

Scalable Topology Aware Object Mapping for Large Supercomputers

Abhinav S Bhatele

Preliminary Examination, October 30th, 2008

Outline

Demonstrate that topology aware mapping has become important again for 3D mesh topologies

Study the effects of contention on message latencies for different interconnects

Develop an automatic topology-aware mapping framework useful for application developers

- The Mapping Problem
- Motivation
- Previous Work

- Evaluative Studies
 - Quantifying message latencies
 - Characterizing applications

- Topology Aware Mapping
 - Topology Manager API
 - Application-specific Mapping
 - Automatic Mapping Framework

Outline

- The Mapping Problem
- Motivation
- Previous Work
- Evaluative Studies
 - Quantifying message latencies
 - Characterizing applications
- Topology Aware Mapping
 - Topology Manager API
 - Application-specific Mapping
 - Automatic Mapping Framework

Demonstrate that topology aware mapping has become important again for 3D mesh topologies

Study the effects of contention on message latencies for different interconnects

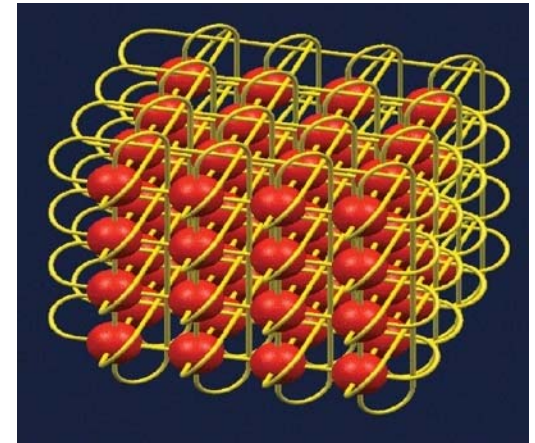
Develop an automatic topology-aware mapping framework useful for application developers

The Mapping Problem

- Given a set of communicating parallel “entities”, map them onto physical processors
- Entities
 - COMM_WORLD ranks in case of an MPI program
 - Objects in case of a Charm++ program
- Aim
 - Balance load
 - Minimize communication traffic

Target Machines

- 3D torus/mesh interconnects
- Blue Gene/P at ANL:
 - 40,960 nodes, torus dimensions 32 x 32 x 40 (?)
- XT4 (Jaguar) at ORNL:
 - 8,064 nodes, torus dimensions 21 x 16 x 24
- Other networks
 - Fat-trees: Infiniband, Federation
 - Kautz graph: SiCortex



