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%)

Guillermo López Taboada
Java for HPC: Myth or Reality?
### 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
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.
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.)

NPB Class B Performance on 1 core (Xeon 5060)

- CG
- EP
- FT
- IS
- MG
- SP

Speedup Relative to MPI (GNU Comp.)

- MPI (GNU Comp.)
- MPI (Intel Comp.)
- MPJ (F-MPJ)
- ProActive
- Java Threads

Guillermo López Taboada
Java for HPC: Myth or Reality?
Emerging Interest in Java for HPC

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. Thank you! Permalink Comments [3]
Current State of Java for HPC

Brian Amedro (OASIS)\(^1\), Vladimir Bodnarchouk (INRIA)\(^2\), Denis Caromel\(^1, 3\), Christian Delbe\(^1\), Fabrice Huet (OASIS)\(^1\), Guillermo López Taboada (INRIA)\(^4\)\(^, 4\) (2008)

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/MPI 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.

---

1. OASIS (INRIA Sophia Antipolis / Laboratoire I3S)  
2. INRIA - Université de Nice Sophia Antipolis - CNRS - UMR6550  
3. ActeweEn  
4. SLOOP (INRIA Sophia Antipolis)  
4. INRIA  
4. University of A Coruna - Computer Architecture Group  
University of A Coruna
## 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)
Listing 1: JOMP example

```java
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

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

Implementation approaches in Java message-passing libraries.
Listing 2: MPJ example

```java
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_WORLDRecv(message, 0, 1, MPI.OBJECT, 0, 13);
            System.out.println(message[0]);
        }
        MPI.Finalize();
    }
}
```
<table>
<thead>
<tr>
<th>Pure Java Impl.</th>
<th>Socket impl.</th>
<th>High-speed network support</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Java IO</td>
<td>Java NIO</td>
<td>Myrinet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>InfiniBand</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCI</td>
</tr>
<tr>
<td>MPJava</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jcluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parallel Java</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>mpiJava</td>
<td></td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P2P-MPI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MPJ Express</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MPJ/Ibis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JMPI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>F-MPJ</td>
<td>✓</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>

Guillermo López Taboada

Java for HPC: Myth or Reality?
Java Communication Libraries Overview

Java HPC Applications *(Develop Efficient Codes)*

Java Message-passing libraries *(Scalable Algorithms)*

Low-level messaging libraries *(MPJ Devices)*

HPC Hardware
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).
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).
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).
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

```java
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

<table>
<thead>
<tr>
<th>MPJ Applications</th>
<th>F-MPJ Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>device layer</td>
<td></td>
</tr>
<tr>
<td>OMXDEV</td>
<td>JNI</td>
</tr>
<tr>
<td>IBVDEV</td>
<td>Java Sockets</td>
</tr>
<tr>
<td>NIODEV/IODEV</td>
<td>Java Threads</td>
</tr>
<tr>
<td>SMPDEV</td>
<td></td>
</tr>
<tr>
<td>JVM</td>
<td></td>
</tr>
<tr>
<td>JNI</td>
<td></td>
</tr>
<tr>
<td>Java Sockets</td>
<td></td>
</tr>
<tr>
<td>Java Threads</td>
<td></td>
</tr>
<tr>
<td>Native comms</td>
<td></td>
</tr>
<tr>
<td>Open-MX</td>
<td>IBV</td>
</tr>
<tr>
<td>TCP/IP</td>
<td></td>
</tr>
<tr>
<td>Myrinet/Ethernet</td>
<td>InfiniBand</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Shared Memory</td>
</tr>
</tbody>
</table>

Guillermo López Taboada
Java for HPC: Myth or Reality?
## Multi-core aware algorithms for collective operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier</td>
<td>BT, Gather+Bcast, BTe, Gather+Bcast Optimized</td>
</tr>
<tr>
<td>Bcast</td>
<td>MST, NBFT, BFT</td>
</tr>
<tr>
<td>Scatter/v</td>
<td>MST, NBFT</td>
</tr>
<tr>
<td>Gather/v</td>
<td>MST, NBFT, NB1FT, BFT</td>
</tr>
<tr>
<td>Allgather/v</td>
<td>NBFT, NBBDE, BBKT, NBBKT, BTe, Gather + Bcast</td>
</tr>
<tr>
<td>Alltoall/v</td>
<td>NBFT, NB1FT, NB2FT, BFT</td>
</tr>
<tr>
<td>Reduce</td>
<td>MST, NBFT, BFT</td>
</tr>
<tr>
<td>Allreduce</td>
<td>NBFT, BBDE, NBBDE, BBKT, BTe, Reduce + Bcast</td>
</tr>
<tr>
<td>Reduce-scatter</td>
<td>BBDE, NBBDE, BBKT, NBBKT, Reduce + Scatter</td>
</tr>
<tr>
<td>Scan</td>
<td>NBFT, OneToOne</td>
</tr>
</tbody>
</table>
## NPB-MPJ Characteristics (10,000 SLOC (Source LOC))

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

GADGET - 2
A code for cosmological simulations of structure formation

- 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

Guillermo López Taboada
Java for HPC: Myth or Reality?
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

