

Realizing a Java Virtual Machine with SEAM

Design Presentation

Patrick Cernko

cernko@ps.uni-sb.de

9. September 2004

Programming Systems Lab,
Saarland University

Motivation

- SEAM: Generic VM framework
- Java Virtual Machine
 - ◆ Published Specification [Lindholm and Yellin, 1999]
 - ◆ Wide spread, object-oriented, real-world VM
- Question to answer
 - ◆ SEAM sensible for OO languages
 - ◆ Efficiency/Overhead to comparable JVMs
- Approach
 - ◆ Reuse Infrastructure from Reference-VM
 - ◆ Measure against Reference-VM

- VM Core
 - ◆ Abstract data store
 - Garbage collector
 - ◆ Generic concurrent execution model
 - “Worker” abstraction
- Language Layer
 - ◆ Language data modeled on top of store
 - ◆ Model code execution using “Workers”

- Objects have Properties
(= “abstract fields” [Bacon et al., 2002])
 - ◆ Locks
 - ◆ Hash value
- Code unit: `.class` file
 - ◆ Provides one Java class
 - ◆ Loaded by need at runtime by *ClassLoader*
 - ◆ Bytecode
- Bytecode execution engine
 - ◆ Executes methods of Java classes

Data Model

- Scalars
 - ◆ int stored as 31Bit value in store
 - ◆ long, float and double must be boxed
- Java Objects
 - ◆ Simple (heavy-weight) one-to-one mapping of fields, e.g. Lock & Hash
 - ◆ Refinements possible
(think-locks [Bacon et al., 1998], indirection)
- Data Model taken from existing prototype

ClassLoader

- Loads `.class` files
 - resolves symbolic references
 - ◆ to other classes and interfaces
 - ◆ to static strings and numbers
 - Methods for creating scalars and arrays
- reuse of `ClassLoader` from existing prototype
(straightforward implementation)

Execution Engine

- BytecodeInterpreter from Kaffe-VM
 - ◆ Well structured and documented
 - ◆ Switch-based single opcode execution
 - ◆ Easy to integrate
 - Well factored accessors
 - ⇒ Own implementation of accessors
- Outlook:
 - ◆ Kaffe provides JIT
 - ◆ Kaffe-JIT shares much infrastructure with interpreter
 - ⇒ JIT possible as optional second step

Timeline

- Milestone 1: Running SEAM-JVM prototype using new interpreter (15.03.2004)
- Milestone 2: enhancing Data Model (15.05.2004)
 - ◆ incremental
 - ◆ combined with writing preliminary thesis
- Milestone 3: Writing final thesis (30.06.2004)
- Milestone 4: Revision of the thesis
- Milestone 5: Final presentation (end of august)

References

- [Bacon et al., 2002] Bacon, D. F., Fink, S. J., and Grove, D. (2002). Space- and Time-Efficient Implementation of the Java Object Model. In Magnusson, B., editor, *Proceedings of the Sixteenth European Conference on Object-Oriented Programming (ECOOP 2002)*, volume 2374 of *Lecture Notes in Computer Science*, pages 111–132, Málaga, Spanien. Springer-Verlag.
- [Bacon et al., 1998] Bacon, D. F., Konuru, R., Murthy, C., and Serrano, M. (1998). Thin Locks: Featherweight Synchronization for Java. In *Proceedings of the ACM SIGPLAN '98, Conference on Programming Language Design and Implementation (PLDI)*, pages 258–268. ACM Press.
- [Brunklaus and Kornstaedt, 2002] Brunklaus, T. and Kornstaedt, L. (2002). A Virtual Machine for Multi-Language Execution. Technical report, Programming Systems Lab. <http://www.ps.uni-sb.de/Papers/abstracts/multivm.pdf> (August 2004).
- [Lindholm and Yellin, 1999] Lindholm, T. and Yellin, F. (1999). *The JavaTM Virtual Machine Specification*. Addison Wesley, 2 edition.