MPJ Express Meets *Gadget*
Towards a Java Code for Cosmological Simulations

Mark Baker*, Bryan Carpenter, and **Aamir Shafi**

DSG, University of Portsmouth
http://dsg.port.ac.uk
*ACET, University of Reading
http://acet.rdg.ac.uk

Distributed Systems Group

Presentation Outline

- Introduction to Java for HPC,
- The Gadget-2 Code,
- Porting Gadget-2 to Java,
- Performance Evaluation,
- Conclusions and Future Work.

Java for HPC

- Java was released by Sun in 1996:
  - A mainstream language in software industry,
- Attractive features include:
  - Portability,
  - Automatic garbage collection,
  - Type-safety at compile time and runtime,
  - Built-in support for multi-threading:
    - A possible option to provide *nested parallelism* on multi-core systems,
- Various people argued that Java is a good option for HPC:
  - HPC has history of *novel* languages,
  - Java incorporates, at least, some of these ideas,
- Performance:
  - Just-In-Time compilers convert source code to byte code,
  - Modern JVMs perform compilation from byte code to native machine code on the fly,
  - But Java has safety features that may limit performance.

Java for HPC

- To date, these arguments have not convinced too many practicing computational scientists:
  - The scarcity of high-profile number-crunching codes implemented in Java does not help the case,
- To address this, we produced a Java version of massively parallel structure formation code Gadget-2:
  - An experiment to evaluate and compare the performance of Java and C versions.

Presentation Outline

- Introduction to Java for HPC,
- The Gadget-2 Code,
- Porting Gadget-2 to Java,
- Performance Evaluation,
- Conclusions and Future Work.

Gadget-2

- Cosmological simulations play a vital role in our understanding of structure formation,
- Gadget-2 is a production-quality code for cosmological N-body (and hydrodynamic) computations:
  - Written by Volker Springel, of the Max Plank Institute for Astrophysics, Garching,
- It is written in the C language–already parallelized using MPI,
- Computational capability and *combined* distributed memory of clusters is very attractive for cosmological simulations.

Distributed Systems Group

Millennium Simulation

- Versions have been used in various research papers in astrophysics literature, including the “Millennium Simulation”,
- Follows evolution of 10^{10} dark matter “particles” from early Universe (z = 127) to current day,
- Performed on 512 nodes of IBM p690:
  - Used 1Terabytes of distributed memory,
  - 350,000 CPU hours – 28 days elapsed time,
  - Floating point performance around 0.2 TFLOPS,
- Around 20Terabytes total data generated.

Distributed Systems Group

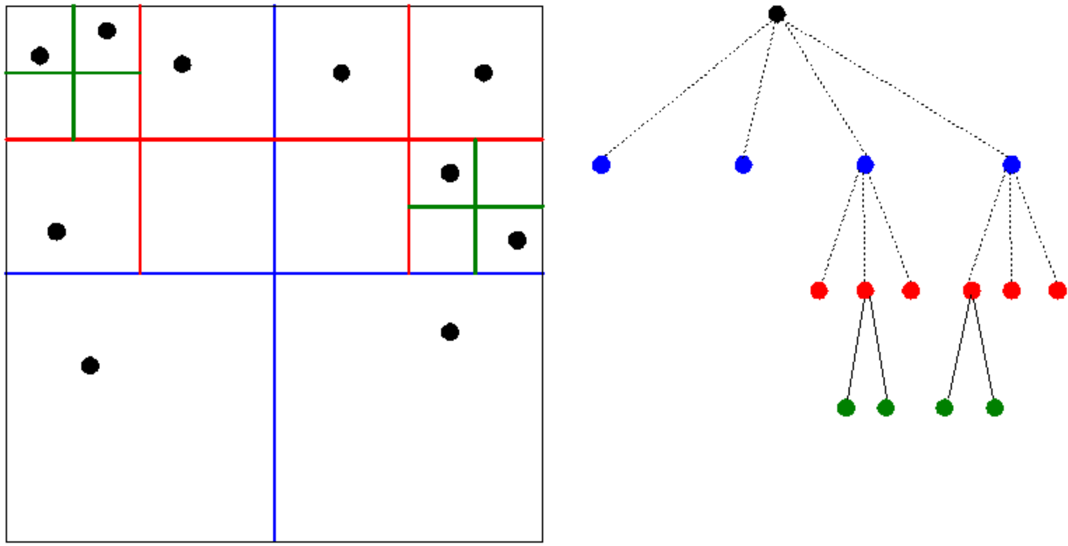
D S G

Dynamics in Gadget

- Gadget is "just" simulating the movement of (a lot of) representative particles under the influence of Newton's law of gravity:
  - Plus some hydrodynamic forces, but these don't affect dominant *dark matter*,
- Two techniques to calculate gravitational forces between particles:
  - Barnes Hut Tree algorithm,
  - TreePM algorithm.