Motivation

- Consider a 3D mesh/torus interconnect
- Message latencies can be modeled by

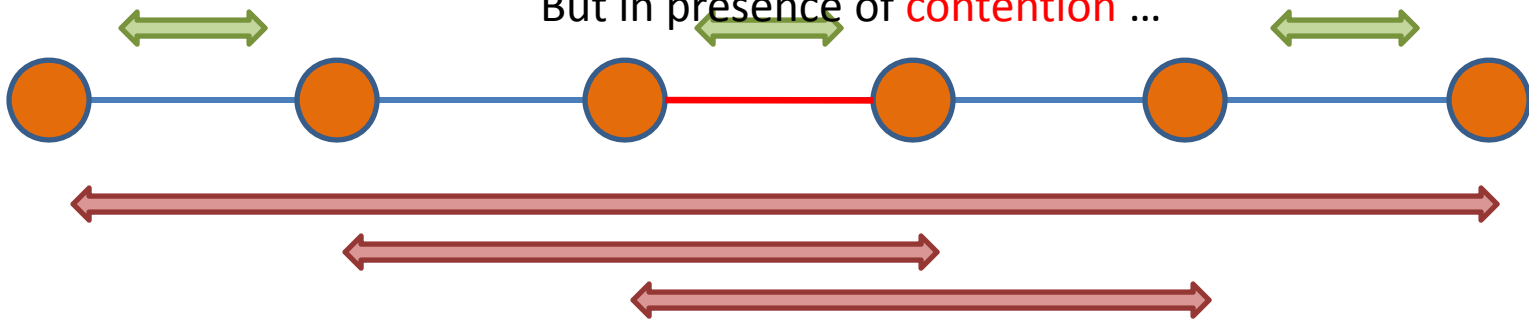
$$(L_f/B) \times D + L/B$$

L_f = length of flit, B = bandwidth,

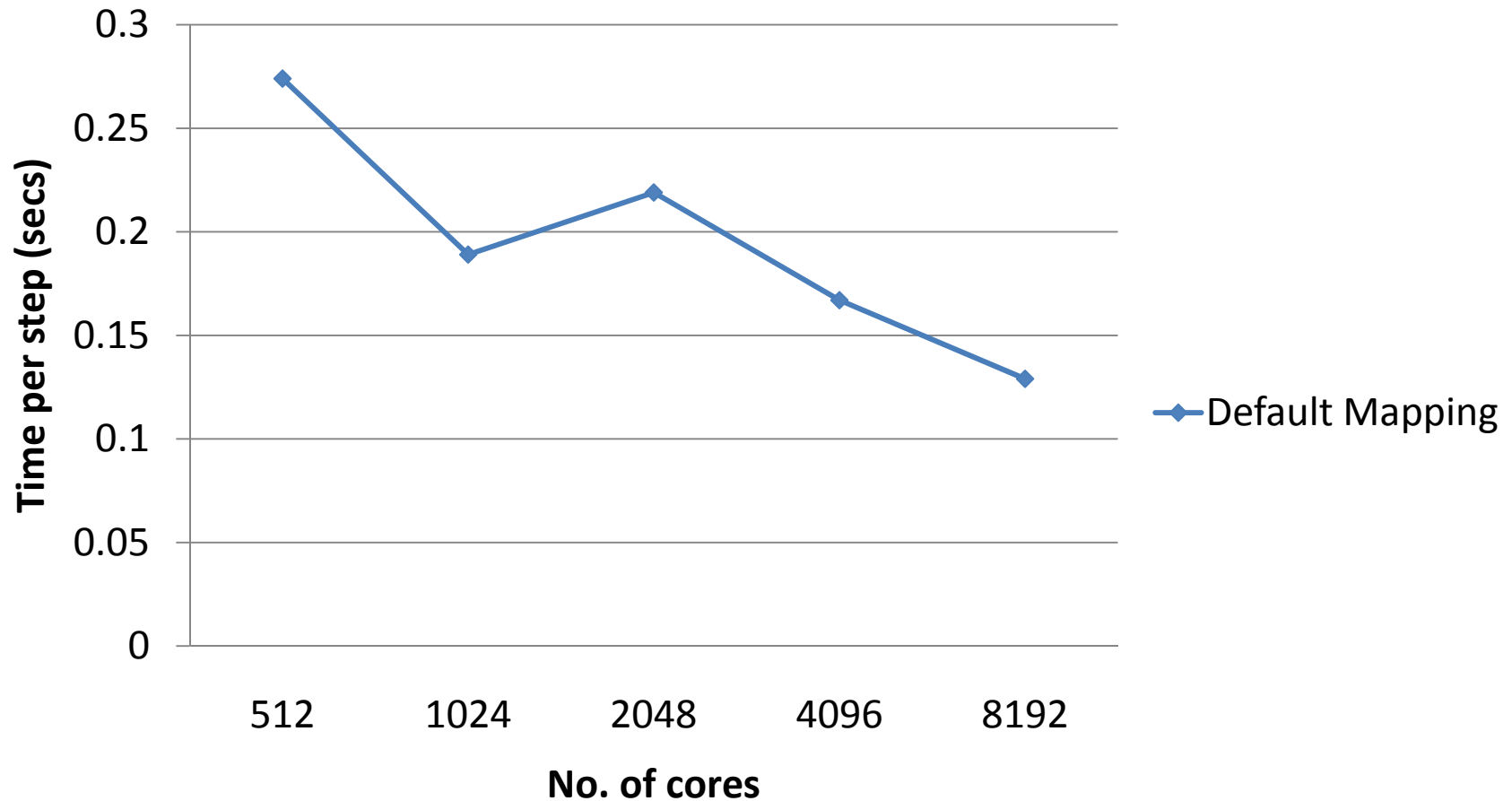
D = hops, L = message size

When $(L_f * D) \ll L$, first term is negligible

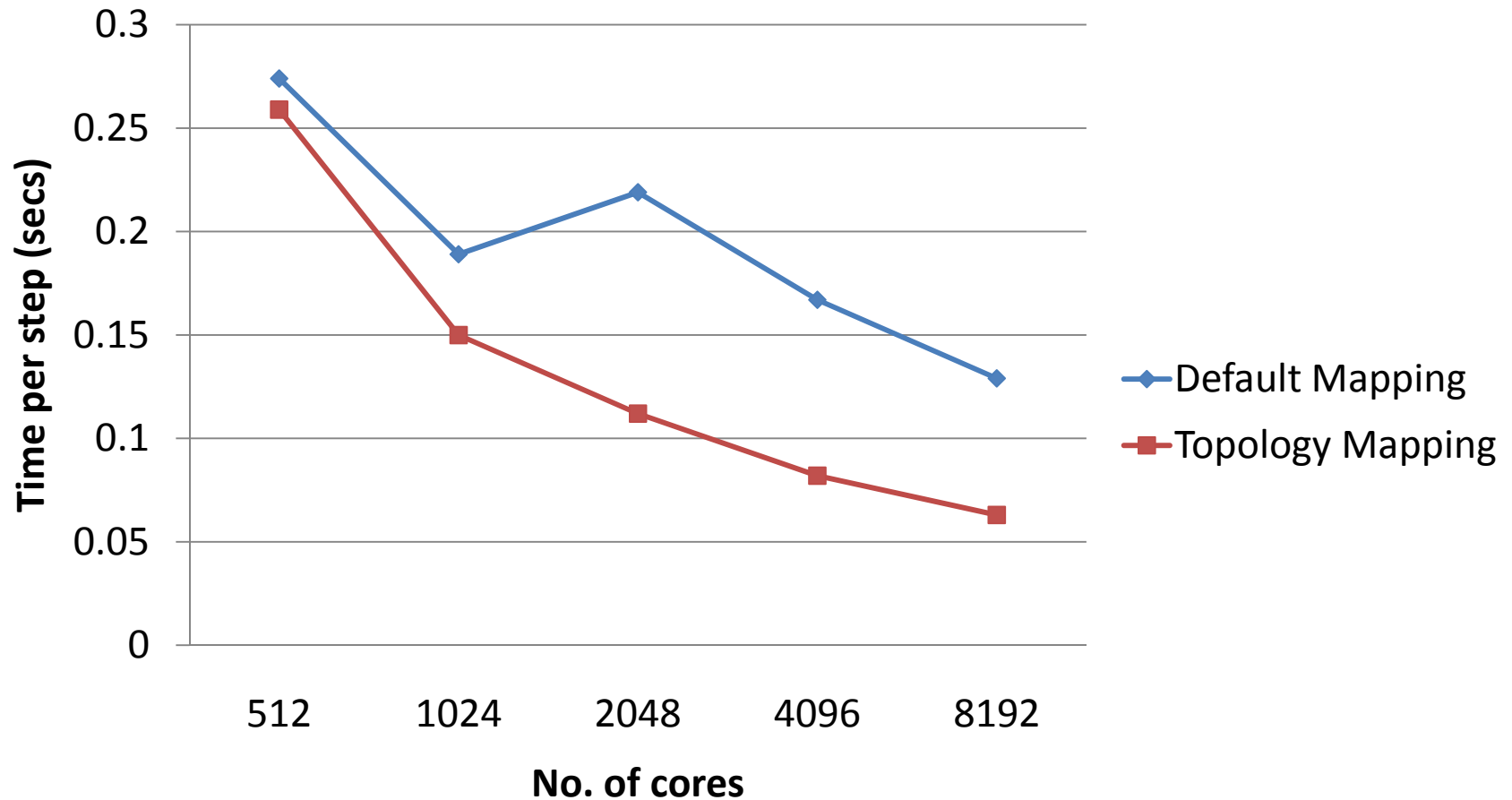
But in presence of contention ...



OpenAtom Performance on Blue Gene/L



OpenAtom Performance on Blue Gene/L



Previous Work

- Much work in the 80s
 - Physical optimization techniques: simulated annealing [1], graph contraction [2], genetic algorithms
 - Heuristic approaches: pairwise exchanges [3], recursive mincut bipartitioning [4]

[1] S. Wayne Bollinger and Scott F. Midkiff, ICPP, 1988.

[2] Francine Berman and Lawrence Snyder, Journal of Parallel and Distributed Computing, 1987.

[3] Shahid H. Bokhari, IEEE Trans. Computers, 1981.

[4] F. Ercal, J. Ramanujam, and P. Sadayappan, 3rd conference on Hypercube concurrent computers and applications, ACM Press, 1988.

