

Java for High Performance Computing: Myth or Reality?

Guillermo López Taboada

Grupo de Arquitectura de Computadores
Universidade da Coruña
<http://gac.udc.es/~gltaboada>
taboada@udc.es

CIEMAT 4/11/2011

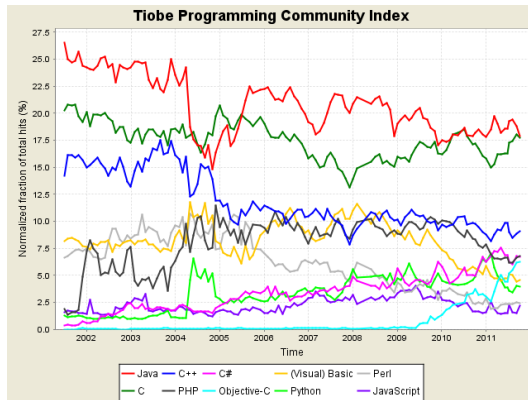
Outline

- 1 Motivation
- 2 Java for High Performance Computing
- 3 Java HPC Codes
- 4 Performance Evaluation
- 5 Conclusions

Java is an Alternative for HPC in the Multi-core Era

Language popularity:
(% skilled developers)

- #1 **Java (17.9%)**
- #2 C (17.7%)
- #3 C++ (9.1%)
- #20 Matlab (0.6%)
- #29 R (0.4%)
- #31 Fortran (0.4%)



Java is an Alternative for HPC in the Multi-core Era

Interesting features:

- Built-in networking
- Built-in multi-threading
- Portable, platform independent
- Object Oriented
- Main training language

Many productive parallel/distributed programming libs:

- Java shared memory programming (high level facilities: Concurrency framework)
- Java Sockets
- Java RMI
- Message-Passing in Java (MPJ) libraries

Java Adoption in HPC

- HPC developers and users usually **want** to use Java in their projects.
- Java code **is no longer slow** (Just-In-Time compilation)!
- But still performance penalties in Java communications:

Pros and Cons:

- high programming productivity.
- but they are **highly** concerned about performance.

Java Adoption in HPC

- HPC developers and users usually **want** to use Java in their projects.
- Java code **is no longer slow** (Just-In-Time compilation)!
- But still performance penalties in Java communications:

JIT Performance:

- Like native performance.
- Java can even outperform native languages thanks to the dynamic compilation.

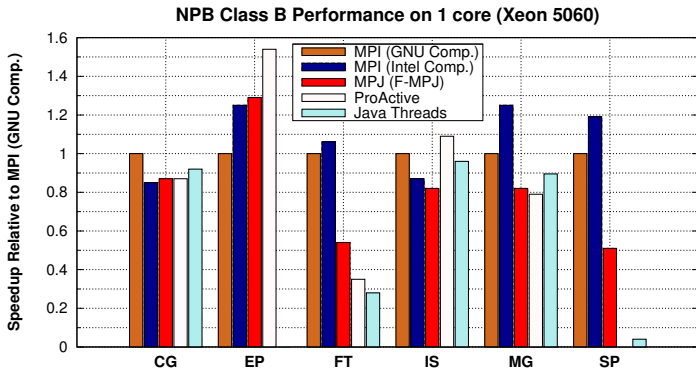
Java Adoption in HPC

- HPC developers and users usually **want** to use Java in their projects.
- Java code **is no longer slow** (Just-In-Time compilation)!
- But still performance penalties in Java communications:

High Java Communications Overhead:

- Poor high-speed networks support.
- The data copies between the Java heap and native code through JNI.
- Costly data serialization.
- The use of communication protocols unsuitable for HPC.

Experimental Results on One Core (relative perf.)



Emerging Interest in Java for HPC



The screenshot shows a Mozilla Firefox browser window with the address bar containing the URL `http://blogs.sun.com/jag/entry/current_state_of_java_for`. The page title is "James Gosling: on the Java Road". The browser's address bar shows "Google" as the search engine. The page content includes a navigation bar with links like "Fun at SIGGRAPH | Main | In Germany this week", a date stamp "TUESDAY SEPTEMBER 02, 2008", and a section titled "Current State of Java for HPC". The text in this section discusses a talk at the AMD keynote, mentioning Denis Caromel of INRIA and a Tech Report on HPC micro benchmarks. A cartoon character of James Gosling is visible on the right side of the page, along with a "BOOKS WORTH READING" section.

