

NPB-MPJ: NAS Parallel Benchmarks Implementation for Message-Passing in Java

Damián A. Mallón, Guillermo L. Taboada*, Juan Touriño,
Ramón Doallo

Computer Architecture Group
Dept. of Electronics and Systems
University of A Coruña, Spain
Email: {dalvarezm,taboada,juan,doallo}@udc.es

Euromicro PDP, 2009

- 1 Motivation
 - Java for High-Performance Computing
 - Message-Passing in Java
 - Previous Works
- 2 Design, Implementation and Optimization of NPB-MPJ
 - NPB-MPJ Objectives
 - NPB-MPJ Design
 - NPB-MPJ Implementation
 - NPB-MPJ Optimization
- 3 Performance Evaluation
 - Experimental Configuration
 - NPB Evaluation on a Gigabit Ethernet Cluster
 - NPB Evaluation on an InfiniBand Cluster
 - Performance of SP Pseudo-application

Java is a Competitive Alternative for HPC in the Multi-core Era

- Java is a (the most) **popular** language.
- Interesting features:
 - Built-in networking
 - Built-in multi-threading
 - Portable, platform independent
 - Object Oriented
 - Main training language
- Many productive parallel/distributed programming libs:
 - Java RMI
 - Java Threads (high level facilities: Futures, ThreadPools. . .)
 - ProActive
 - Message-Passing in Java (MPJ) libraries
 - Java OpenMP implementations
 - . . .

Java Adoption in HPC:

- HPC developers and users **want** to use Java in their projects, but they are **highly** concerned about performance.
- Java is no longer slow (Just-In-Time compilation)!
- But there is no suitable benchmark to assess this in HPC!

Message-Passing in Java

- Message-passing is the main HPC programming model
- MPJ (MPI-like bindings for Java) Implementations
 - mpiJava. MPJ Java wrapper library over native MPI implementations (e.g. OpenMPI, MPICH).
 - MPJ Express. MPJ pure (100%) Java library.
 - MPJ/Ibis. MPJ pure Java library.
 - Parallel Java, Jcluster, P2P-MPI... (15 more –older–)
- Our goal: Implement a suitable benchmark to evaluate Java performance for HPC, particularly for MPJ

Java Benchmarks in HPC. The Java Grande Forum Benchmark Suite gathers most of the benchmarks in Java for HPC

- Four groups:
 - Sequential Benchmarks
 - Language comparison benchmarks (sequential C vs Java)
 - Multi-threaded codes
 - MPJ benchmarks
- Three levels:
 - Micro-benchmarks (e.g. ping-pong, bcast, threads sync.)
 - Kernel benchmarks (e.g. LU fact., Sparse matrix mult.)
 - Applications (e.g. N-body simul., Montecarlo, Ray tracer)

Java NPB Implementations.

- NPB-JAV: Java Multi-threaded (CG, FT, IS, MG, SP)
- NPB-PA: Java ProActive (CG, EP, FT, IS, MG)
- MPJ NPB
 - JavaMPI (2001): EP and IS (1050 SLOC)
 - MPJava (2003): CG (1000 SLOC)
 - P2P-MPI (2006): EP and IS (1050 SLOC)

NPB-MPJ allows:

- the comparison among MPJ implementations (>15)
- the evaluation of MPJ against other Java parallel libraries (ProActive)
- the assessment of MPJ versus MPI
- example of best Java programming practices for performance in HPC

NPB-MPJ Design

- NPB-MPI based (Fortran except IS and DT –C–)
- SPMD
- “Plain Objects” Design
 - Instead of complex number Object, 2 double array
 - One class per benchmark
 - C-like implementation
 - Less “object orientation” in heavy loaded methods

NPB-MPJ Implementation: 11.000 SLOC (Source LOC)

- CG (Conjugate Gradient) 1000 SLOC
- EP (Embarrassingly Parallel) 350 SLOC
- FT (FFT) 1700 SLOC
- IS (Integer Sort) 700 SLOC
- MG (MultiGrid kernel) 2000 SLOC
- DT (Data Traffic) 1000 SLOC
- SP (Scalar Pentadiagonal) 4300 SLOC

NPB-MPJ Implementation Issues:

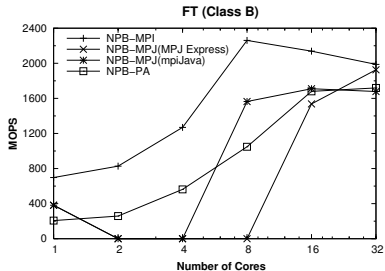
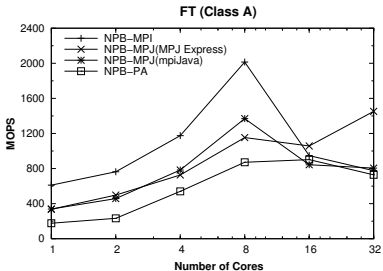
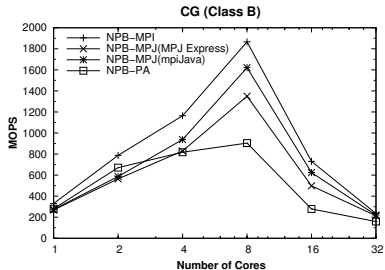
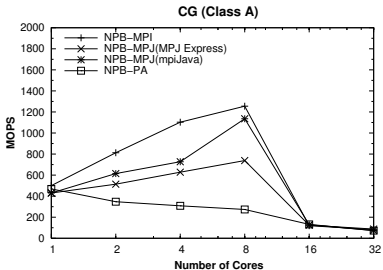
- No multi-dimensional arrays support in Java.
 - NPB uses 5 dims arrays \rightarrow Java defined `arr[v][w][x][y][z]`
 - MPI C can send contiguous elements (2 rows from a 2 dimensional array)
- “Array flattening” Technique. Mapping an n-dimensional array in a 1-dimensional array.
 - Positioning method (`position3D(x,y,z,d1,d2,d3)`)
 - Example: complex number array `compArr[2][N]`
 - Defined `compArr[2*N]`
 - Access `compArr[REAL][x] \rightarrow compArr[position2D(REAL,x,2,N)]`
 - Hides array layout

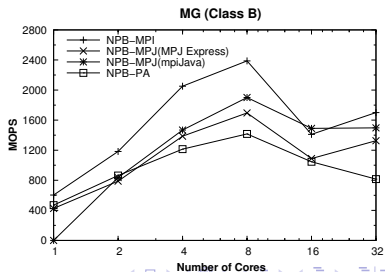
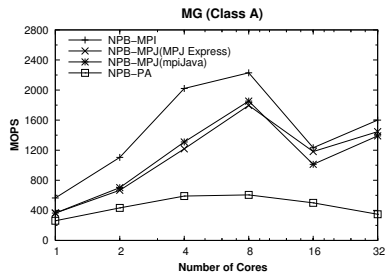
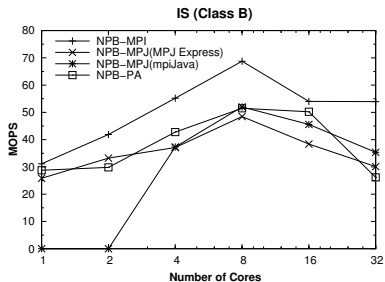
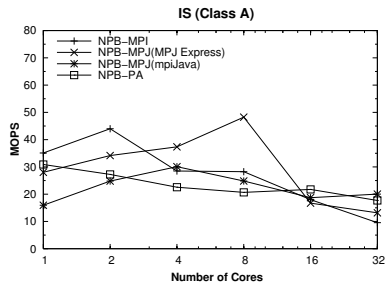
NPB-MPJ Optimization

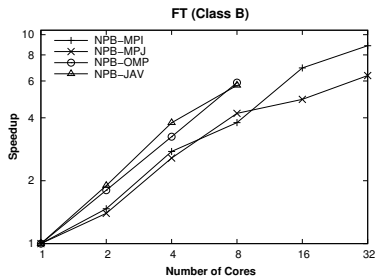
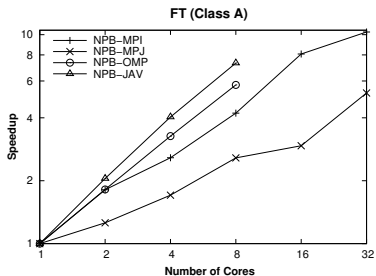
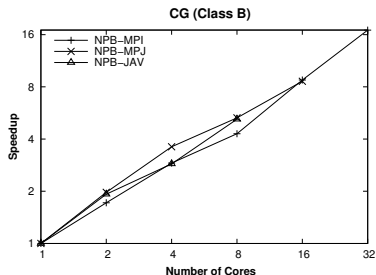
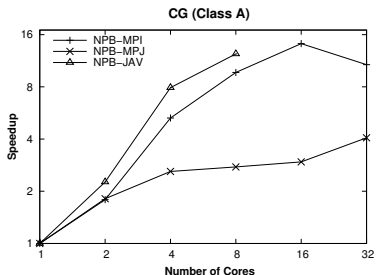
- JVM JIT compilation of heavy methods with runtime information
- JVM JIT paradoxes:
 - An “optimized” code is slower than an “unoptimized” code
 - Method inlining reduces performance
- Structured programming is the best option
 - Small frequent methods are better. E.g. `compArr[position2D(REAL,x,2,N)]` is better than `compArr[REAL+2*x]`
 - NPB-MPJ code refactored, obtaining significant improvements (up to 2800% for SP)