Difference from previous work

Then	Now
Mainly for theoretical object graphs on hypercubes, shuffle exchange and other theoretical networks	Object graphs from real applications on 3D torus/mesh topologies
Most techniques were used offline – slow	Fast, runtime solutions
Demonstrated on graphs with 10-100 nodes	Scalable techniques for large machines
No cardinality variation	Multiple objects per processor
Not tested with real applications on actual machines – theoretical work	Targeted at production codes – tested with real applications

Difference from recent work

- [5] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. C. Sexton, and R. Walkup. Optimizing task layout on the Blue Gene/L supercomputer. IBM Journal of Research and Development, 49(2/3), 2005.
- Use of simulated annealing – slow and solution is developed offline
- [6] Hao Yu, I-Hsin Chung, and Jose Moreira. Topology mapping for Blue Gene/L supercomputer. In SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, page 116, New York, NY, USA, 2006. ACM
- Node mappings for simple scenarios (1D rings, 2D meshes, 3D)
 - Only useful in case of simple near-neighbor communication

Outline

- The Mapping Problem
- Motivation
- Previous Work
- **Evaluative Studies**
 - Quantifying message latencies
 - Characterizing applications
- Topology Aware Mapping
 - Topology Manager API
 - Application-specific Mapping
 - Automatic Mapping Framework

Demonstrate that topology aware mapping has become important again for 3D mesh topologies

Study the effects of contention on message latencies for different interconnects

Develop an automatic topology-aware mapping framework useful for application developers

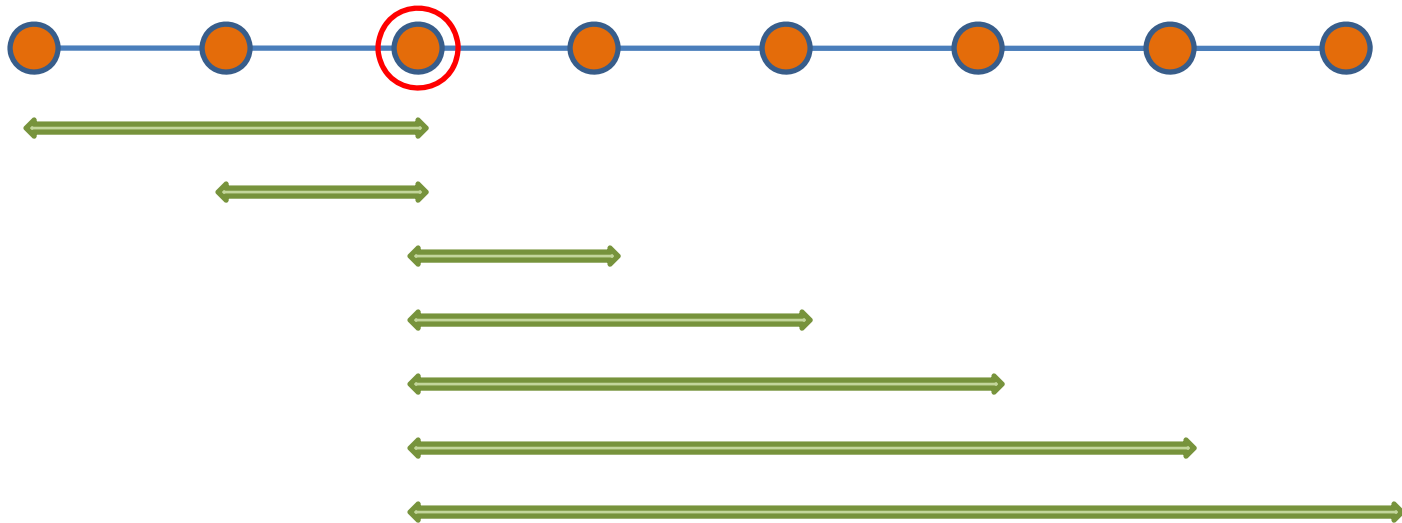
MPI Benchmarks†

- Quantification of message latencies and dependence on hops
 - No sharing of links (no contention)
 - Sharing of links (with contention)