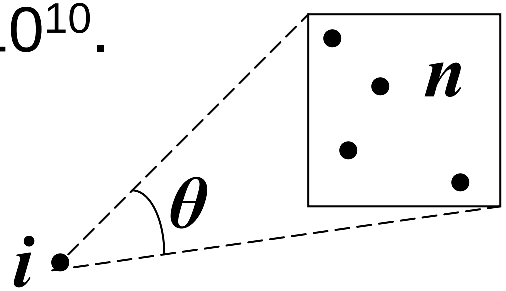
Barnes Hut Tree

- Imagine all particles in the universe is encapsulated in a cube,
- First divide cubical region of 3D space into $2^3 = 8$ regions, halving each dimension,
- For every sub-region that contains any particles, divide again recursively to make an *oct-tree*, until "leaf" regions have at most one particle,
- A 2D example shown in the figure.



Barnes Hut Force Computation

- To compute the force on a particle *i*, traverse tree starting from root:
  - if a node *n* is "distant from" *i*, just add contribution to force on *i* from centre of mass of *n* – no need to visit children of *n*,
  - if node *n* is "close to" *i*, visit children of *n* and recurse,
  - The definition is "distant from" and "close to" depend on the opening angle shown in the figure,
- Complexity:
  - On average, number of nodes "opened" to compute force on *i* is *O(*log *N)*, as opposed to visiting *O(N)* particles in naïve algorithm,
- A huge win when $N \approx 10^{10}$.