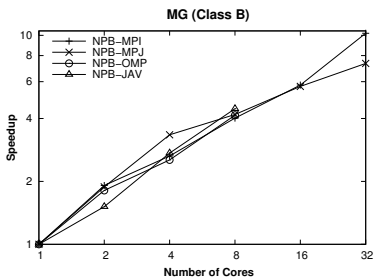
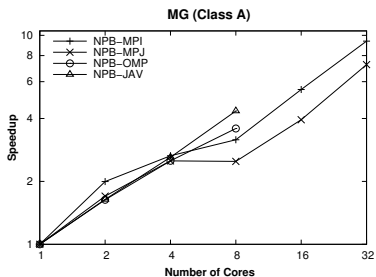
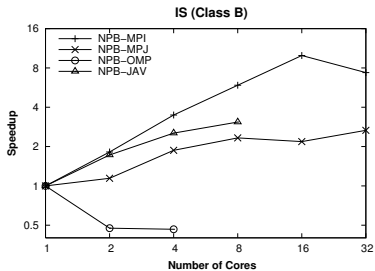
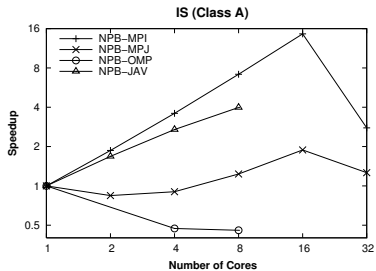
Testbeds:

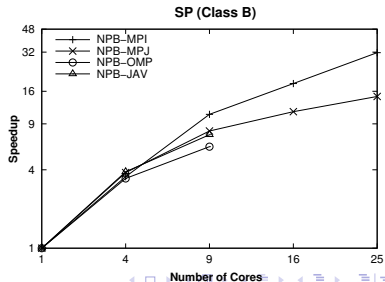
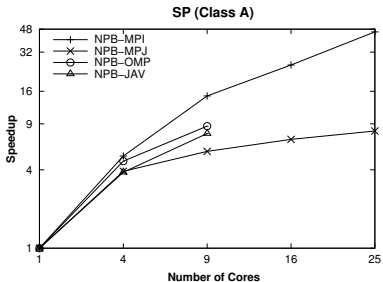
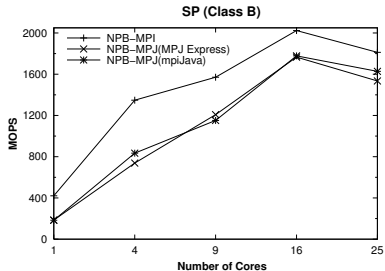
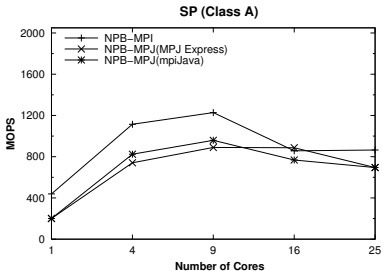
- Gigabit Ethernet cluster (8 dual dual-core nodes)
 - 2 x Intel Xeon 5060 dual-core at 3.2 GHz
 - 4 GB RAM
 - Linux, gcc 4.1.2
 - Sun JDK 1.6.0_02, MPICH2 1.0.7 (ssm)
- InfiniBand cluster (4 HP Integrity rx7640 nodes)
 - 8 x Montvale Itanium2 dual-core (IA64) at 1.6 GHz
 - 128 GB RAM
 - Linux, C compiler Intel icc 9
 - BEA JRockit 5.0 (R27.6), HP MPI 2.2.5.1
- MPJ Express 0.27, mpiJava 1.2.5x, ProActive 3.2











Summary

- NPB-MPJ is the **complete** NPB MPJ implementation
- NPB-MPJ allows the **performance comparison** of MPJ against MPI, OpenMP and other Java solutions for HPC (Threads, ProActive).
- NPB-MPJ serves to **evaluate** MPJ implementations (>10).

- Analysis of the Results
 - MPJ can **compete in performance** with MPI.
 - It is **feasible to increase** Java performance.

Questions?

Contact: Guillermo L. Taboada *taboada@udc.es*
<http://www.des.udc.es/~gltaboada/>

Computer Architecture Group, Dept. of Electronics and
Systems

University of A Coruña, Spain

For Further Reading I



B. Amedro, V. Bodnartchouk, D. Caromel, C. Delbé, F. Huet, and G. L. Taboada.

Current State of Java for HPC.

In *INRIA Technical Report RT-0353*, pages 1–24, INRIA Sophia Antipolis, Nice, France, 2008,
<http://hal.inria.fr/inria-00312039/en/>.



B. Amedro, D. Caromel, F. Huet, and V. Bodnartchouk.

Java ProActive vs. Fortran MPI: Looking at the Future of Parallel Java.

In *Proc. 10th Intl. Workshop on Java and Components for Parallelism, Distribution and Concurrency (IWJacPDC'08)*, Miami, FL, USA, page 134b (8 pages), 2008.