† <http://charm.cs.uiuc.edu/~bhatele/phd/contention.htm>

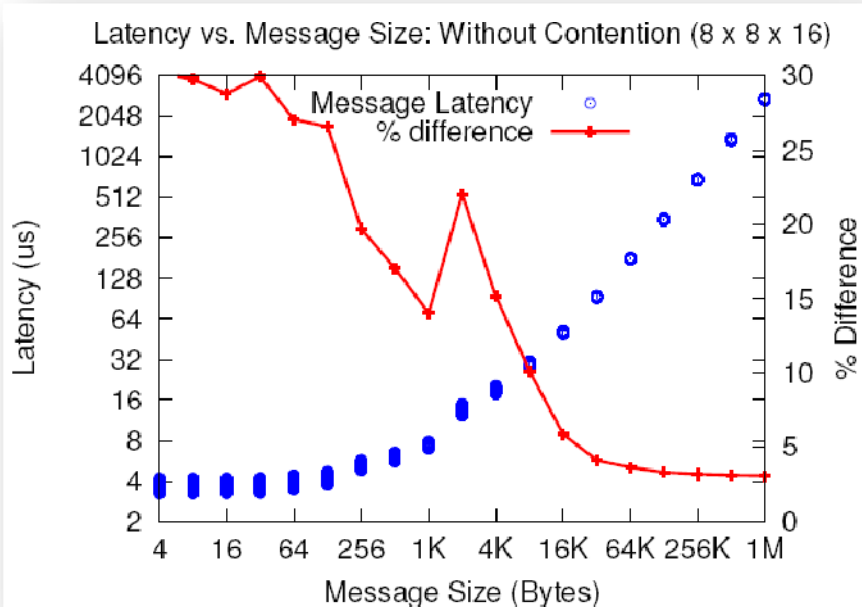
WOCON: No contention

- A master rank sends messages to all other ranks, one at a time (with replies)