James Gosling: on the Java Road

« Fun at SIGGRAPH | Main | In Germany this week »

TUESDAY SEPTEMBER 02, 2008

Current State of Java for HPC

At the last JavaOne I did a walk-on talk during the AMD keynote where I talked about how incredible HotSpot's performance had become - beating the best C compilers. I ended my talk with a joking comment that "the next target is Fortran". Afterwards, Denis Caromel of inria came up to me and said "you're already there". He and some colleagues had been working on some comparisons between java and Fortran for HPC. Their final report *Current State of Java for HPC* has been made available as a Tech Report and makes pretty interesting reading. There are a lot of HPC micro benchmarks in it which look great. Thanks! [Permalink](#) [Comments](#) [3]

INRIA

8 Fallacies bio Standards

BOOKS WORTH READING

Current State of Java for HPC

HAL - INRIA :: [inria-00312039, version 1] Current State of Java for HPC - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://hal.inria.fr/inria-00312039/en

Getting Started Latest BBC Headlines Grupo de Arquitectu... Actualidade UDC La Voz de Galicia ... PANTHER - News

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

Home Submit Browse Search Help Services

english version

inria-00312039, version 1

[See detailed view](#)

Current State of Java for HPC

Brian Amedro ¹, Vladimir Bodnartchouk ², Denis Caromel ^{1,3}, Christian Delbe ¹, Fabrice Huet ², Guillermo L. Taboada ^{3,4} (2008)

[BibTeX EndNote...](#)

[Access to full text](#)
PDF

Abstract: About ten years after the Java Grande effort, this paper aims at providing a snapshot of the current status of Java for High Performance Computing. Multi-core chips are becoming mainstream, offering many ways for a Java Virtual Machine (JVM) to take advantage of such systems for critical tasks such as Just-In-Time compilation or Garbage Collection. We first perform some micro benchmarks for various JVMs, showing the overall good performance for basic arithmetic operations. Then we study a Java implementation of the Nas Parallel Benchmarks, using the ProActive middleware for distribution. Comparing this implementation with a Fortran/MP1 one, we show that they have similar performance on computation intensive benchmarks, but still have scalability issues when performing intensive communications. Using experiments on clusters and multi-core machines, we show that the performance varies greatly, depending on the Java Virtual Machine used (version and vendor) and the kind of computation performed.

^a - Faculty of Informatics, University of A Coruna, Spain

¹: **OASIS** (INRIA Sophia Antipolis / Laboratoire ISS)
INRIA - Université de Nice Sophia-Antipolis - CNRS UMR6070

²: **ActiveEon**
ActiveEon

³: **SLOOP** (INRIA Sophia Antipolis)
INRIA

⁴: **University of A Coruna - Computer Architecture Group**
University of A Coruna

Done

Java for High Performance Computing

Current options in Java for High Performance Computing:

- Java Shared Memory Programming
- Java Sockets
- Java RMI
- Message-Passing in Java (MPJ)

Java for HPC

Java Shared Memory Programming:

- Java Threads
- Concurrency Framework (ThreadPools, Tasks ...)
- Parallel Java (PJ)
- Java OpenMP (JOMP and JaMP)

JOMP

Listing 1: JOMP example

```
public static void main (String argv[]) {  
    int myid;  
    //omp parallel private(myid)  
    {  
        myid = OMP.getThreadNum();  
        System.out.println ("Hello from " + myid);  
    }  
  
    //omp parallel for  
    for (i=1;i<n;i++) {  
        b[i] = (a[i] + a[i-1]) * 0.5;  
    }  
}
```

Java Communication Libraries Overview

Java HPC Applications

Java Message-passing libraries

Java RMI / Low-level messaging libraries

Java Sockets libraries

HPC Communications Hardware

Java Sockets

Standard and widely extended low-level programming interface for networked communications.

Current implementations:

- **IO sockets**
- NIO sockets
- Ibis sockets
- Java Fast Sockets

Pros and Cons:

- easy to use.
- but only TCP/IP support.
- lack non-blocking communication.
- lack HPC tailoring.

Java Sockets

Standard and widely extended low-level programming interface for networked communications.

Current implementations:

- IO sockets
- **NIO sockets**
- Ibis sockets
- Java Fast Sockets

Pros and Cons:

- provides non-blocking communication.
- but only TCP/IP support.
- lack HPC tailoring.
- difficult use.

Java Sockets

Standard and widely extended low-level programming interface for networked communications.

Current implementations:

- IO sockets
- NIO sockets
- **Ibis sockets**
- Java Fast Sockets

Pros and Cons:

- easy to use.
- with Myrinet support.
- but lack non-blocking communication.
- lack HPC tailoring.

Java Sockets

Standard and widely extended low-level programming interface for networked communications.

Current implementations:

- IO sockets
- NIO sockets
- Ibis sockets
- **Java Fast Sockets**

Pros and Cons:

- easy to use.
- efficient high-speed networks support.
- efficient shared memory protocol.
- with HPC tailoring.
- but lack non-blocking support.

