The role of Java for scientific computing

Roldan Pozo

Mathematical Software Group

National Institute of Standards and Technology

Why Java?

- Portability of the Java Virtual Machine (JVM)
- minimize memory leaks and pointer errors
- security model
- network-aware environment
- Parallel and Distributed computing
  - Threads
  - Remote Method Invocation (RMI)
- Integrated graphics
- Widely adopted
  - embedded systems, browsers, devices
  - widely adopted for teaching, development

Java code example: SOR

```
public static final void SOR(double omega, double G[][], int num_iterations)
  {
      int M = G.length;
      int N = G[0].length;

      for (int p=0; p<num_iterations; p++) {
          for (int i=1; i<M-1; i++) {
              for (int j=1; j<N-1; j++)
                  G[i][j] = omega * (G[i-1][j] + Gi[i+1][j] +
                      G[i][j-1] + G[i][j+1]) / 4.0 + (1 – omega) * Gi[j];
          }
      }
  }
```

Portability & Performance

- binary portability is Java's greatest strength
- virtual machine (JVM) bytecodes are the key
- most languages (or subsets) can generate these
  - C, C++, Fortran, BASIC, LISP, etc.
- Issue:
  - can performance be obtained at ***bytecode*** level?

Virtual Machine technologies

- IBM 360/30 OS (late 60's)
- UCSD Pascal
- PVM
- Inferno (Lucent)
- VMWare
- Microsoft .NET
- Intel x86 architectures

Why *not* Java?

- Performance
  - interpreters too slow
  - poor optimizing compilers
  - virtual machine
  - security model
    - array bounds checking

Why *not* Java?

- lack of scientific software
  - computational libraries
  - numerical interfaces
  - major effort to port from f77/C
    - no *universal* translator (f2java)

Why *not* Java?

- original language has no standard support for
  – complex data type
  – operator overloading
  – true multidimensional arrays
  – generic types (pre v. 1.5)

Java Numerics Working Group

- evaluate the suitability of Java technology for numerical applications
- voice community consensus on needed changes to language, environment
- communicate needs to developers
- catalyze development of standard APIs for core numerical functions

Performance

What are we really measuring?

- language vs. virtual machine (VM)
- Java -> bytecode translator
  - common (static) compiler optimizations
- bytecode execution (VM)
  - interpreted
  - just-in-time compilation (JIT)
  - adaptive compilers (e.g.HotSpot)
- underlying hardware

Java optimization techniques

- Native methods (JNI)
- stand-alone compliers (.java -> .exe)
- modified JVMs
  - (fused mult-adds, bypass array bounds checking)
- aggressive bytecode optimization
  - JITs, flash compilers, HotSpot
- bytecode transformers
- concurrency

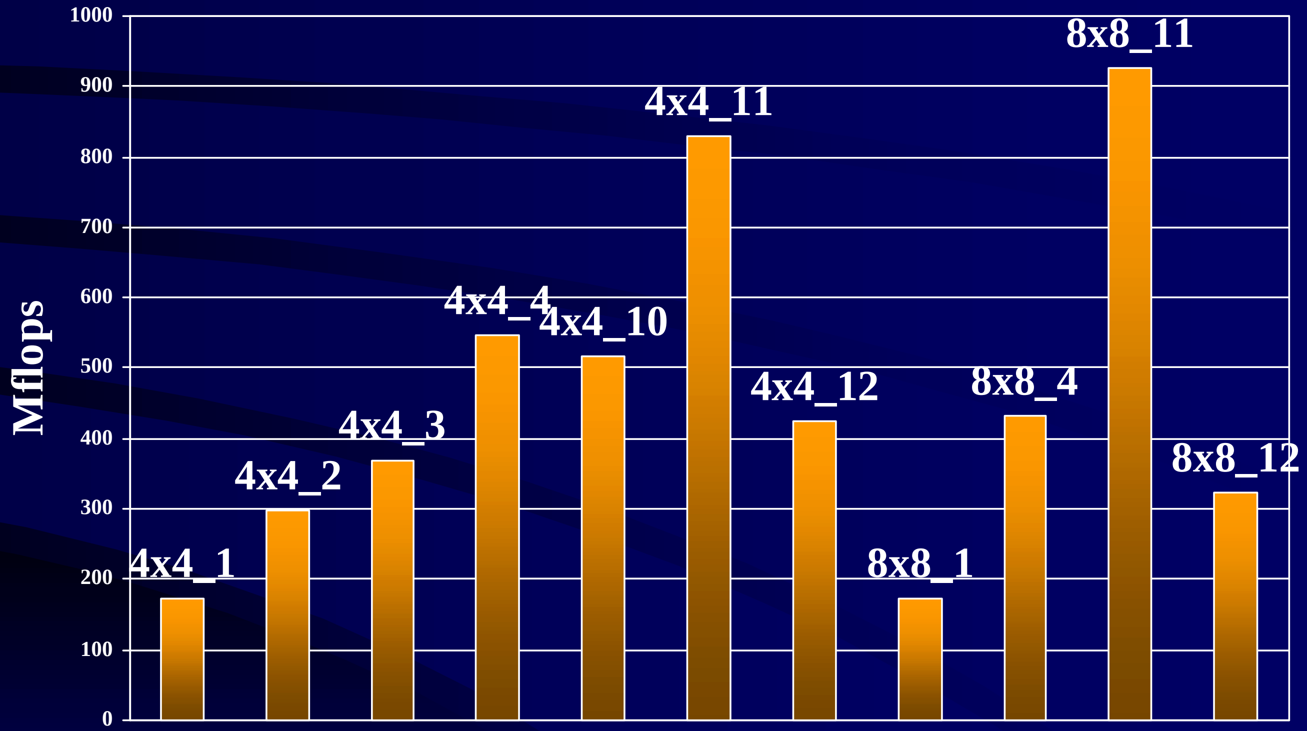
Other numeric issues

- array bounds checking
- floating point model subset of IEEE 754
  - no IEEE extended formats (80 bit FPU stack)
- many compiler optimizations (e.g. associativity) disallowed
- cannot use hardware acceleration
  - e.g no mapping sin(x) to x86 opcodes
  - no fused multiply-adds

Optimizing Java linear algebra

- Use native Java arrays: A[][]
- algorithms in 100% Pure Java
- exploit
  - multi-level blocking (40x40 cache, 8x8 on-chip)
  - loop unrolling
  - indexing optimizations
  - maximize on-chip / in-cache operations
- code in pure Java etc.