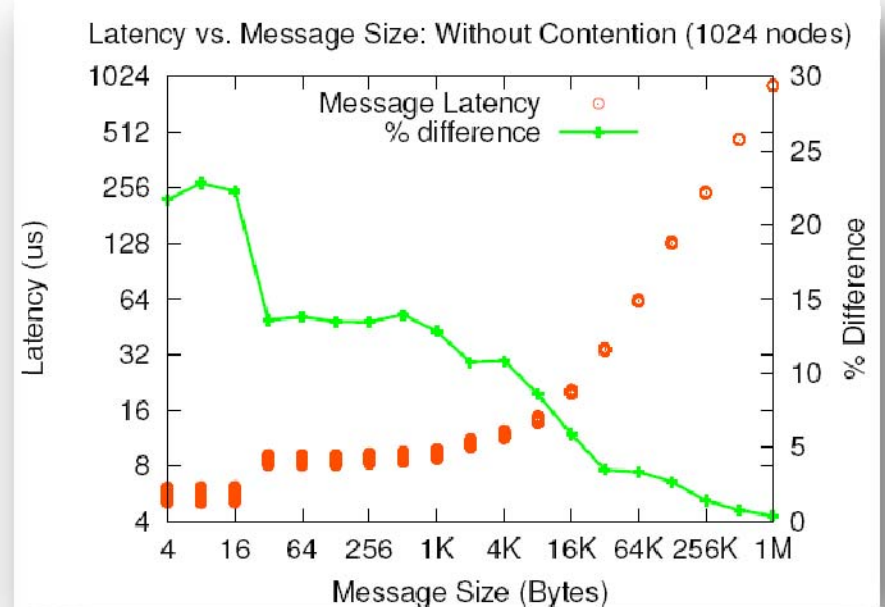


WOCON: Results

$$(L_f/B) \times D + L/B$$



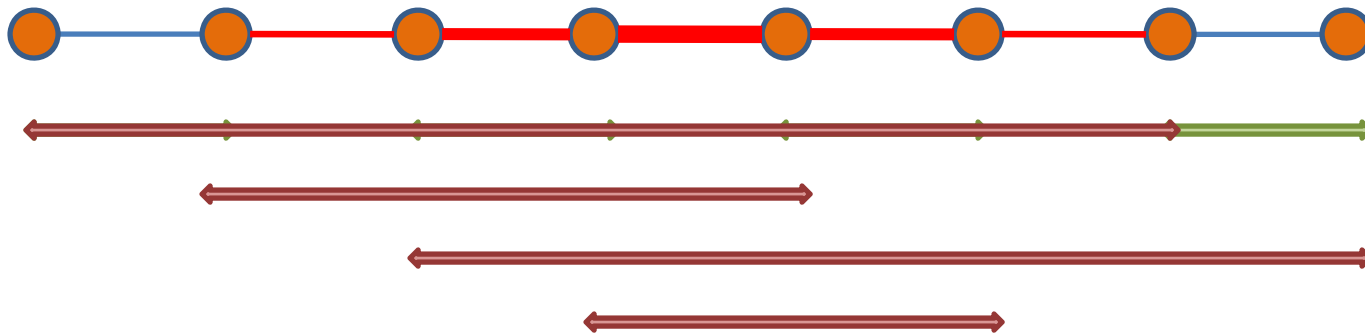
ANL Blue Gene/P



ORNL XT3

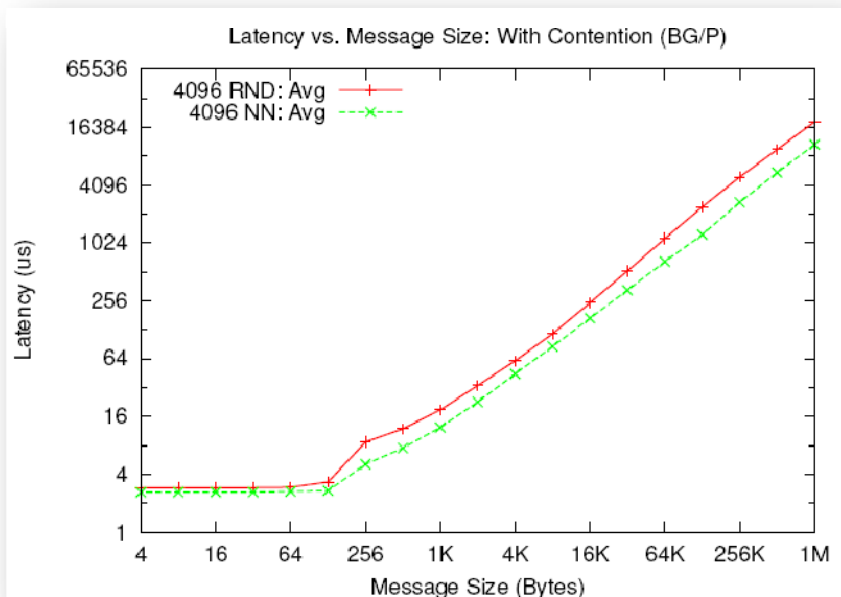
WICON: With Contention

- Divide all ranks into pairs and everyone sends to their respective partner simultaneously