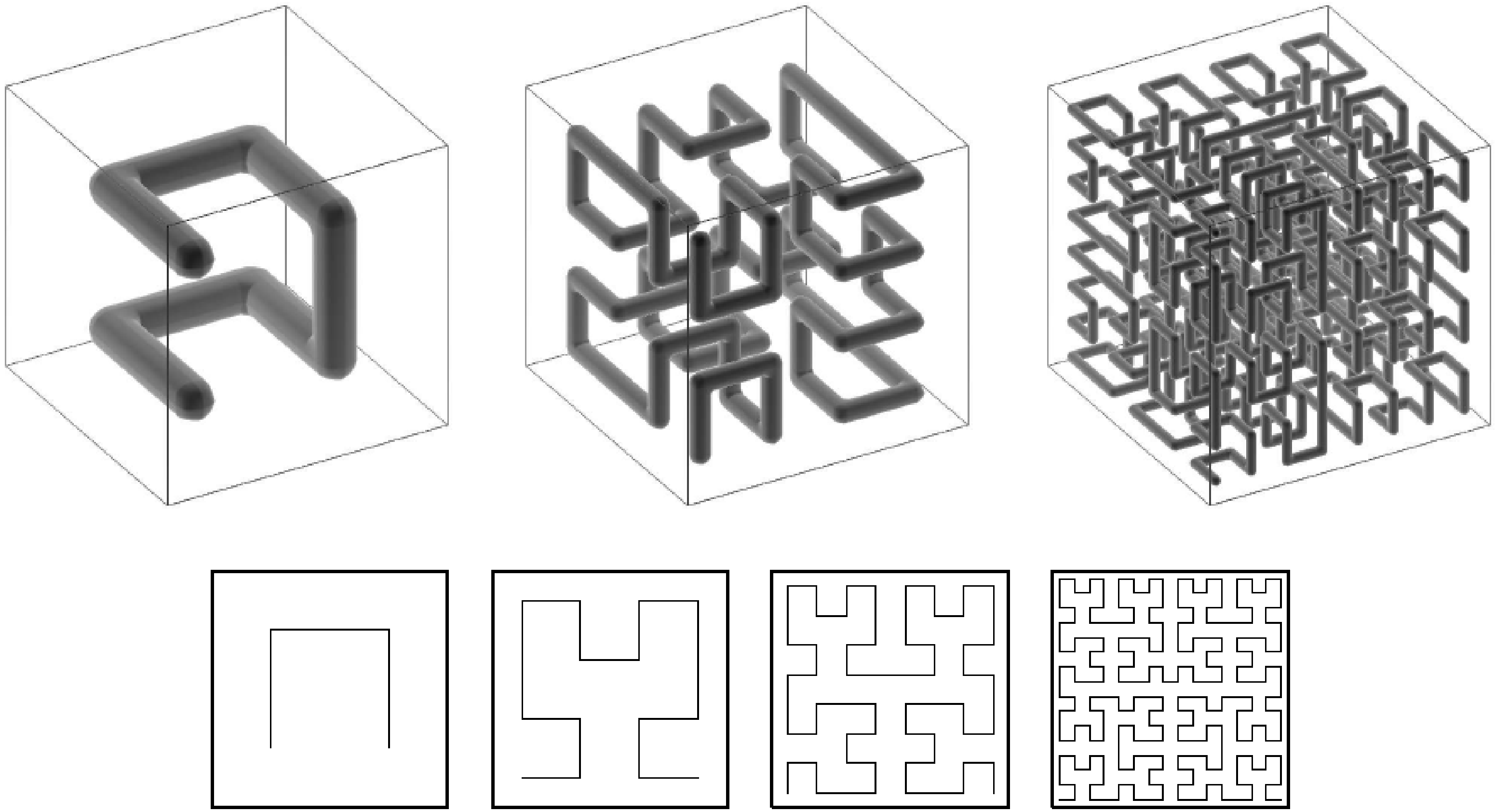
Distributed Systems Group

Domain Decomposition

- Need to divide space and/or particle set into *domains*, where each domain is handled by a single processor:
  - Cannot divide space or particle evenly,

- Uses a space-filling curve called *Peano-Hilbert* Curve.

Peano-Hilbert Curve



- Picture borrowed from **http://www.mpa-garching.mpg.de/gadget/gadget2-paper.pdf**