Matmult optimization strategies



*8x8_11: 8x8 blocks, dot-product version, B column scalarized, rows of A aliased
*8x8_12: 8x8 blocks, dot-product version, A & B scalarized

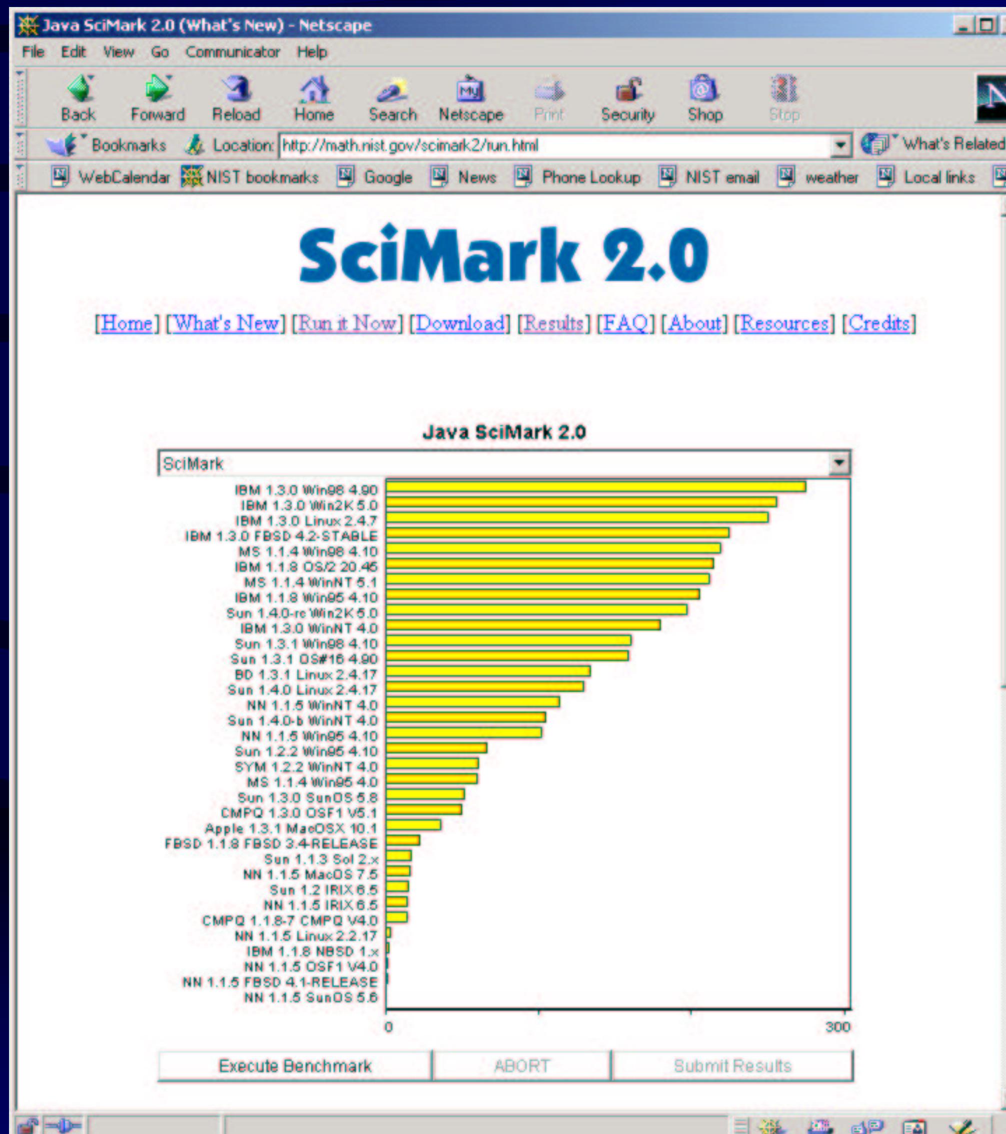
*2.8 GHz P4, IBM JVM (classic) 1.3.1, Windows XP

Java Benchmarking Efforts

- Caffine Mark
- SPECjvm98
- Java Linpack
- Java Grande Forum Benchmarks
- SciMark
- Image/J benchmark
- BenchBeans
- VolanoMark
- Plasma benchmark
- RMI benchmark
- JMark
- JavaWorld benchmark
- ...

SciMark Benchmark

- Numerical benchmark for Java, C/C++
- composite results for five kernels:
  - FFT (complex, 1D)
  - Successive Over-relaxation
  - Monte Carlo integration
  - Sparse matrix multiply
  - dense LU factorization
- results in Mflops
- two sizes: small, large

