Java for High Performance Computing

Jordi Garcia Almiñana

14 de Octubre de 1998
de la era post-internet

Index of the Talk

- The Java Programming Language
  - Java for Scientific Computation
  - Java limitations for Scientific Computation
- The Java Runtime System
  - The Java Virtual Machine environment
  - Alternatives to the basic JVM specification
- Java for High Performance Computing
  - Definition of HPC language extensions for Java
  - Java restructurers, compilers, and optimizers
  - Parallel systems for High Performance Java

The Java Programming Languaje

- Features of Java
  - Object-Oriented environment
  - Platform independence
  - Network support
  - Security
    - Runtime environment
    - Secure remote transactions
  - Multithreading support
  - Automatic memory management
  - Fault tolerance through exception handling

The Java Programming Languaje

- Java for Scientific Computation
  - OO programming is preferable for design and maintainability
    - Data type safety
    - Automatic memory management (garbage collection)
  - Java arithmetic is fully protable
    (in other languages, loss of precission in `float` temporary expressions can give different results)
  - Truly portable threads and synchronization model
  - Portable libraries for distributed programming
  - GUI, programmer force, availability, ...

The Java Programming Languaje

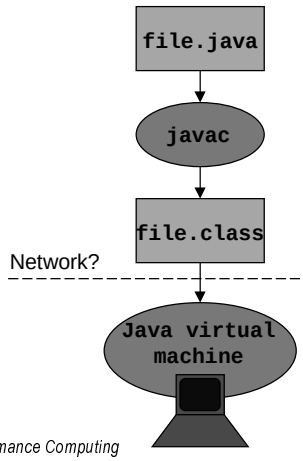
- Java Limitations for Scientific Computation
  - Java´s floating point restricted specification
    - Limits exploitation of current and future chips architecture
  - No support to complex numbers
  - Internal implementation of arrays
    - Non-consecutive memory layout scheme
    - Multidimensional arrays implemented as arrays of arrays (no unrolling, loss of locality, …)
    - Array boundary checking for every single array access
  - The Java run-time exception mechanism

The Java Programming Languaje

- Java Limitations for Scientific Computation
  - No support to high degrees of parallelism
    - Parallel programming only through threads mechanism
  - No facility for explicit locality management
  - Other compiler related issues
    - Unavailability of the complete program at compile time
    - The exception mechanism limits code movement
    - High level abstraction of JVM instruction set hides many machine dependent optimization opportunities

The Java Runtime System

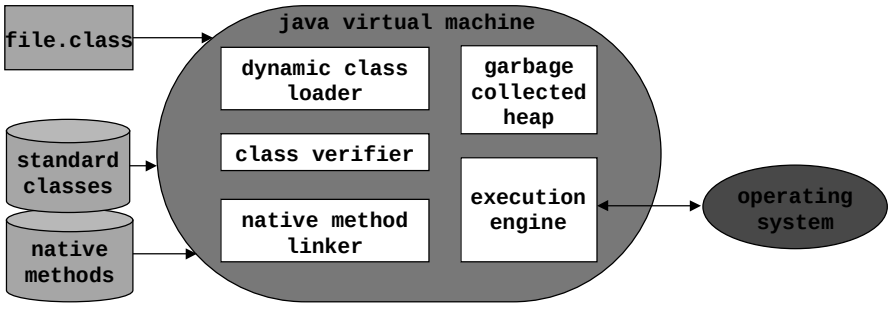
- Java Virtual Machine: bytecode interpreter



- Java source code application
- Java to bytecode translation (performed only once)
- Java bytecode class file (JVM instruction set)
- Java Virtual Machine Specification
  - format of class files
  - semantics of instructions

The Java Runtime System

- Structure of the Java Virtual Machine



  - The JVM implementation can be tailored for high performance, low memory use, or portability

The Java Runtime System

- Drawbacks of JVM: Speed!
  - Java bytecode is interpreted
  - Bytecode verification rules and exception mechanism
  - High-level structure about original program is lost
  - Local view during invocation of classes
  - Stack-based operands (no registers)
- Alternatives:
  - Just-in-time code generation
  - Java static compilation to native code
  - Java OS and Java Processors

The Java Runtime System

- Just-in-time Code Generation
  - Java bytecode to native object code on-the-fly
  - Just before the method is run for the first time
  - Based on the idea of inlining the machine code
  - Certain optimizations can be performed
    - Elimination of redundant instructions
    - Use hardware registers for local variables
    - Optimization of method stack usage
  - However compilation speed requirements constrain the quality of the optimizations
  - Improvements over interpretation by a factor of 10!

The Java Runtime System

- Java Static Compilation to Native Code
  - Traditional resource-intensive optimization techniques can improve the performance of the generated code
    - instruction schedulling
    - common subexpression elimination
    - inter-module analysis
    - constant propagation
    - global register allocation
  - Run time checking can be optimized or removed
  - …but portability and security can be lost!

The Java Runtime System

- Java OS
  - Java applications run directly on hardware platforms without requiring a host OS
- Java Processors
  - Hardware implementation of a Java interpreter
  - Chips created specifically to run Java bytecode
    - Sun implementations: picoJava, microJava

Java for HPC

- Different levels of research

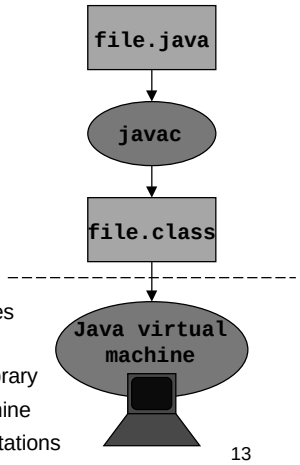
- Language extension
- Mathematical libraries

- Restructuring compiler

- Bytecode optimization

- JIT optimization techniques

- Runtime parallel kernel library
- Parallel Java Virtual Machine
- Hardware Java implementations



Java for HPC

- HPC Language Extensions for Java
  - Language extension
    - multidimensional arrays
    - data parallel distributed loops
    - communication primitives
  - Java numerical (matrix-oriented) libraries
  - Floating-point precission and complex numbers
- Other Java Dialects
  - Titanium: language and system for parallelism
  - Spar: semi-automatic parallel compilation

Java for HPC

- Java Restructurers
  - Source-to-source tools that extract parallelism (implicitly from loops or explicitly from annotations)
- Java Compilers
  - Java to parallel bytecode translation
  - Definition of IR for parallel optimization
- Bytecode Optimizers
  - Techniques for automatic detection of parallelism
  - Machine-dependent optimizations
- Just-in-time optimizations

Java for HPC

- Parallel Systems for HPC in Java
  - Parallel implementation of a JVM
    - Programming based on SPMD Java
    - Based on PVM or MPI implementations
  - Environment definition for network parallel programs
    - Heterogeneous environment (run-time systems or parallel libraries)
  - Parallel sparse computation in Java
    - Runtime environment with continuous compilation

The Survey

- Available (draft version) at URL:

file:/usr/users.coderch/ac/jordig/pub/Java4HPC/main.html