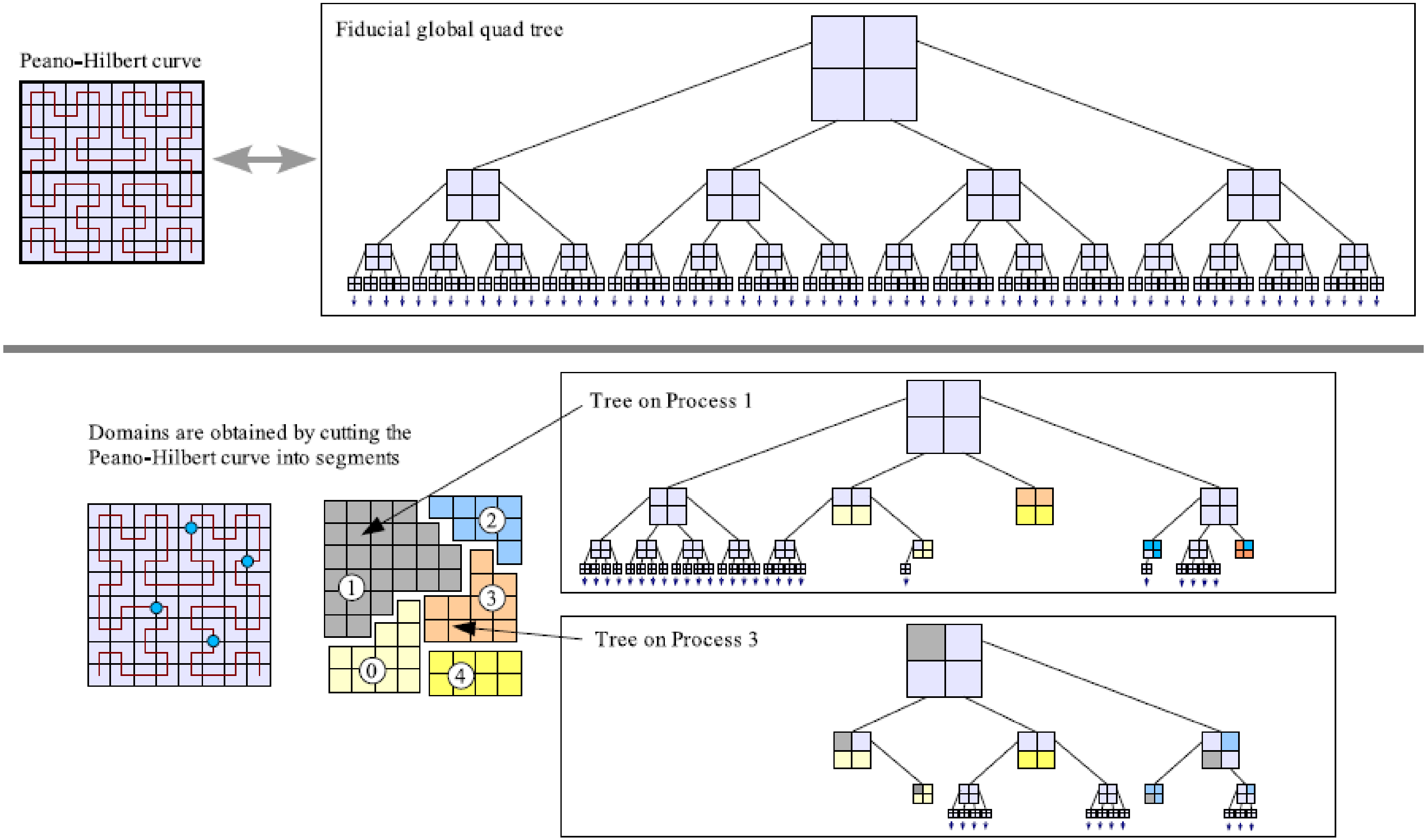
Decomposition based on P-H Curve

- Peano-Hilbert Curve:
  - Gadget applies the recursion 20 times, logically dividing space into up to $2^{20} \times 2^{20} \times 2^{20}$ ``cells'' on the Peano-Hilbert curve.
  - Then can label each cell by its location along the Peano-Hilbert curve–2^{60} possible locations comfortably fit into a 64-bit word.
- Because "number of cells'' are greater than "number of particles'', segments of linear P-H curve sparsely populated.
- Sort particles by their Peano-Hilbert key, then divide evenly into *P* domains.
  - Intuitively – stretch out the P-H curve with particles dotted along it; segment it into *P* parts where each part has the same number of particles.

Distributed BH Tree

- Particles along the Peano-Hilbert curve could map easily to Barnes Hut tree,
- Pseudo-particles:
  - Remotely owned particles,
- Gravitational contributions from pseudo-particles:
  - Import and export of particles,
- Next slide.

Distribution of BH Tree in Gadget



- Ibid.

Presentation Outline

- Introductin to Java for HPC,
- The Gadget-2 Code,
- Porting Gadget-2 to Java,
- Performance Evaluation,
- Conclusions and Future Work.