Remote Method Invocation

RMI (Remote Method Invocation)

- Widely extended
- RMI-based middleware (e.g., ProActive)
- RMI Optimizations:
 - KaRMI
 - Manta
 - Ibis RMI
 - **Opt RMI**

Java Message-Passing Libraries

Message-passing is the main HPC programming model.

- Implementation approaches in Java message-passing libraries.

Implementation approaches

- RMI-based.
- Wrapping a native library (e.g., MPI libraries: OpenMPI, MPICH).
- Sockets-based.
- Low-level communication device.

Listing 2: MPJ example

```
import mpi.* ;

public class Hello {

    public static void main (String argv[]) {
        MPI.Init(args);
        int rank = MPI.COMM_WORLD.Rank() ;

        if (rank == 0){
            String[] msg = new String[1];
            msg[0] = new String("Hello");
            MPI.COMM_WORLD.Send(msg, 0, 1, MPI.OBJECT, 1, 13);
        } else if (rank == 1) {
            String[] message = new String[1];
            MPI.COMM_WORLD.Recv(message, 0, 1, MPI.OBJECT, 0, 13);
            System.out.println(message[0]);
        }
        MPI.Finalize() ;
    }
}
```

	Pure Java Impl.	Socket impl.		High-speed network support			API		
		Java IO	Java NIO	Myrinet	InfiniBand	SCI	mpiJava 1.2	JGF MPJ	Other APIs
MPJava	✓		✓						✓
Jcluster	✓	✓							✓
Parallel Java	✓	✓							✓
mpiJava				✓	✓	✓	✓		
P2P-MPI	✓	✓	✓				✓		
MPJ Express	✓		✓	✓			✓		
MPJ/Ibis	✓	✓		✓				✓	
JMPI	✓	✓							✓
F-MPJ	✓	✓	✓	✓	✓	✓	✓		

Java Communication Libraries Overview

Java HPC Applications (**Develop Efficient Codes**)

Java Message-passing libraries (**Scalable Algorithms**)

Low-level messaging libraries (**MPJ Devices**)

HPC Hardware

iodev: Low-level Message-Passing Library

The use of pluggable low-level communication devices is widely extended in message-passing libraries.

Message-passing Low-level Devices:

- **MPICH/MPICH2 ADI/ADI3 (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).**
- OpenMPI BTL (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- MPJ Express xdev (NIO sockets, MX for Myrinet, and shared memory).
- F-MPJ xxdev (NIO/IO sockets, MX for Myrinet, IBV for InfiniBand, and shared memory).

iodev: Low-level Message-Passing Library

The use of pluggable low-level communication devices is widely extended in message-passing libraries.

Message-passing Low-level Devices:

- MPICH/MPICH2 ADI/ADI3 (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- **OpenMPI BTL (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).**
- MPJ Express xdev (NIO sockets, MX for Myrinet, and shared memory).
- F-MPJ xxdev (NIO/IO sockets, MX for Myrinet, IBV for InfiniBand, and shared memory).

iodev: Low-level Message-Passing Library

The use of pluggable low-level communication devices is widely extended in message-passing libraries.

Message-passing Low-level Devices:

- MPICH/MPICH2 ADI/ADI3 (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- OpenMPI BTL (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- **MPJ Express xdev (NIO sockets, MX for Myrinet, and shared memory).**
- F-MPJ xxdev (NIO/IO sockets, MX for Myrinet, IBV for InfiniBand, and shared memory).

iodev: Low-level Message-Passing Library

The use of pluggable low-level communication devices is widely extended in message-passing libraries.

Message-passing Low-level Devices:

- MPICH/MPICH2 ADI/ADI3 (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- OpenMPI BTL (GM/MX for Myrinet, IBV/VAPI for InfiniBand, and shared memory).
- MPJ Express xdev (NIO sockets, MX for Myrinet, and shared memory).
- **F-MPJ xxdev (NIO/IO sockets, MX for Myrinet, IBV for InfiniBand, and shared memory).**

xxdev API. Public interface of the *xxdev.Device* class

```
public abstract class Device {  
    static public Device newInstance(String deviceImpl);  
    public int[] init(String[] args);  
    public int id();  
    public void finish();  
  
    public Request isend(Object buf, int dst, int tag);  
    public Request irecv(Object buf, int src, int tag, Status stts);  
  
    public void send(Object buf, int dst, int tag);  
    public Status recv(Object buf, int src, int tag);  
  
    public Request issend(Object buf, int dst, int tag);  
    public void ssend(Object buf, int dst, int tag);  
  
    public Status iprobe(int src, int tag, int context);  
    public Status probe(int src, int tag, int context);  
    public Request peek();  
}
```