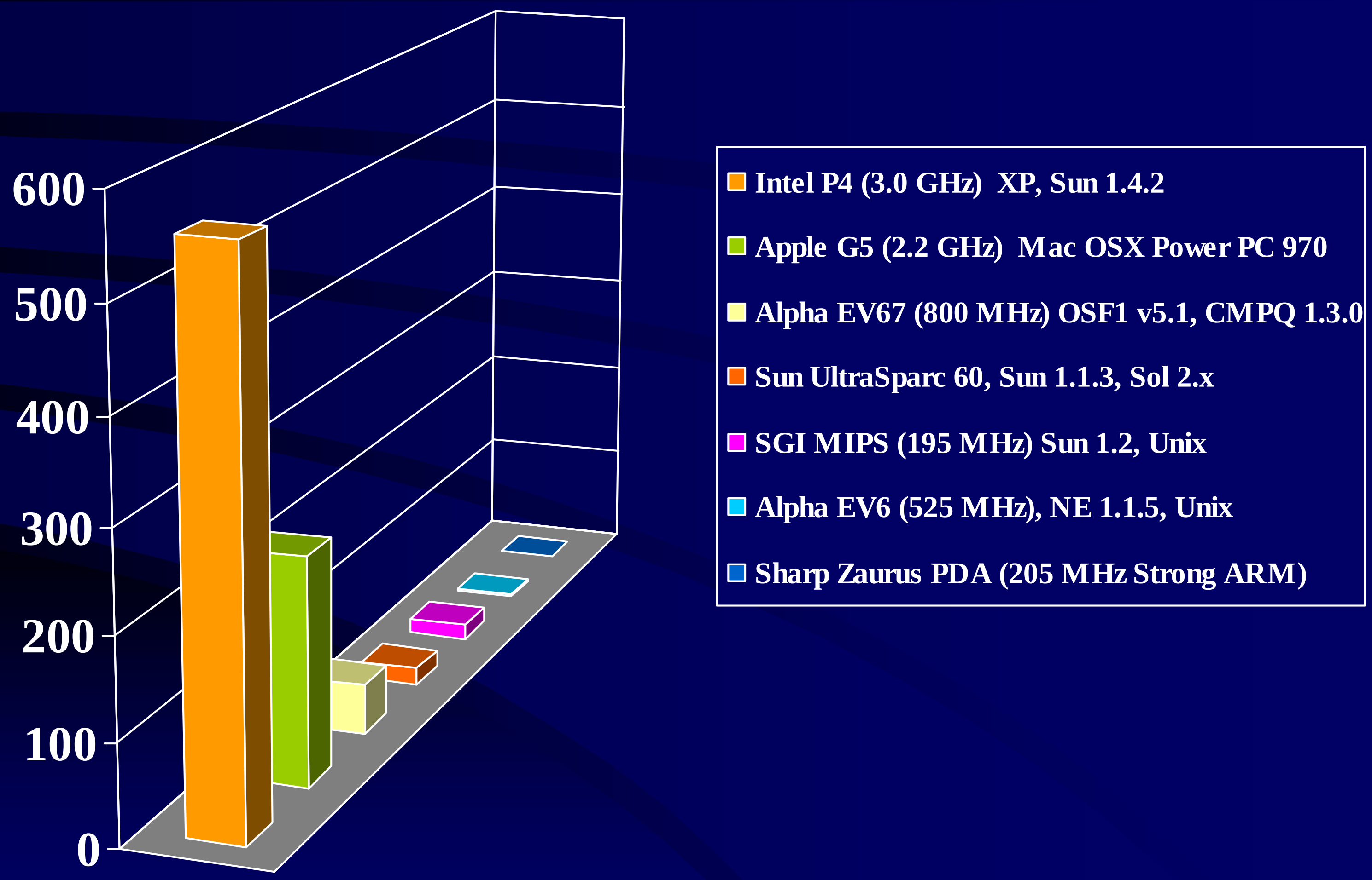


Java SciMark 2.0 (What's New) - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://math.nist.gov/scimark2/run.html
What's Related
WebCalendar NIST bookmarks Google News Phone Lookup NIST email weather Local links
SciMark 2.0
[Home] [What's New] [Run it Now] [Download] [Results] [FAQ] [About] [Resources] [Credits]
Java SciMark 2.0
SciMark
IBM 1.3.0 Win98 4.90
IBM 1.3.0 Win2K 5.0
IBM 1.3.0 Linux 2.4.7
IBM 1.3.0 FBSD 4.2-STABLE
MS 1.1.4 Win98 4.10
IBM 1.1.8 OS/2 20.45
MS 1.1.4 WinNT 5.1
IBM 1.1.8 Win95 4.10
Sun 1.4.0-rc Win2K 5.0
IBM 1.3.0 WinNT 4.0
Sun 1.3.1 Win98 4.10
Sun 1.3.1 OS#16 4.90
BD 1.3.1 Linux 2.4.17
Sun 1.4.0 Linux 2.4.17
NN 1.1.5 WinNT 4.0
Sun 1.4.0-b WinNT 4.0
NN 1.1.5 Win95 4.10
Sun 1.2.2 Win95 4.10
SYM 1.2.2 WinNT 4.0
MS 1.1.4 Win95 4.0
Sun 1.3.0 SunOS 5.8
CMPQ 1.3.0 OSF1 V5.1
Apple 1.3.1 MacOSX 10.1
FBSD 1.1.8 FBSD 3.4-RELEASE
Sun 1.1.3 Sol 2.x
NN 1.1.5 MacOS 7.5
Sun 1.2 IRIX 6.5
NN 1.1.5 IRIX 6.5
CMPQ 1.1.8-7 CMPQ V4.0
NN 1.1.5 Linux 2.2.17
IBM 1.1.8 NBSD 1.x
NN 1.1.5 OSF1 V4.0
NN 1.1.5 FBSD 4.1-RELEASE
NN 1.1.5 SunOS 5.6
0
300
Execute Benchmark
ABORT
Submit Results