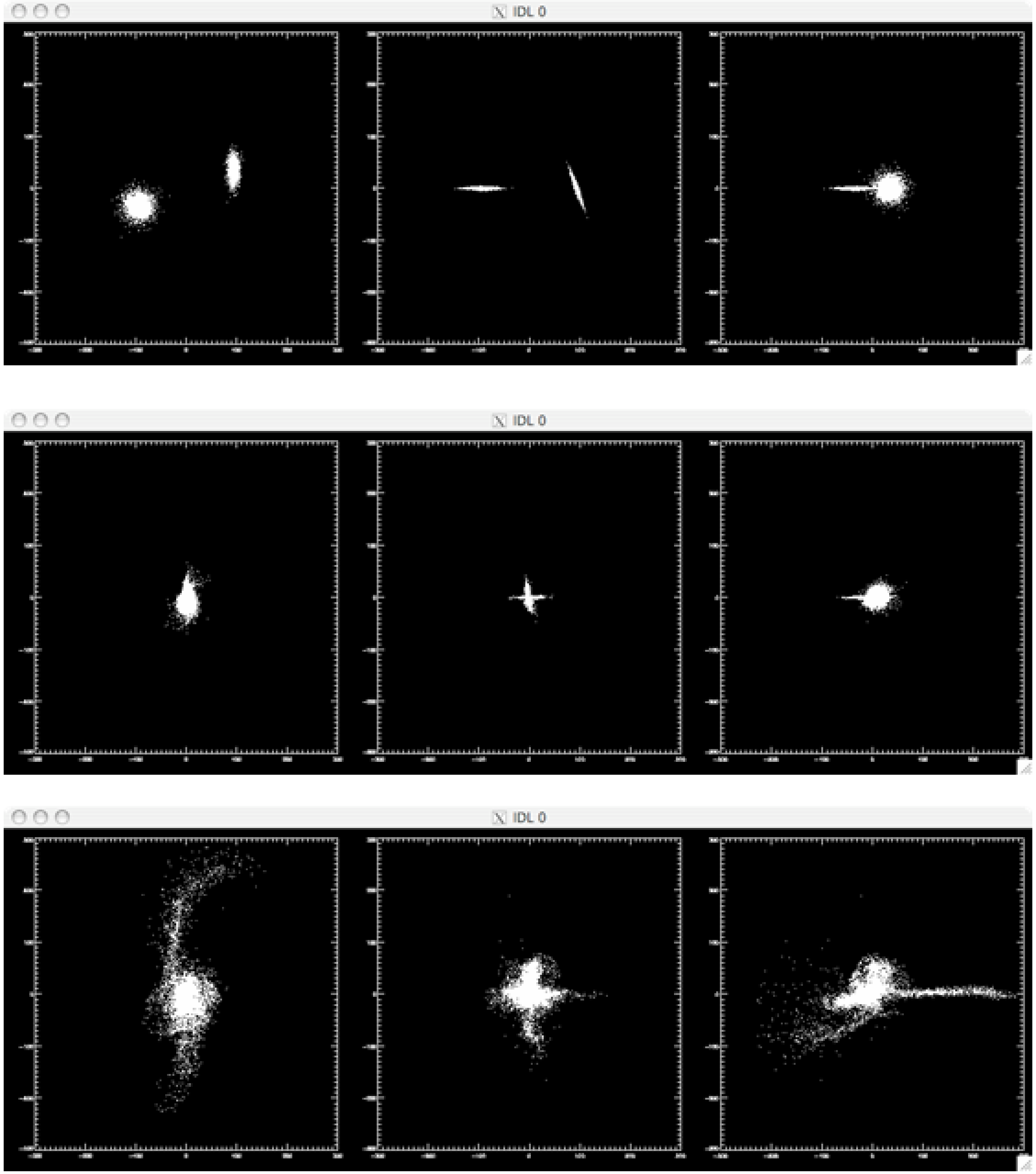
Porting Gadget-2 to Java

- Manually translated 17K lines of C code,
- Dependencies on:
  - MPI library for parallelization:
    - Replace MPI calls with MPJ Express,
  - GNU scientific library (but only a handful of functions):
    - The required methods were hand translated to Java,
  - FFTW – library for parallel Fourier transforms:
    - Not needed because we disabled TreePM algorithm,
- We have successfully run *Colliding Galaxies* and *Cluster Formation* example simulations:
  - These use pure Dark Matter – hydrodynamics code not yet tested.

