part 2 (656K)
- Link to: presentation (html)