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

<table>
<thead>
<tr>
<th></th>
<th>Startup latency (microseconds)</th>
<th>Bandwidth (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gig. Ethernet</td>
<td>50/55/60</td>
<td>1000/920/900</td>
</tr>
<tr>
<td>10G Ethernet</td>
<td>5/10/50</td>
<td>10000/9000/5000</td>
</tr>
<tr>
<td>10G Myrinet</td>
<td>1/2/30</td>
<td>10000/9300/4000</td>
</tr>
<tr>
<td>InfiniBand</td>
<td>1/2/20</td>
<td>16000/12000/6000</td>
</tr>
<tr>
<td>SCI</td>
<td>1.4/3/50</td>
<td>5333/2400/800</td>
</tr>
</tbody>
</table>
Motivation
Java for High Performance Computing
Java HPC Codes
Performance Evaluation
Conclusions

Experimental Configuration
F-MPJ Performance
Java Performance for HPC

Point-to-Point Performance

Point-to-point Performance on InfiniBand (DAS-4)

Latency (µs)

F-MPJ (ibvdev) - IBV
MPJE (niodev) - IPoIB
IntelMPI 4 - IBV

Message size (bytes)

Bandwidth (Gbps)

Guillermo López Taboada
Java for HPC: Myth or Reality?
Point-to-Point Performance on Shared Memory (DAS-4)

- F-MPJ (ibvdev)
- F-MPJ (smpdev)
- MPJE (smpdev)
- IntelMPI 4

Message size (bytes)

Latency (µs)

Bandwidth (Gbps)
Collective Operations Performance

Broadcast Performance on DAS-4 (512 Processes)

- F-MPJ (ibvdev) - IBV
- Intel MPI 4 - IBV

Message size (bytes)

Aggregated Bandwidth (Gbps)
Collective Operations Performance

Broadcast Performance on DAS–4 (48 Threads)

- F-MPJ (smpdev)
- MPJE (smpdev)
- IntelMPI 4

Message size (bytes) vs. Aggregated Bandwidth (Gbps)
NPB-MPJ Performance

CG C Class (x86-64 cluster)

- NPB-MPI
- NPB-MPJ (F-MPJ)
- NPB-MPJ (MPJE)
- NPB-OMP
- NPB-JAV

Guillermo López Taboada
Java for HPC: Myth or Reality?
NPB-MPJ Performance

CG C Class (x86-64 cluster)

Number of Cores

Speedup

NPB-MPI
NPB-MPJ (F-MPJ)
NPB-MPJ (MPJE)
NPB-OMP
NPB-JAV
NPB-MPJ Performance

FT C Class (x86-64 cluster)

- NPB-MPI
- NPB-MPJ (F-MPJ)
- NPB-MPJ (MPJE)
- NPB-OMP
- NPB-JAV

MOPS vs Number of Cores

Guillermo López Taboada
NPB-MPJ Performance

FT C Class (x86-64 cluster)

- NPB-MPI
- NPB-MPJ (F-MPJ)
- NPB-MPJ (MPJE)
- NPB-OMP
- NPB-JAV

Guillermo López Taboada
Java for HPC: Myth or Reality?
NPB-MPJ Performance

Guillermo López Taboada  |  Java for HPC: Myth or Reality?
NPB-MPJ Performance

IS C Class (x86-64 cluster)

- NPB-MPI
- NPB-MPJ (F-MPJ)
- NPB-MPJ (MPJE)
- NPB-OMP
- NPB-JAV

Number of Cores

Speedup

1 8 16 32 64 128

0 4 8 12 16 20

Guillermo López Taboada
Java for HPC: Myth or Reality?
NPB-MPJ Performance

MG C Class (x86-64 cluster)

- NPB-MPI
- NPB-MPJ (F-MPJ)
- NPB-MPJ (MPJE)
- NPB-OMP
- NPB-JAV

Guillermo López Taboada
Java for HPC: Myth or Reality?
NPB-MPJ Performance

MG C Class (x86-64 cluster)

---

<table>
<thead>
<tr>
<th>Number of Cores</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
</tr>
</tbody>
</table>

NPB-MPI
NPB-MPJ (F-MPJ)
NPB-MPJ (MPJE)
NPB-OMP
NPB-JAV

Guillermo López Taboada
Java for HPC: Myth or Reality?
ProtTest 3: multithread implementation

![Graph showing speedup vs number of threads for different benchmarks.](image-url)
ProtTest 3: MPJ implementation

![Graph showing speedup vs number of cores for different configurations. The graph includes multiple lines representing different MPJ implementations and data sizes.](image_url)
ProtTest 3: Hybrid implementation
Java Gadget Performance

Gadget (x86–64 cluster)

Runtime (seconds)
Number of Cores

Gadget  (x86−64 cluster)

MPI
F−MPJ
MPJE

Runtime (seconds)
Number of Cores

Guillermo López Taboada
Java for HPC: Myth or Reality?
Java for HPC: Myth or Reality?

Java Gadget Performance

Gadget (x86–64 cluster)

Number of Cores

Speedup

- MPI
- F-MPJ
- MPJE

Guillermo López Taboada
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/~gttaboada