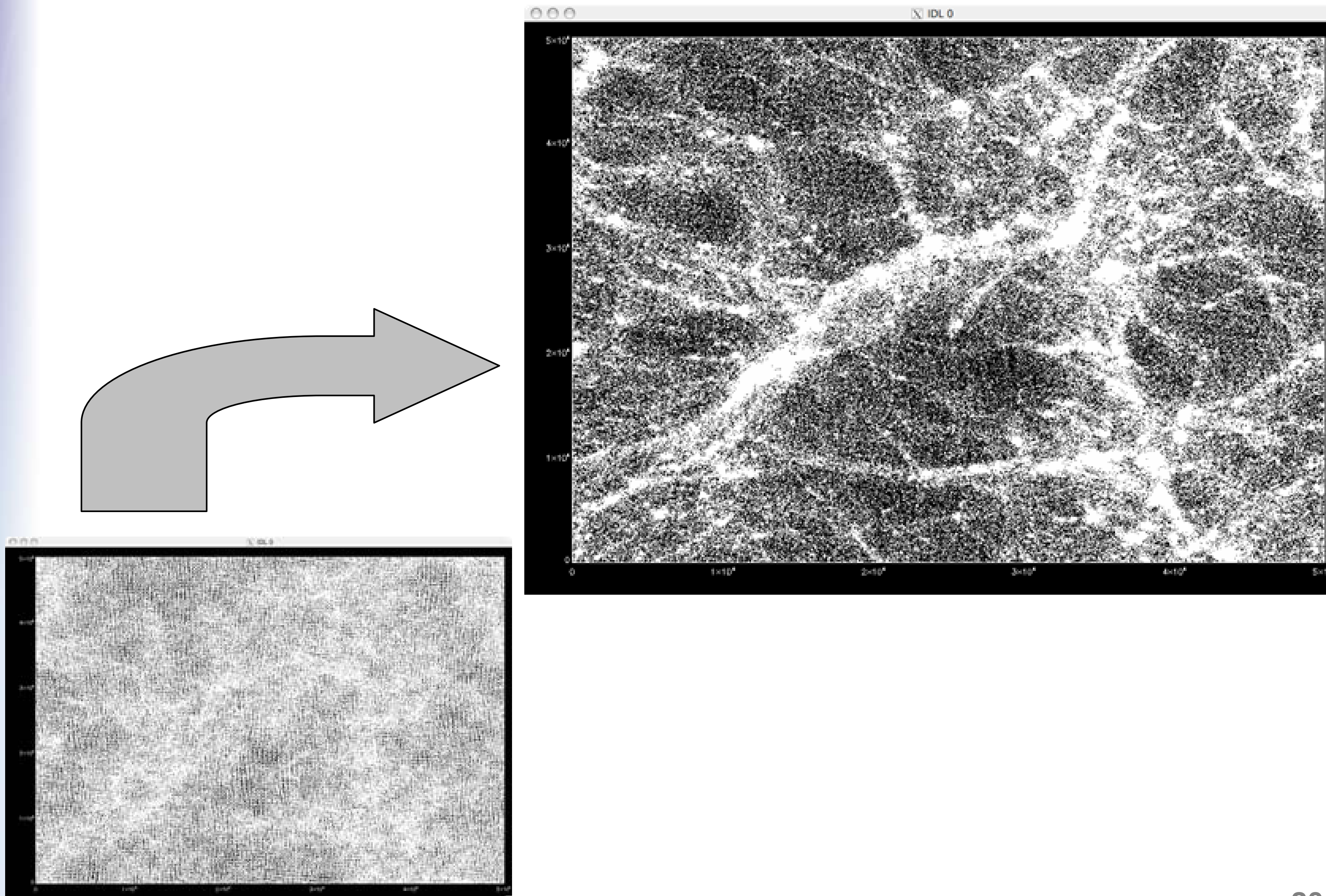
Introduction to MPJ Express

- MPJ Express is a Java messaging system:
  - Thread-safe communication devices using Java NIO and Myrinet:
    - Maintain compatibility with Java threads,
  - Portable bootstrapping mechanism,
  - "Full" implementation Java MPI bindings defined by Java Grande,
  - Will eventually supersede mpiJava,
- MPJ Express released in 2005,
- Realized we need a realistic exemplar code to drive further improvements of our software.

Colliding Galaxies Simulation



Cluster Formation Simulation



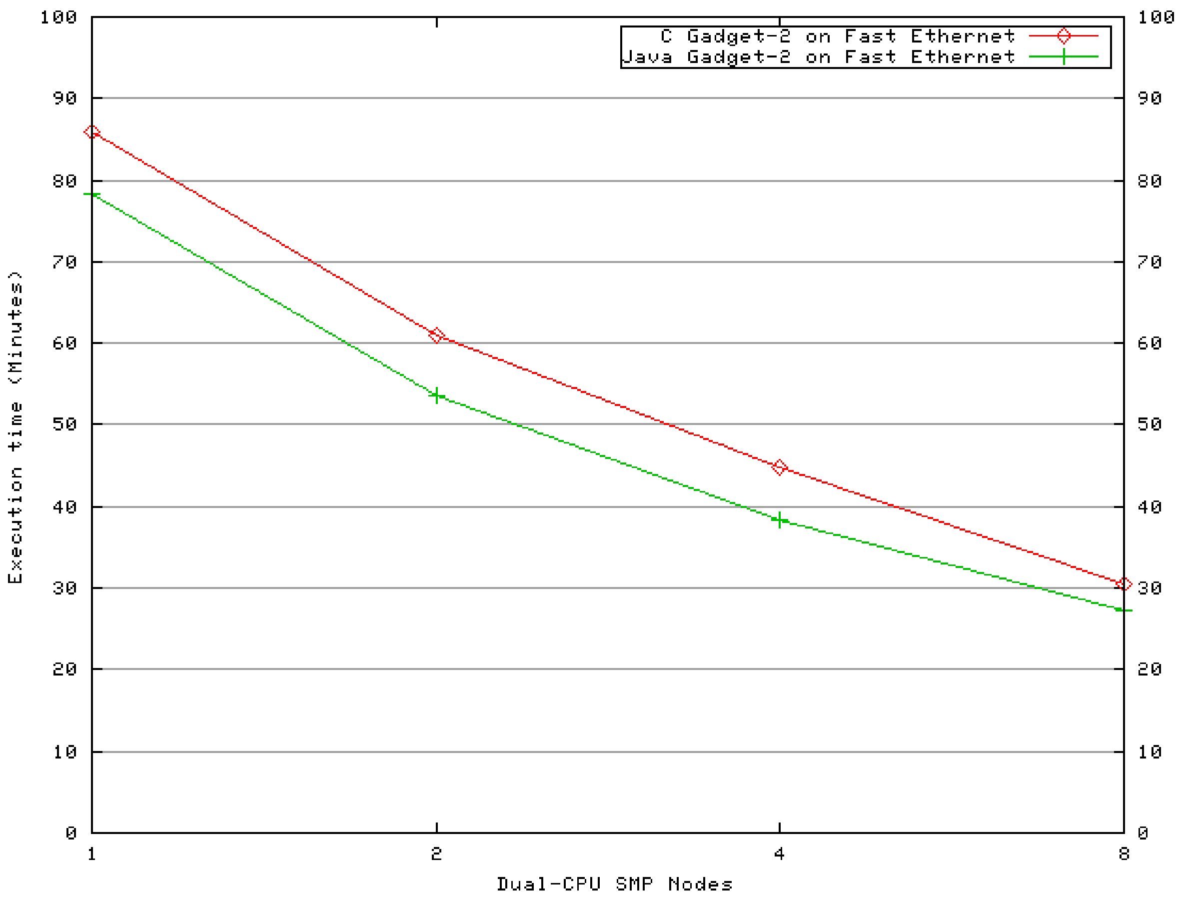
Presentation Outline

- Introduction to Java for HPC,
- The Gadget-2 Code,
- Porting Gadget-2 to Java,
- Performance Evaluation,
- Conclusions and Future Work.