F-MPJ Communication Devices

MPJ Applications				
F-MPJ Library				
device layer	omxdev	ibvdev	niodev/iodev	smpdev
JVM	JNI		Java Sockets	Java Threads
native comms	Open-MX	IBV	TCP/IP	
	Myrinet/Ethernet	InfiniBand	Ethernet	Shared Memory

Multi-core aware algorithms for collective operations:

Operation	Algorithms
Barrier	BT, Gather+Bcast, BTe, Gather+Bcast Optimized
Bcast	MST, NBFT, BFT
Scatter/v	MST, NBFT
Gather/v	MST, NBFT, NB1FT, BFT
Allgather/v	NBFT, NBBDE, BBKT, NBBKT, BTe, Gather + Bcast
Alltoall/v	NBFT, NB1FT, NB2FT, BFT
Reduce	MST, NBFT, BFT
Allreduce	NBFT, BBDE, NBBDE, BTe, Reduce + Bcast
Reduce-scatter	BBDE, NBBDE, BBKT, NBBKT, Reduce + Scatter
Scan	NBFT, OneToOne

NPB-MPJ Characteristics (10,000 SLOC (Source LOC))

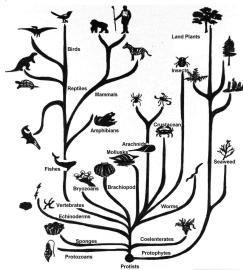
Name	Operation	SLOC	Communicat. intensiveness	Kernel	Applic.
CG	Conjugate Gradient	1000	Medium	✓	
EP	Embarrassingly Parallel	350	Low	✓	
FT	Fourier Transformation	1700	High	✓	
IS	Integer Sort	700	High	✓	
MG	Multi-Grid	2000	High	✓	
SP	Scalar Pentadiagonal	4300	Medium		✓

NAS Parallel Benchmarks NPB-MPJ

NPB-MPJ Optimization:

- JVM JIT compilation of heavy and frequent methods with runtime information
- Structured programming is the best option
 - Small frequent methods are better.
 - mapping elements from multidimensional to one-dimensional arrays (array flattening technique:
`arr3D[x][y][z] → arr3D[pos3D(lenghtx, lengthy, x, y, z)]`)
 - NPB-MPJ code refactored, obtaining significant improvements (up to 2800% performance increase)

ProtTest 3

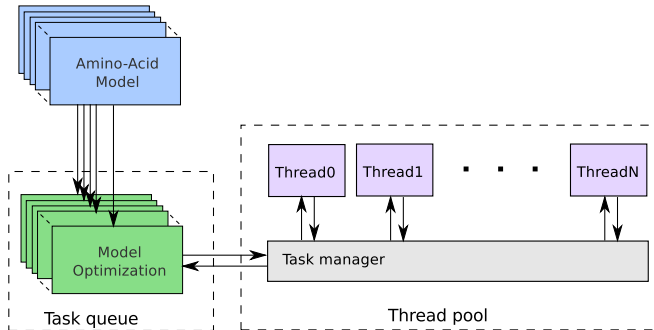


- One of the most popular tools for selecting models of protein evolution.
 - Almost 4,000 registered users.
 - Over 700 citations.
- Written in Java.
- Intensive in computational needs.
- ProtTest 3 designed to take advantage of parallel processing.

Shared Memory Implementation

Java concurrence API

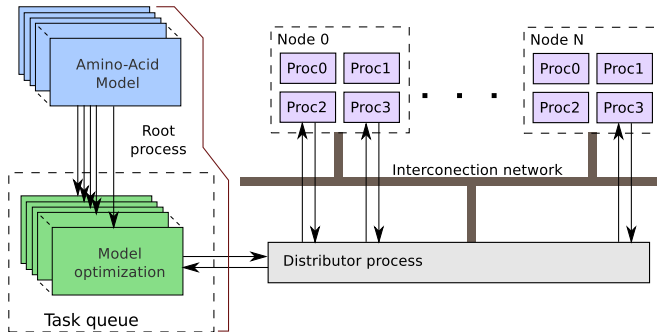
- Implementation of a thread pool.
- Dynamic task distribution over the pool.



Distributed Memory Implementation

Message Passing in Java

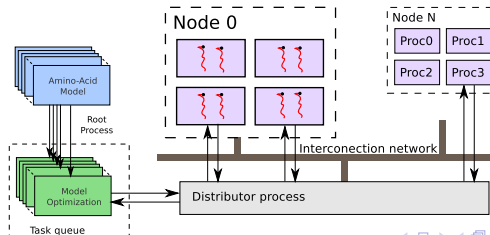
- Allow both distributions (static and dynamic).
- Includes a distributor process with a negligible workload.



