Realizing a Java Virtual Machine with SEAM

Final Presentation

Patrick Cernko
cernko@ps.uni-sb.de

September 16, 2004
Programming Systems Lab,
Saarland University
Motivation

- Java Virtual Machine
  - Wide spread, object-oriented VM
  - Published specification [Lindholm and Yellin, 1999]
- SEAM [Brunklaus and Kornstaedt, 2002]
  - Pluggable components & generic services to build VMs
- Existing JVM prototype
  - Naive prototype implementation using SEAM
- Questions to answer
  - Efficiency/Overhead to comparable JVMs
Approach

- Refinement of existing prototype
  - Analysis of object model
  - Analysis of bytecode execution
- Reuse components from other JVMs
- Evaluation of refined JVM using standard benchmarks
Core

- Abstract data store
  - Garbage collector
- Generic concurrent execution model
  - Evaluator abstraction

Language layer

- Language data modeled on top of store
- Language services modeled using evaluators
JVM-SEAM: Architecture

- ClassLoader
- Base libraries
- Code and object store
- GARBAGE COLLECTOR
- Store
- Scheduler
- Evaluator
- Execution
JVM-SEAM: Components

- Base-Libraries
  - Completely reused from Prototype

- ClassLoader
  - Mostly reused from Prototype

- Store
  - Code and object store: JVM data model on top of SEAM Store

- Bytecode execution
  - Evaluator: Implemented as interpreter using engine from JVM kaffe
Execution Engine

Bytecode Interpreter from Kaffe-VM

- Well structured and documented
- Mapping
  - Bytecode → Micro-Language → accessors
  - Own implementation of accessors
- Some accessors didn’t match JVM-SEAM model
  - Small changes in kaffe’s code-base required
  & workarounds in Accessor-Implementations
## Evaluation

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>JVM-SEAM</th>
<th>Prototype</th>
<th>kaffe</th>
</tr>
</thead>
<tbody>
<tr>
<td>fib</td>
<td>1.9⁻¹</td>
<td>1.8⁻¹</td>
<td>1.0</td>
</tr>
<tr>
<td>tak</td>
<td>2.3⁻¹</td>
<td>2.1⁻¹</td>
<td>1.0</td>
</tr>
<tr>
<td>nrev</td>
<td>3.6⁻¹</td>
<td>4.6⁻¹</td>
<td>1.0</td>
</tr>
<tr>
<td>quickarray</td>
<td>1.5⁻¹</td>
<td>1.4⁻¹</td>
<td>1.0</td>
</tr>
<tr>
<td>queens</td>
<td>2.3⁻¹</td>
<td>2.4⁻¹</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Speed normalized to kaffeVM

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>JVM-SEAM</th>
<th>Prototype</th>
<th>HotSpot™</th>
</tr>
</thead>
<tbody>
<tr>
<td>fib</td>
<td>4.7</td>
<td>4.9</td>
<td>1.0</td>
</tr>
<tr>
<td>tak</td>
<td>4.3</td>
<td>4.5</td>
<td>1.0</td>
</tr>
<tr>
<td>nrev</td>
<td>4.7</td>
<td>3.7</td>
<td>1.0</td>
</tr>
<tr>
<td>quickarray</td>
<td>3.6</td>
<td>4.0</td>
<td>1.0</td>
</tr>
<tr>
<td>queens</td>
<td>3.2</td>
<td>3.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Speed normalized to HotSpot™
Discussion

- JVM-SEAM beats Prototype in *fib*, *tak* & *quickarray* (integer arithmetics)
- Prototype still faster in *nrev* & *queens* (method invocation) → but kaffe performs uneven slower! ➞ kaffe lacks performance in method invocation & object creation
- JVM-SEAM beats kaffe with the same interpreter with twice up to three times speed!
Summary

- Integration of kaffe-interpretation in JVM-SEAM
- Refinement of prototype partly successful
- JVM-SEAM beats kaffe

⇒ SEAM is usable for JVM implementations
Future Work

- Object model layout
  - improved concurrency locking [Onodera and Kawachiya, 1999]
  - Space and time efficiency [Bacon et al., 2002]

- JIT compiler
  - kaffe JITs [kaffe, 2004]
  - Jalapeño compilers [Arnold et al., 2000]