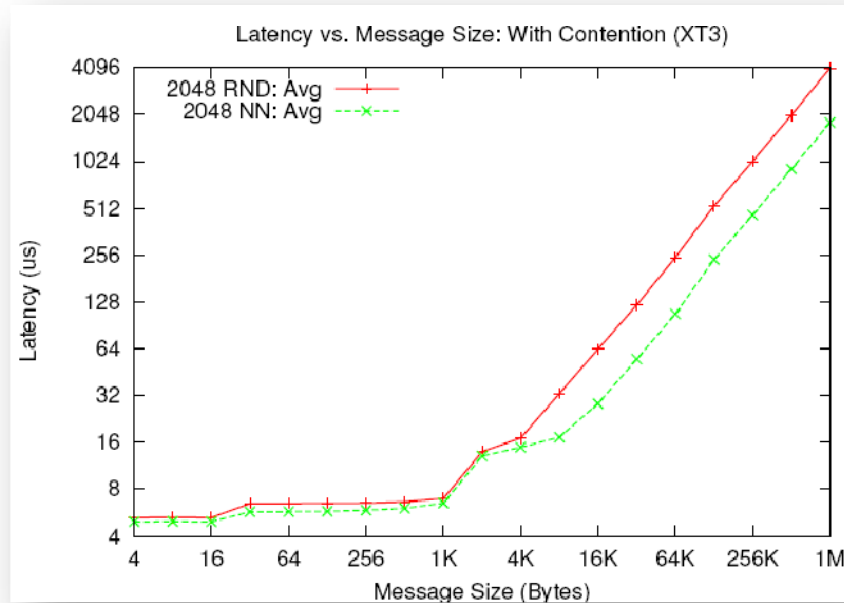


Read Neighbor: NN

WICON: Results



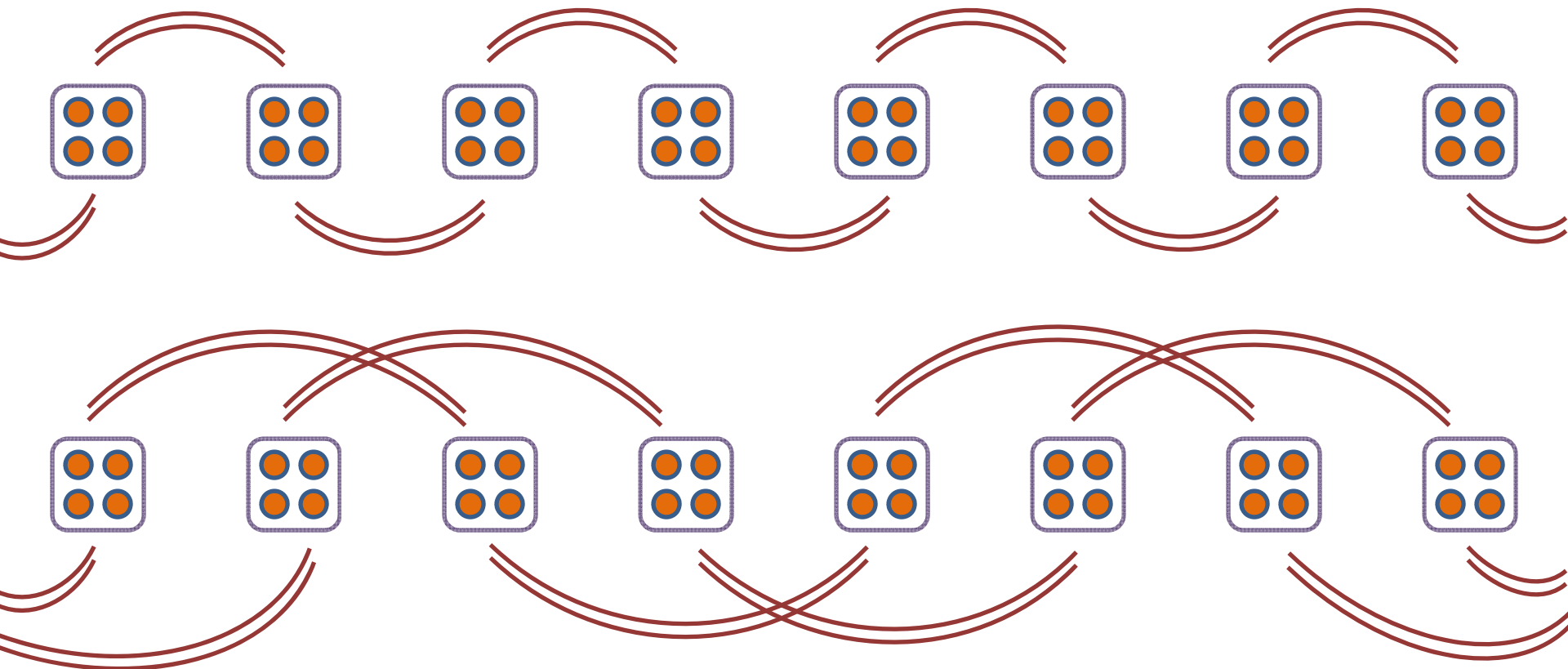