Hybrid Shared/Distributed Memory Implementation

MPJ + OpenMP

- Scalability is limited by the task-based high level parallelization.
- Solution:
 - Two-level parallelism.
 - Combination of message passing with multithread computation of likelihood.
 - Implementation of a parallel version of PhyML using OpenMP.



Gadget Cosmological Simulation Project Webpage

The screenshot shows a Mozilla Firefox browser window displaying the website for Gadget-2 cosmological simulations. The browser's address bar shows the URL <http://www.mpa-garching.mpg.de/gadget/>. The page features a large, colorful image of a galaxy cluster at the top. Below the image, the text "GADGET - 2" is prominently displayed in a blue banner, followed by the subtitle "A code for cosmological simulations of structure formation". A left-hand navigation menu is visible, with categories including General, Software, Documentation, and Publications. The main content area lists several simulation projects, each accompanied by a small thumbnail image and a blue hyperlink:

- [Millennium Simulation](#)
- [Colliding disk galaxies](#)
- [Hydrodynamical simulations of cosmic structure formation](#)
- [Merging galaxies with quasar feedback](#)
- [Constrained Realizations of the Local Universe](#)
- [High-resolution simulations of a cluster of galaxies](#)

Experimental Configuration:

DAS-4 VU cluster (74 nodes)

- 2xIntel Xeon 5620 Quad-core CPU (8 cores with hyper-threading per node)
- 24 GB RAM
- InfiniBand Network 32 Gbps (Mellanox MT26428 QDR)
- Linux, OpenJDK 1.6, F-MPJ, MPJ Express, IntelMPI
- Special shared memory node (node075):
 - 4xAMD Opteron 6172 12-core (48 cores) and 128 GB RAM

Departmental x86-64 cluster (16 nodes)

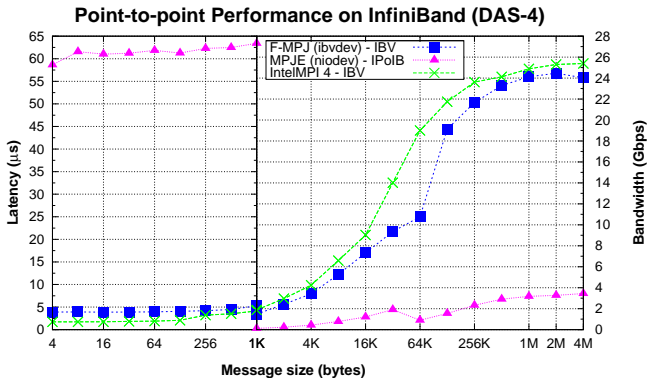
- 2xIntel Xeon 5620 Quad-core CPU (8 cores with hyper-threading per node)
- 8 GB RAM
- InfiniBand Network 16 Gbps (QLogic QLE7240 DDR)
- Linux, Sun JDK 1.6, F-MPJ, MPJ Express, OpenMPI, MVAPICH

HPC Communications Hardware

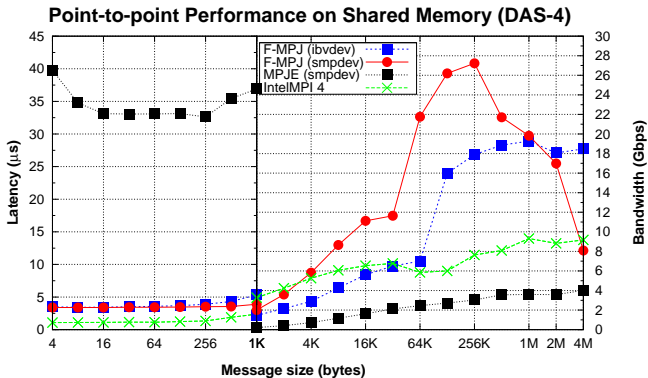
Performance of current HPC networks (Theoretical/C/Java):

	Startup latency (microseconds)	Bandwidth (Mbps)
Gig. Ethernet	50/55/60	1000/920/900
10G Ethernet	5/10/50	10000/9000/5000
10G Myrinet	1/2/30	10000/9300/4000
InfiniBand	1/2/20	16000/12000/6000
SCI	1.4/3/50	5333/2400/800