SciMark database of results

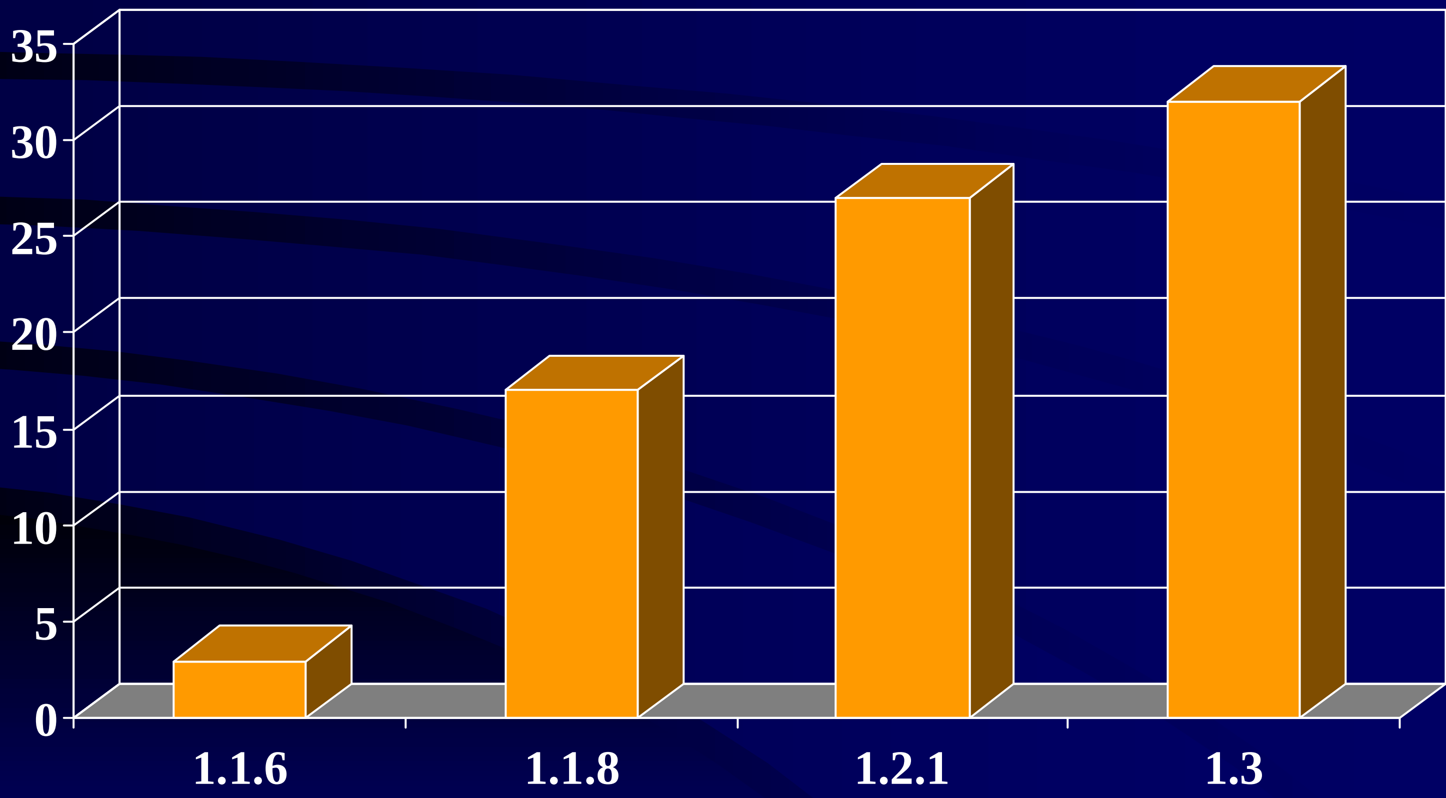
- Over 2,200 separate results for
- Hardware platforms
  - laptops, desktops, and PDAs: Intel, Apple, Sun, IBM, Apple, Sharp, AMD
- Java Virtual Machines (JVM)
  - Sun, IBM, Apple, Blackdown, Microsoft, Compaq, HP, Symantec, Netscape, BEA, NSIcom, Golden Code, FreeBSD
- Operating Systems
  - Windows 95/98/2000/ME/NT/XP, OS/2, AIX, NetBSD, Linux, IRIX, FreeBSD, Ultrix, Solaris, SunOS, Apple OS X, Compaq, NETWARE, OSF1

Some SciMark 2.0 results



JVMs have improved over time

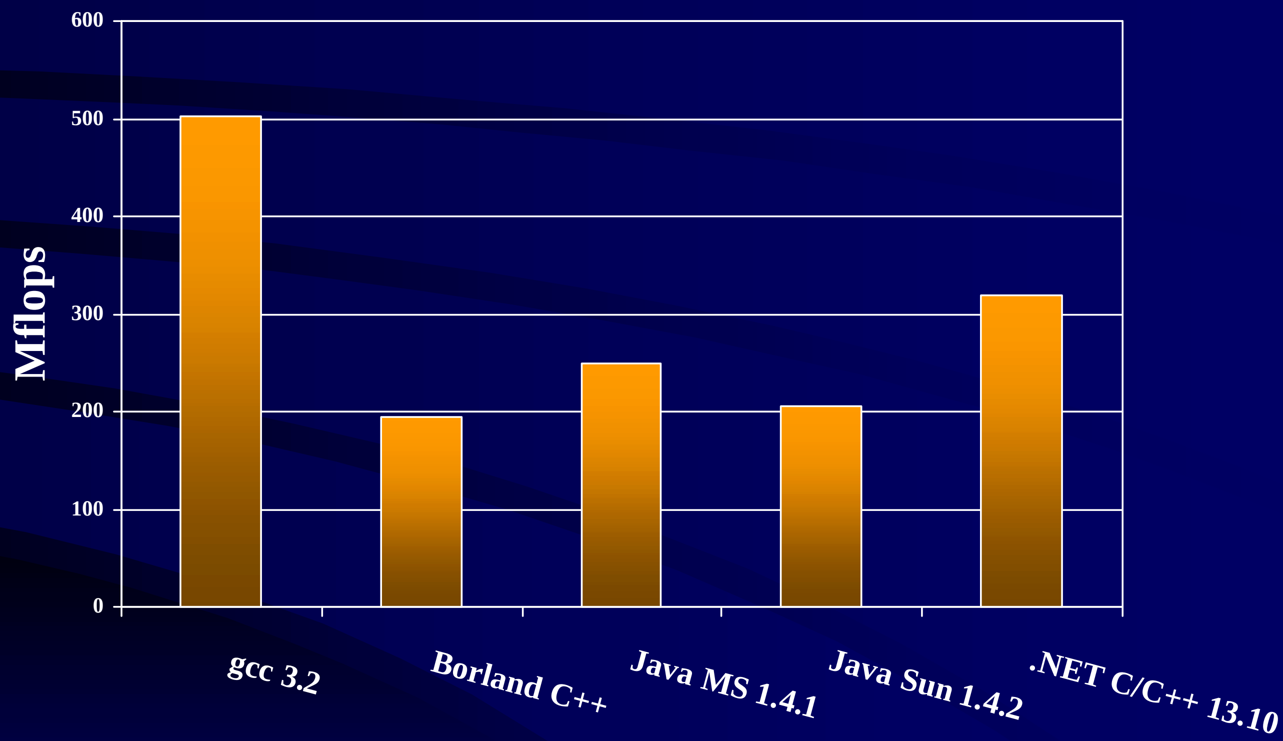
(Scimark scores)