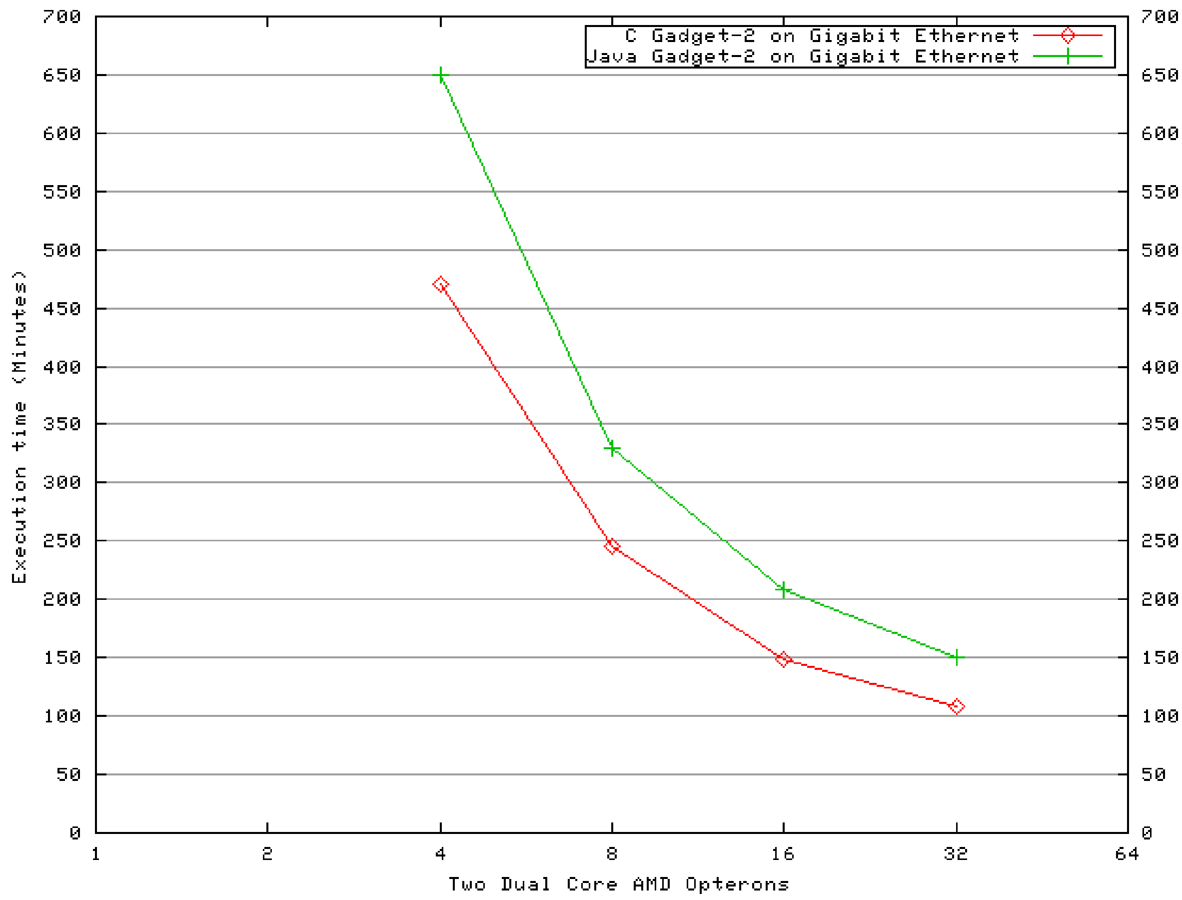
Java Optimizations

- Initial benchmarking revealed that the Java code is slower by a factor of 2-3,
- Custom serialization and de-serialization:
  - Replacing Java object communication with primitive datatypes,
- Flattening sensitive data-structures in the hope of exploiting processor cache efficiently,
  - Many data structures contained object arrays and consecutive objects are not consecutive in memory,
- Avoiding expensive array operations,
- Improved collective algorithms in MPJ Express.