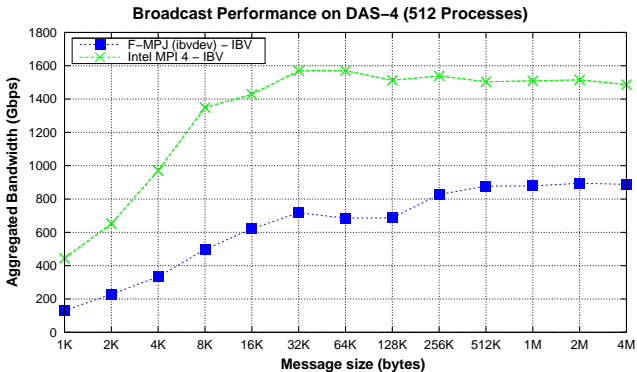
Point-to-Point Performance



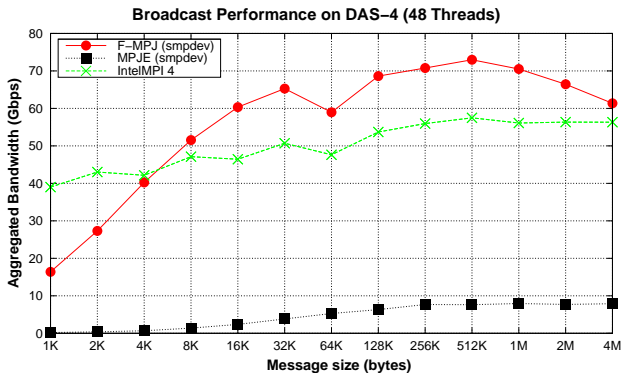
Point-to-Point Performance



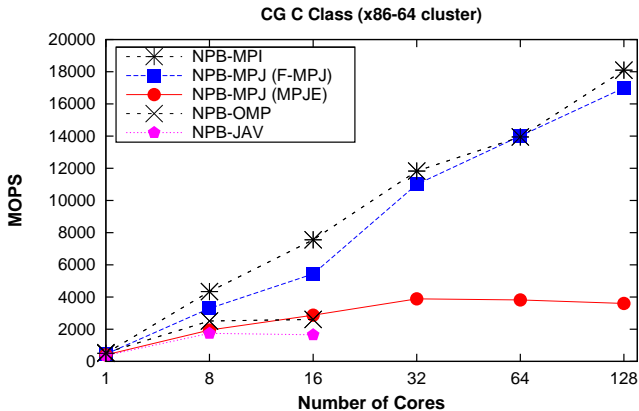
Collective Operations Performance



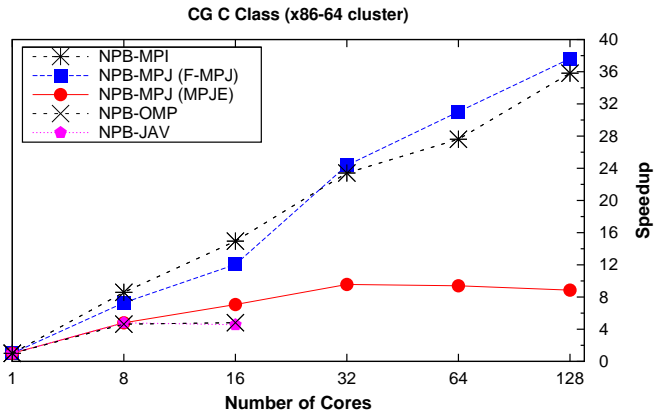
Collective Operations Performance



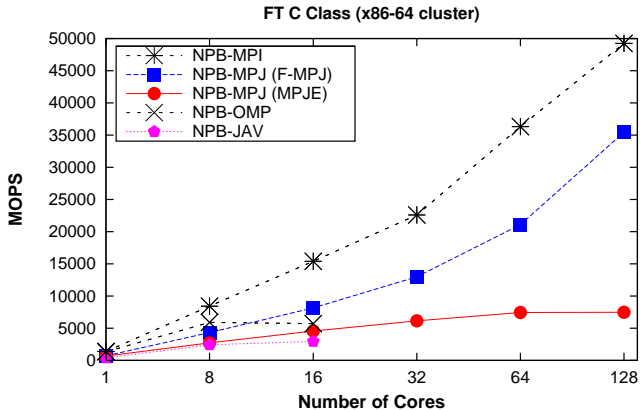
NPB-MPJ Performance



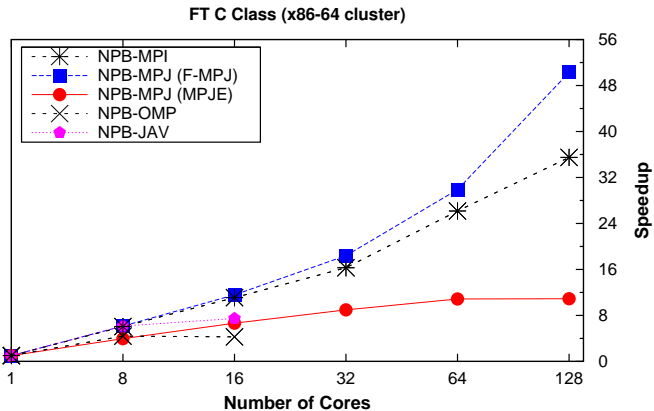
NPB-MPJ Performance



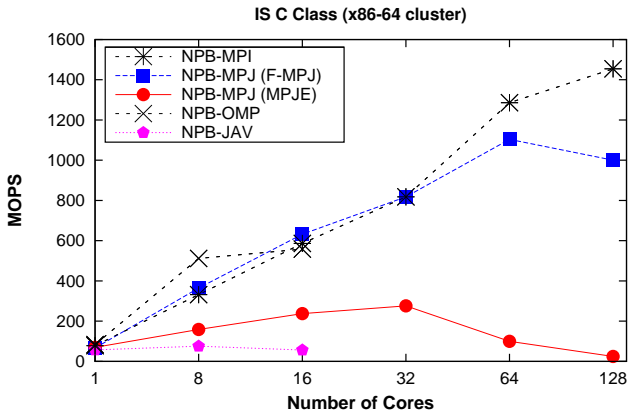
NPB-MPJ Performance



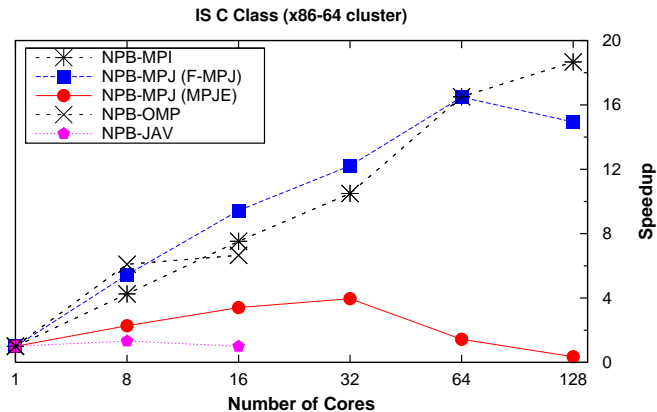
NPB-MPJ Performance



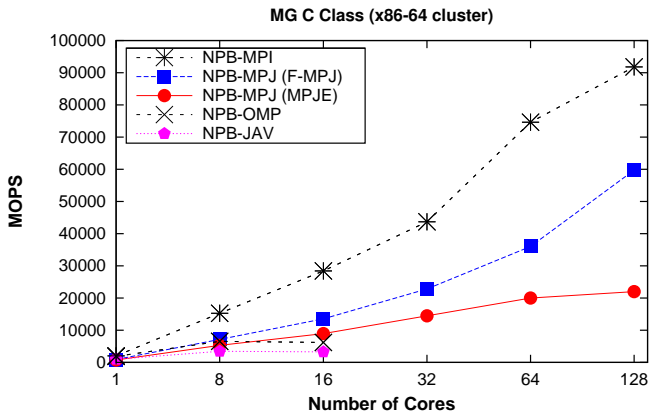
NPB-MPJ Performance



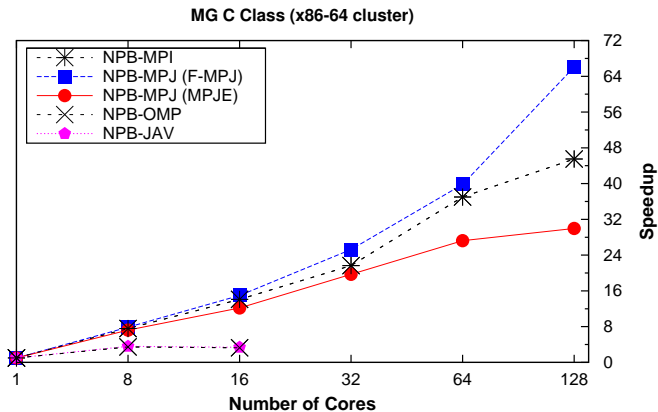