SciMark : 333 MHz Sun Ultra 10

SciMark 2 results

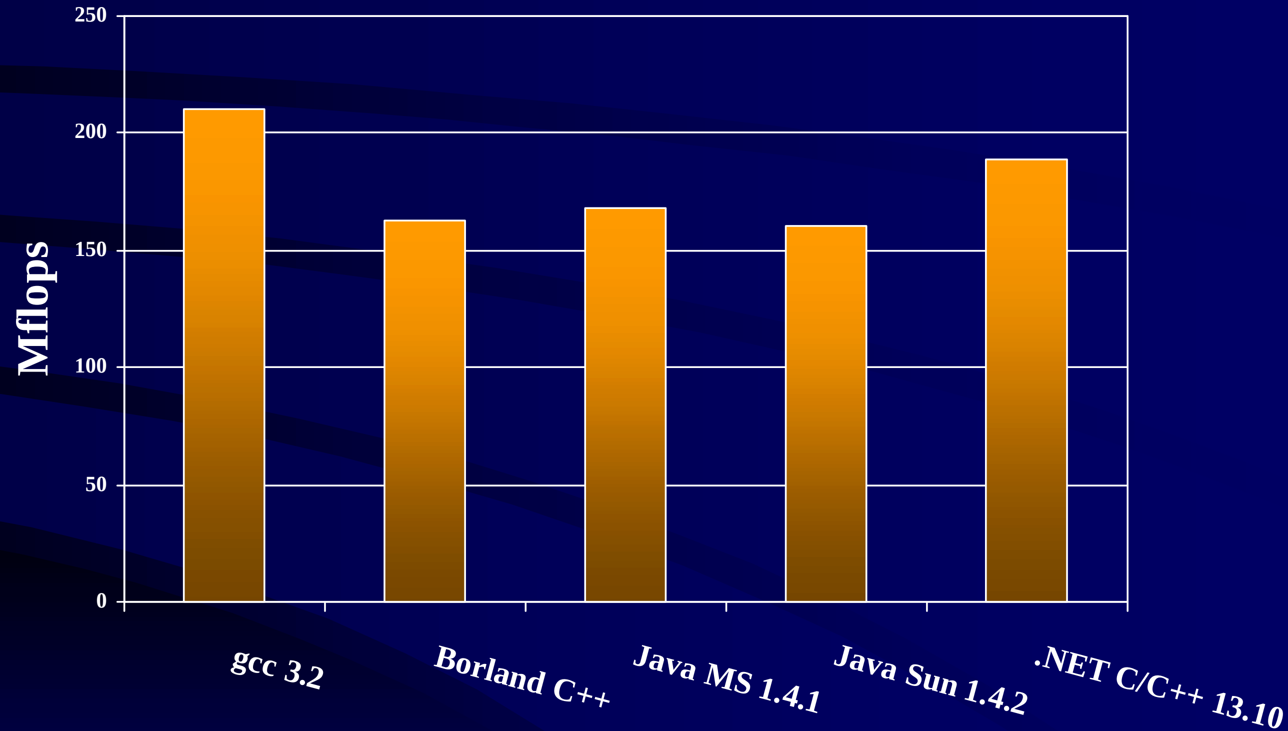
Small in-cache problem sizes



*2.5 GHz P4, gcc –O6 –funroll-loops, bcc32 –O, .NET C/C++ -G7 –Ox-a

SciMark 2 results

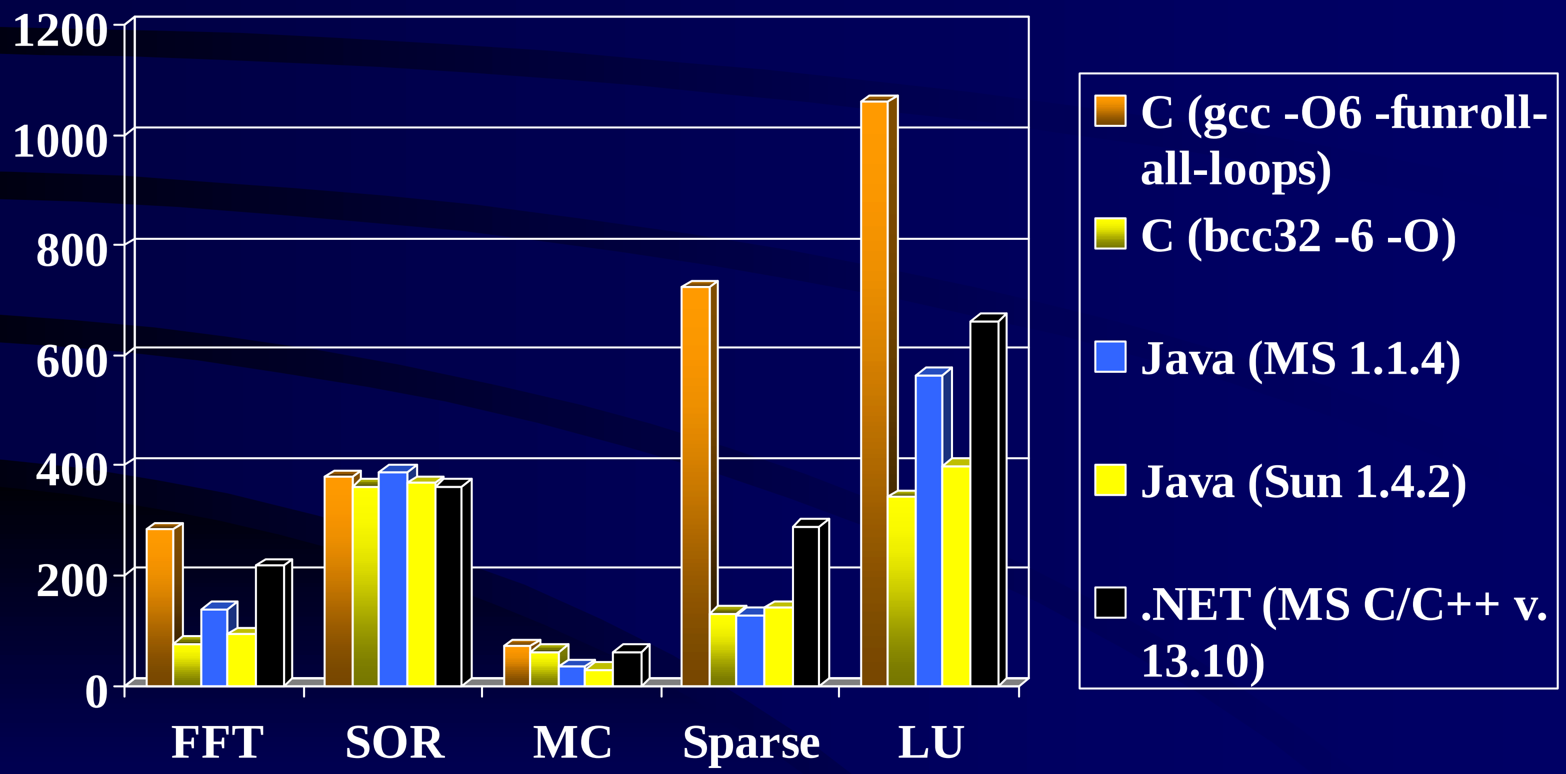