Execution Time for the Colliding Galaxies Simulation



Execution Time for the Cluster Formation Simulation



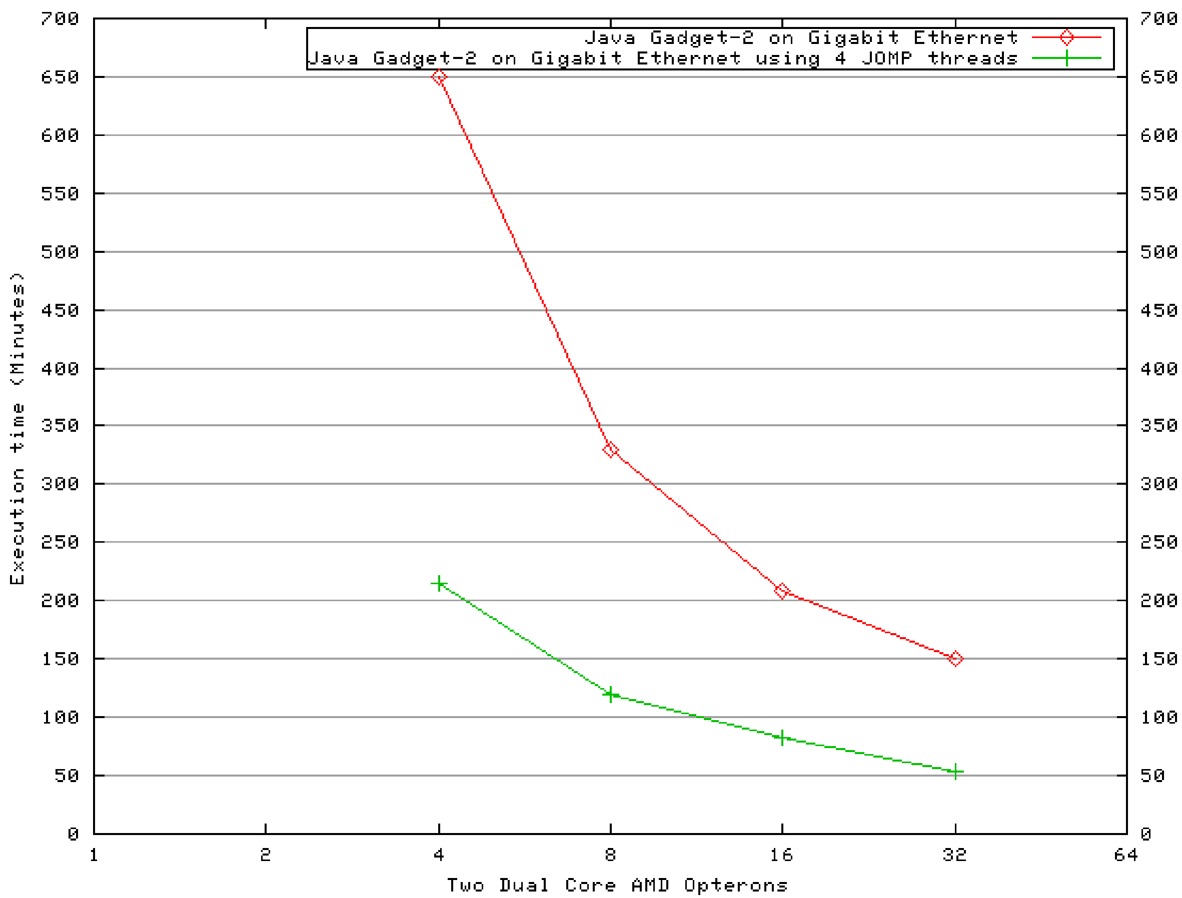
Presentation Outline

- Introduction to Java for HPC,
- The Gadget-2 Code,
- Porting Gadget-2 to Java,
- Performance Evaluation,
- Conclusions and Future Work.

Nested Parallelism in Java Gadget-2

- Recently produced Java Gadget-2 version that exploits nested parallelism on multi-core systems:
  - –Java or Java OpenMP (JOMP) threads in gravity calculation stages:
    - Overlapping computation and communication,
  - –Thoughts on OpenMP,
- Next slide.

Nested Parallelism in Java Gadget-2



Distributed Systems Group

Conclusions and Future Work

- Java has attractive features for HPC:
  - Threading might play an important role in future for multi-core systems,
  - Computational performance is improving,
- Future Work:
  - Address memory footprint concerns:
    - Biggest simulation with Java, so far, contained 56 million particles in total with 3.5 million particles on each node,
    - Millennium Simulation contained 10 billion particles with 20 million particles,
  - Release Java Gadget-2 in near future,
  - Continue improving MPJ Express.

Distributed Systems Group

Questions

?