NPB-MPJ Performance



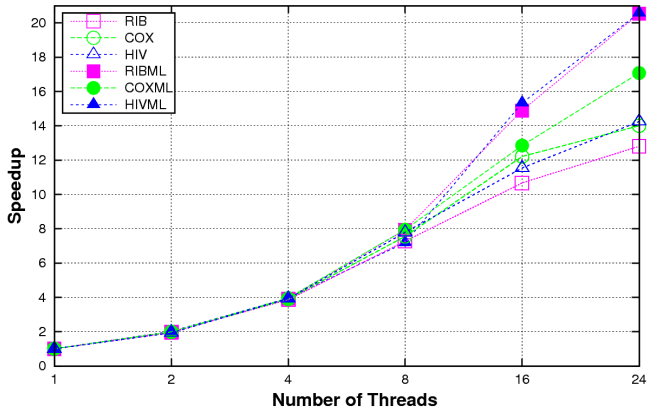
NPB-MPJ Performance



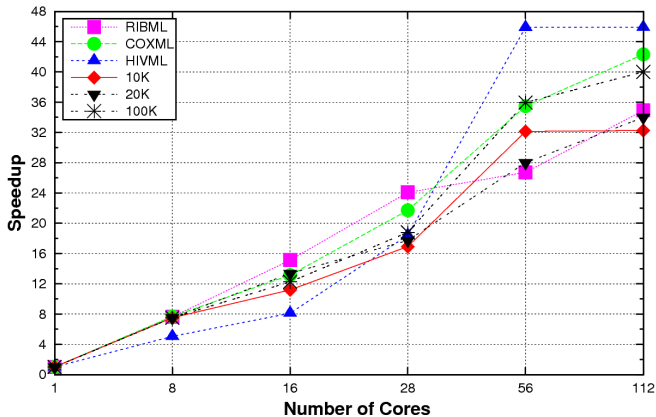
NPB-MPJ Performance



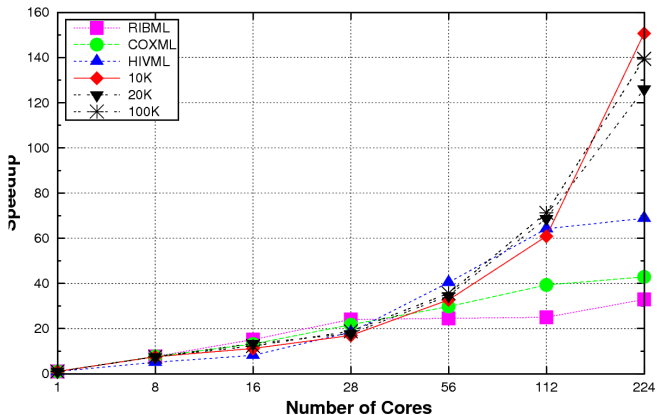
ProtTest 3: multithread implementation



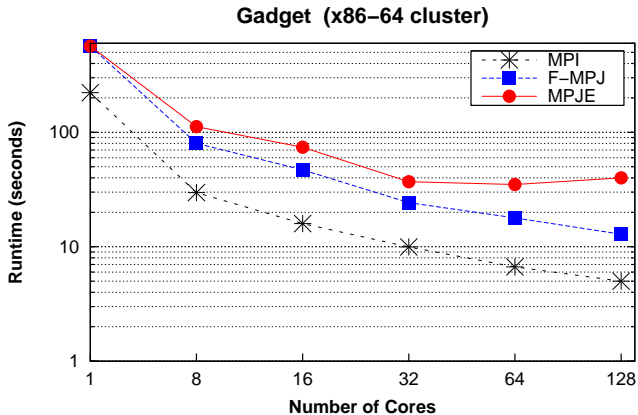
ProtTest 3: MPJ implementation



ProtTest 3: Hybrid implementation

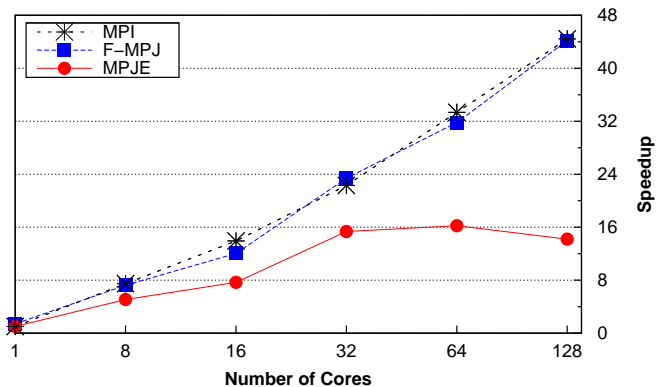


Java Gadget Performance



Java Gadget Performance

Gadget (x86-64 cluster)



Summary

- Current state of Java for HPC (interesting/feasible alternative)
- Available programming models in Java for HPC:
 - Shared memory programming
 - Distributed memory programming
 - **Distributed shared memory programming**
- **Active research** on Java for HPC (>30 projects)
- **Active work** on Java HPC projects (ESA Gaia, Petro-seismic JavaSeis...)
- ...but still not a mainstream language for HPC
- Adoption of Java for HPC:
 - It is an alternative for programming multi-core clusters (tradeoff some performance for appealing features)
 - Performance evaluations are **highly important**
 - Analysis of current projects (promotion of joint efforts)

Questions?

JAVA FOR HIGH PERFORMANCE COMPUTING: MYTH OR REALITY?

CIEMAT 2011

Guillermo López Taboada (taboada@udc.es)
Computer Architecture Group, University of A Coruña
<http://gac.udc.es/~gltaboada>