Large out-of-cache problem sizes



*2.5 GHz P4, gcc –O6 –funroll-all-loops, bcc32 –O, .NET C/C++ -G7 –Ox-a

SciMark: Java vs. C

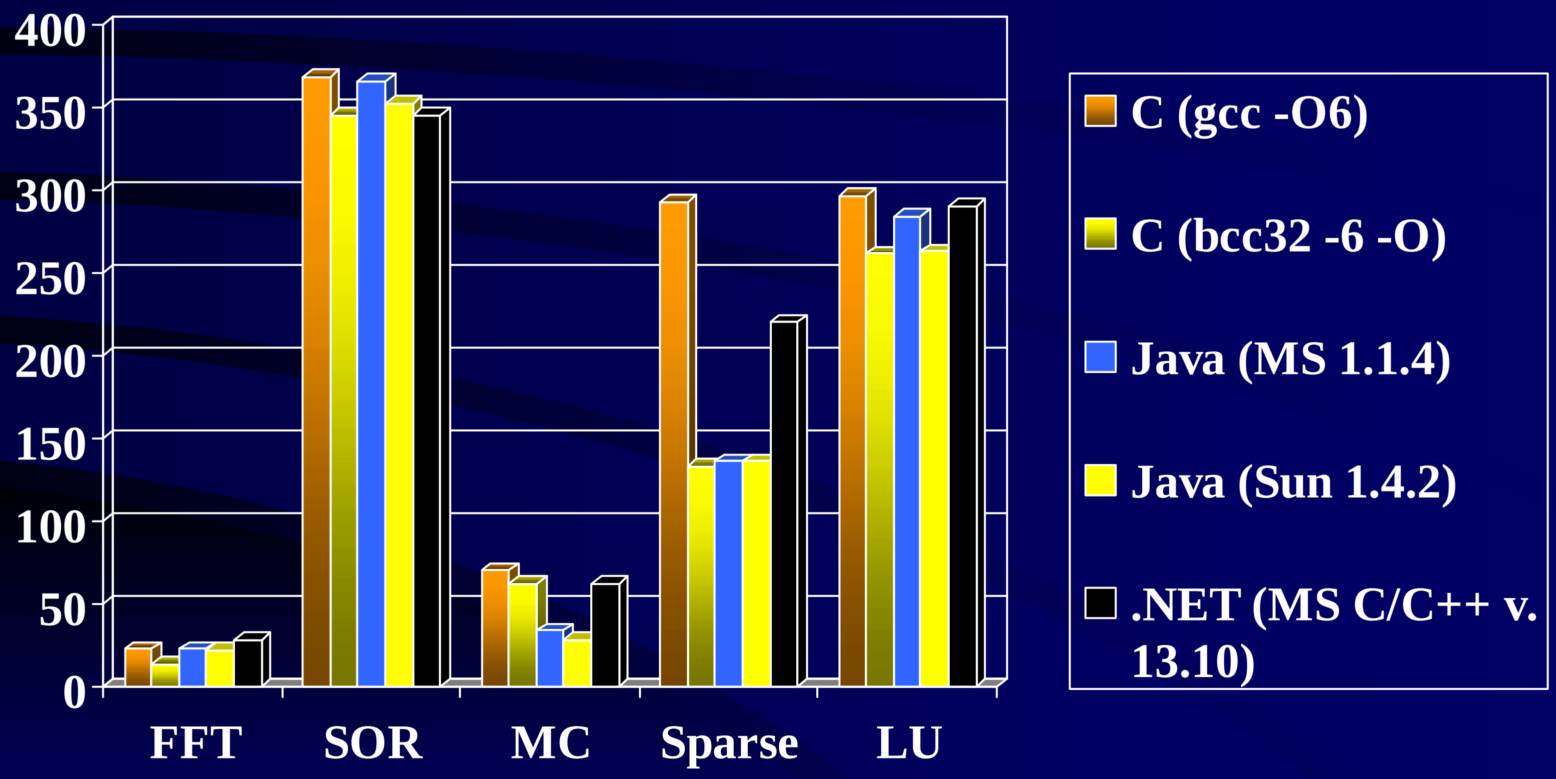
Small in-cache problem sizes



(Intel P4 2.5GHz, Windows XP)

SciMark: Java vs. C

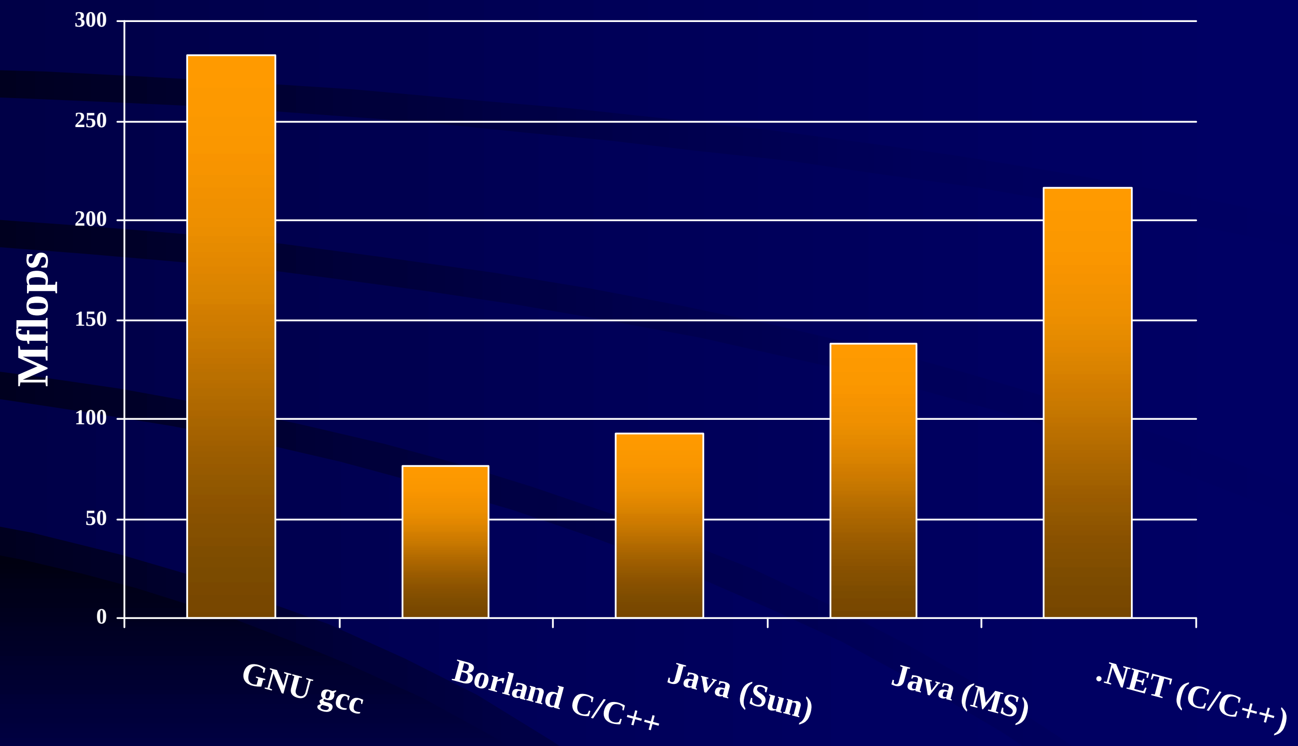
Large out-of-cache problem sizes



(Intel P4 2.5GHz, Windows XP)

SciMark FFT results

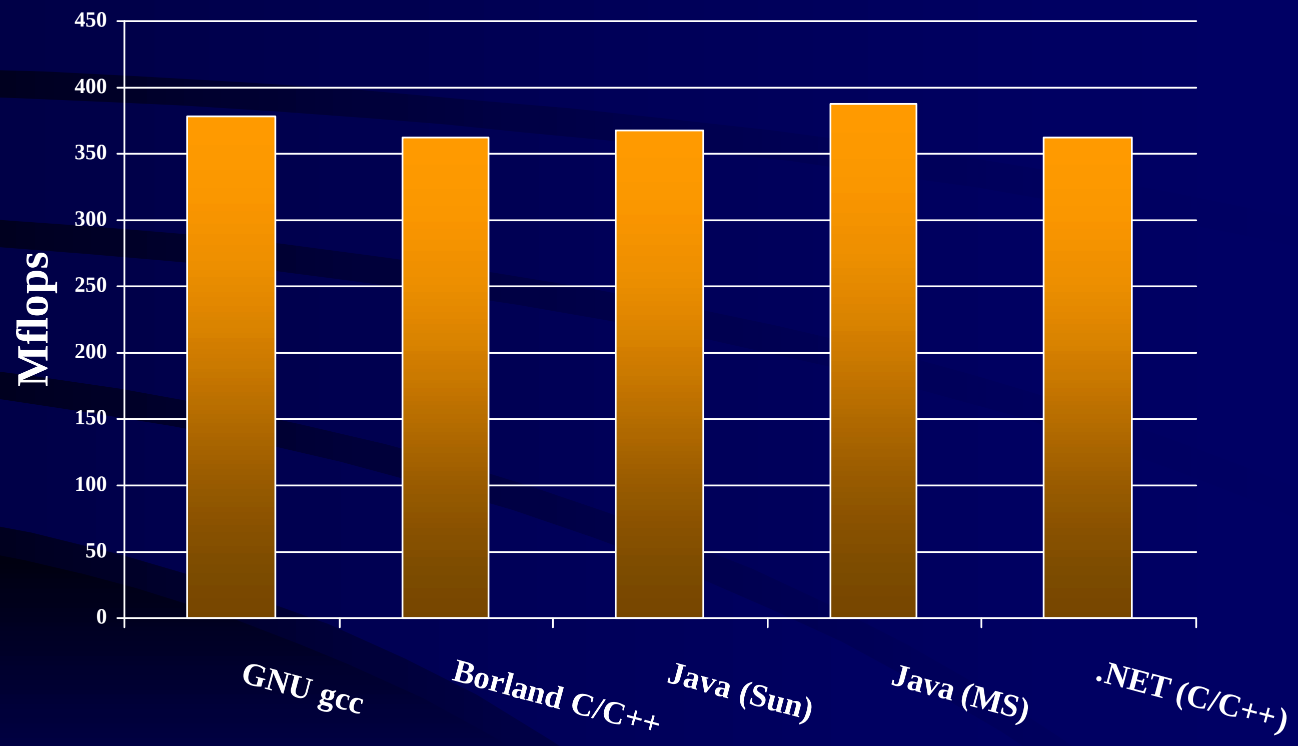
Intel 2.5 GHz Pentium 4 (Mflops)



*gcc –O6 –funroll-all-loops, bcc32 –O, MS JVM 1.1.4, Sun JVM 1.4.2, .NET C/C++ -G7 –Ox-a

SciMark SOR results

Intel 2.5 GHz Pentium 4 (Mflops)



*gcc –O6 –funroll-all-loops, bcc32 –O, MS JVM 1.1.4, Sun JVM 1.4.2, .NET C/C++ -G7 –Ox-a

Current SciMark high scores
(May 28, 2004)

- Scimark high score: 555Mflops*
  - FFT: 338 Mflops
  - Jacobi: 761 Mflops
  - Monte Carlo: 20 Mflops
  - Sparse matmult: 527 Mflops
  - LU factorization: 1127 Mflops

* Intel 3 GHz Pentium 4, Sun JVM 1.3.2, Windows XP

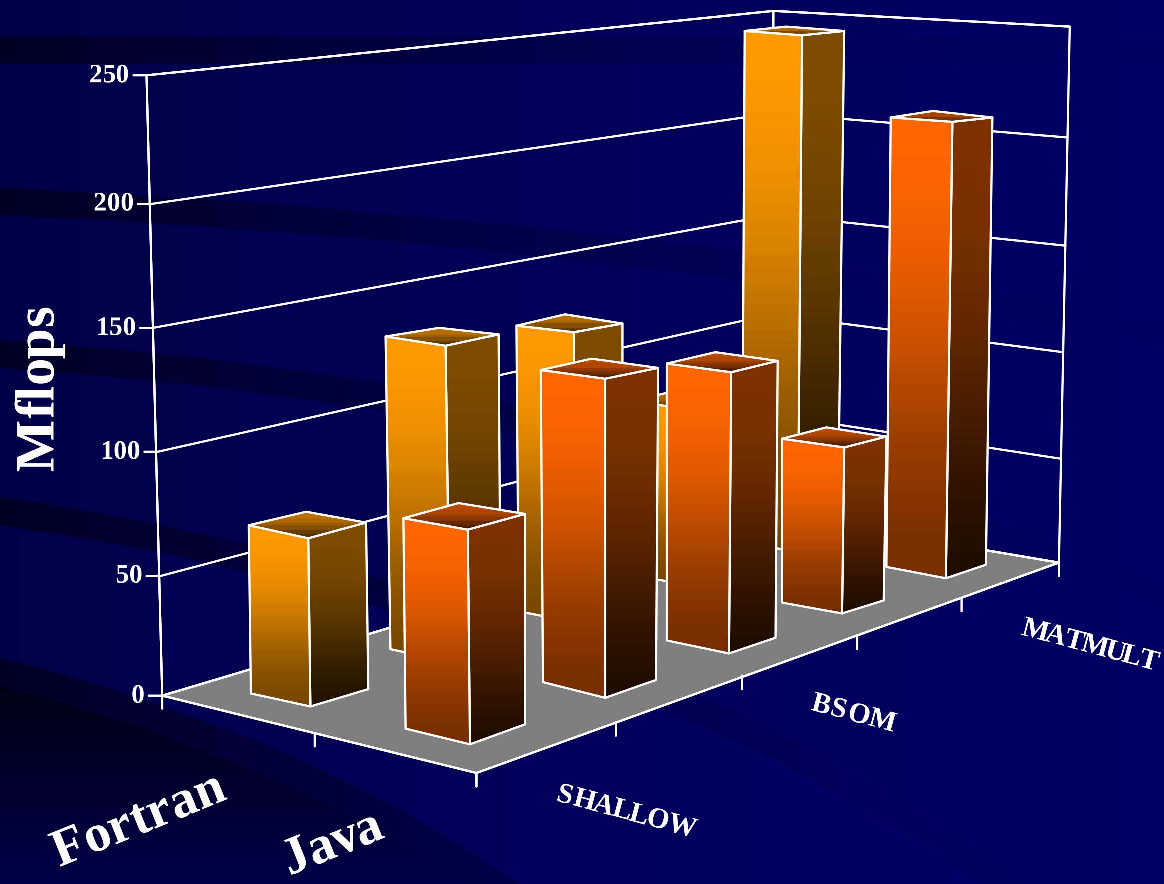
Other optimization approaches...

- Use an aggressive optimizing compiler
- code using Array classes which mimic Fortran storage
  - e.g. A[i][j] becomes A.get(i,j)
  - ugly, but can be fixed with operator overloading extensions
- exploit hardware (FMAs)
- result: 85+% of Fortran on RS/6000

IBM High Performance Compiler

- Moreria, et. al
- native compiler (.java -> .exe)
- requires source code
- can't embed in browser, but…
- produces very fast codes

Java vs. Fortran Performance



*IBM RS/6000 67MHz POWER2 (266 Mflops peak) AIX Fortran, HPJC

Yet another approach...

- HotSpot
  - Sun Microsystems (now default on JVMs)
- Progressive profiler/compiler
- trades off aggressive compilation/optimization at code bottlenecks
- quicker start-up time than JITs
- tailors optimization to application

Concurrency

- Java threads
  - runs on multiprocessors in NT, Solaris, AIX
  - provides mechanisms for locks, synchornization
  - can be implemented in native threads for performance
  - no native support for parallel loops, etc.

Concurrency

- Remote Method Invocation (RMI)
  - extension of RPC
  - high-level than sockets/network programming
  - works well for functional parallelism
  - works poorly for data parallelism
  - serialization is expensive
  - no parallel/distribution tools

Java numerical software

(libraries & tools)

Java Numerics Working Group

- industry-wide consortium to establish tools, APIs, and libraries
  - IBM, Intel, Compaq/Digital, Sun, MathWorks, VNI, NAG
- component of Java Grande Forum
- updated Java's floating-point model
- served as focal point for numeric activities
  - proposals and implementations for
    - complex, arrays, mult-adds, fdlibm
  - libraries, compilers, language extensions

Solutions for

- floating point model
- true multidimensional arrays
- complex data types
- lightweight objects
- operator overloading
- generic typing (templates)

Java Numerical Software

- General
  - Apflot, Colt, JADE, Java3D (matrix), ArciMath BigDecimal, IBM Alphaworks, Jsci, Spline++, JMSL (Visual Numerics), jCrunch, mpjava, RngPack, OpsResearch,
- Linear Algebra
  - JAMA, Jampack, Java LAPACK, Matrix Toolkit, Owlpack
- Language extensions (compilers)
  - multi-dimensional arrays (IBM), complex numbers (zeta) , HPJava, Cj, Titanium

Parallel Java projects

- Java-MPI
- JavaPVM
- Titanium (UC Berkeley)
- HPJava
- DOGMA
- JTED
- Jwarp
- DARP
- Tango
- DO!
- Jmpi
- MpiJava
- JET Parallel JVM

Current developments

- Java v. 1.5
  - generics, for-each loops, auto-boxing, enums, varargs, static import, metadata
- Microsoft C# (.NET)
  - Built around Microsoft Windows (Win32 API)
  - Common language runtime (CLR)
  - Native language: C#
  - supports subsets of C, C++, and Fortran
  - Some support outside MS
    - dotGNU,
    - mono (Novell)

Conclusions

- Java's strength:
  - binary portability; single software distribution
  - competitive performance (0.5x rule-of-thumb)
- Java's obstacles:
  - no standard support for multidimensional arrays, complex numbers, and operator overloading
  - limited numeric software and library support
  - no blind conversion of C/Fortran codes
- Can be solved (technologically)
  - but need standards and support

Scientific Java Resources

- Java Numerics Group
  - http://math.nist.gov/javanumerics
- Java Grande Forum
  - http://www.javagrade.org
- SciMark Benchmark
  - http://math.nist.gov/scimark