ANL Blue Gene/P

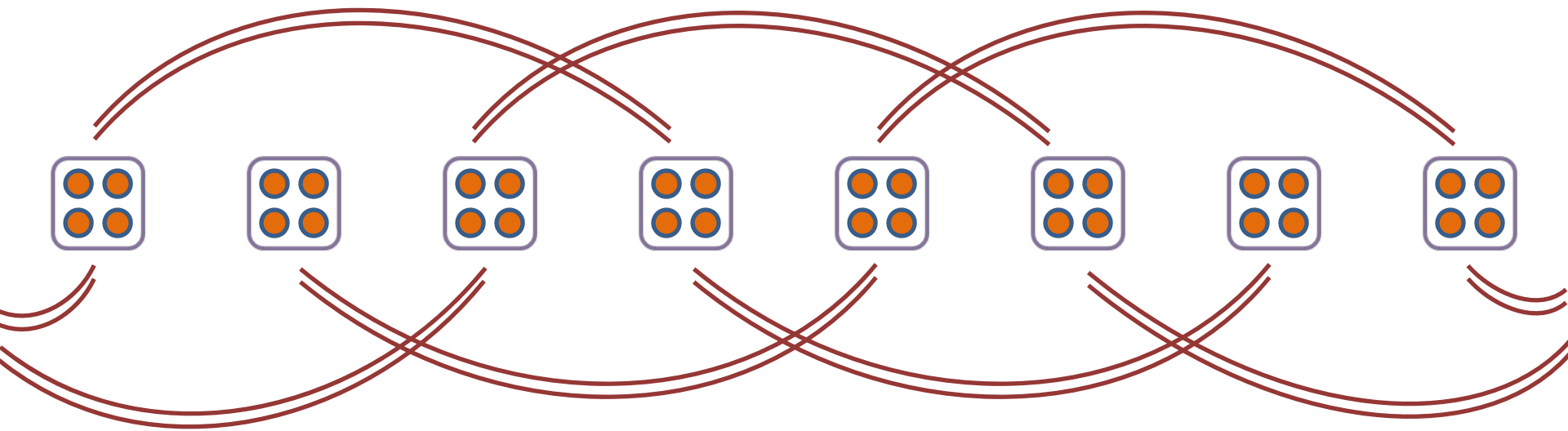


PSC XT3

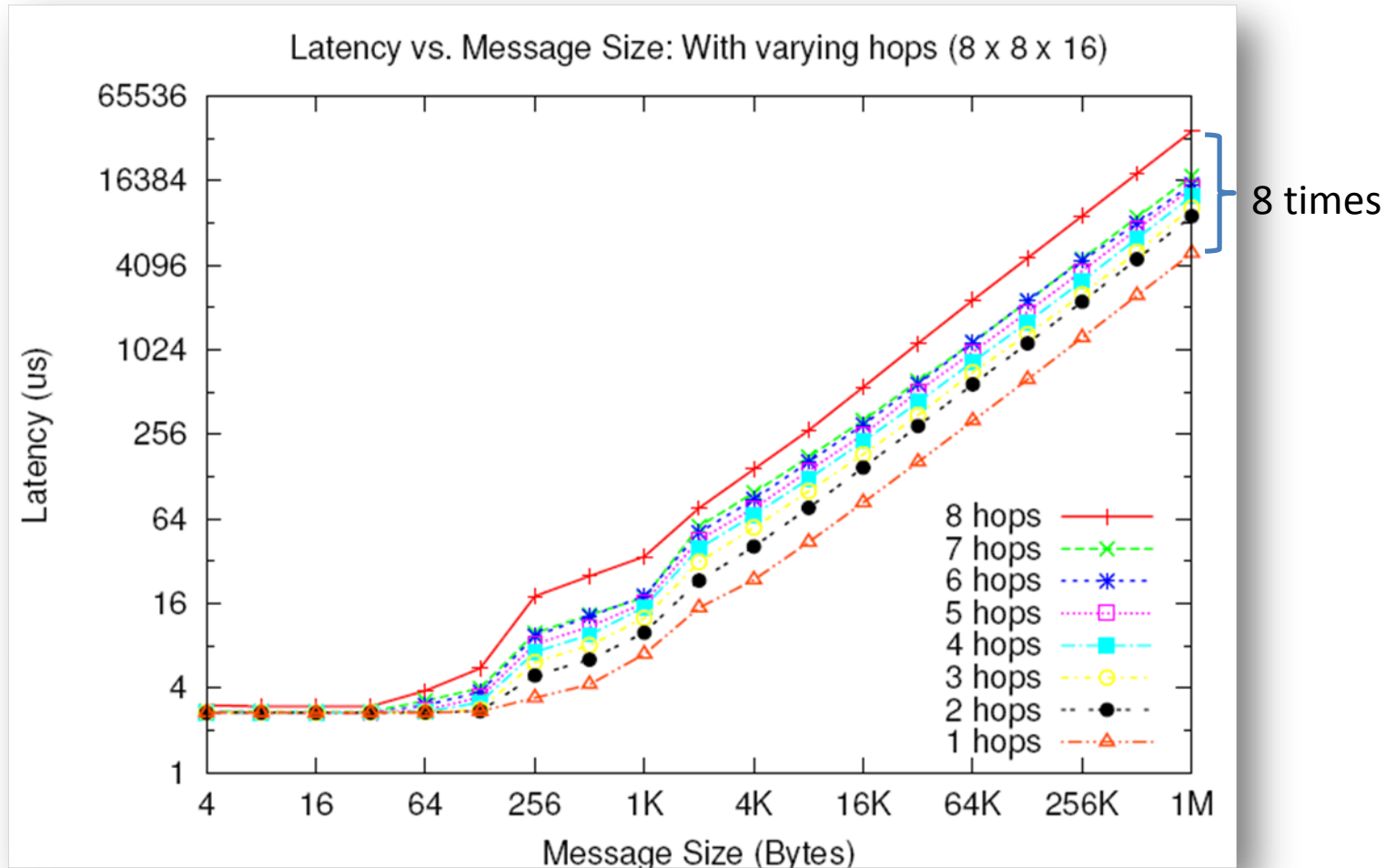
Message Latencies and Hops

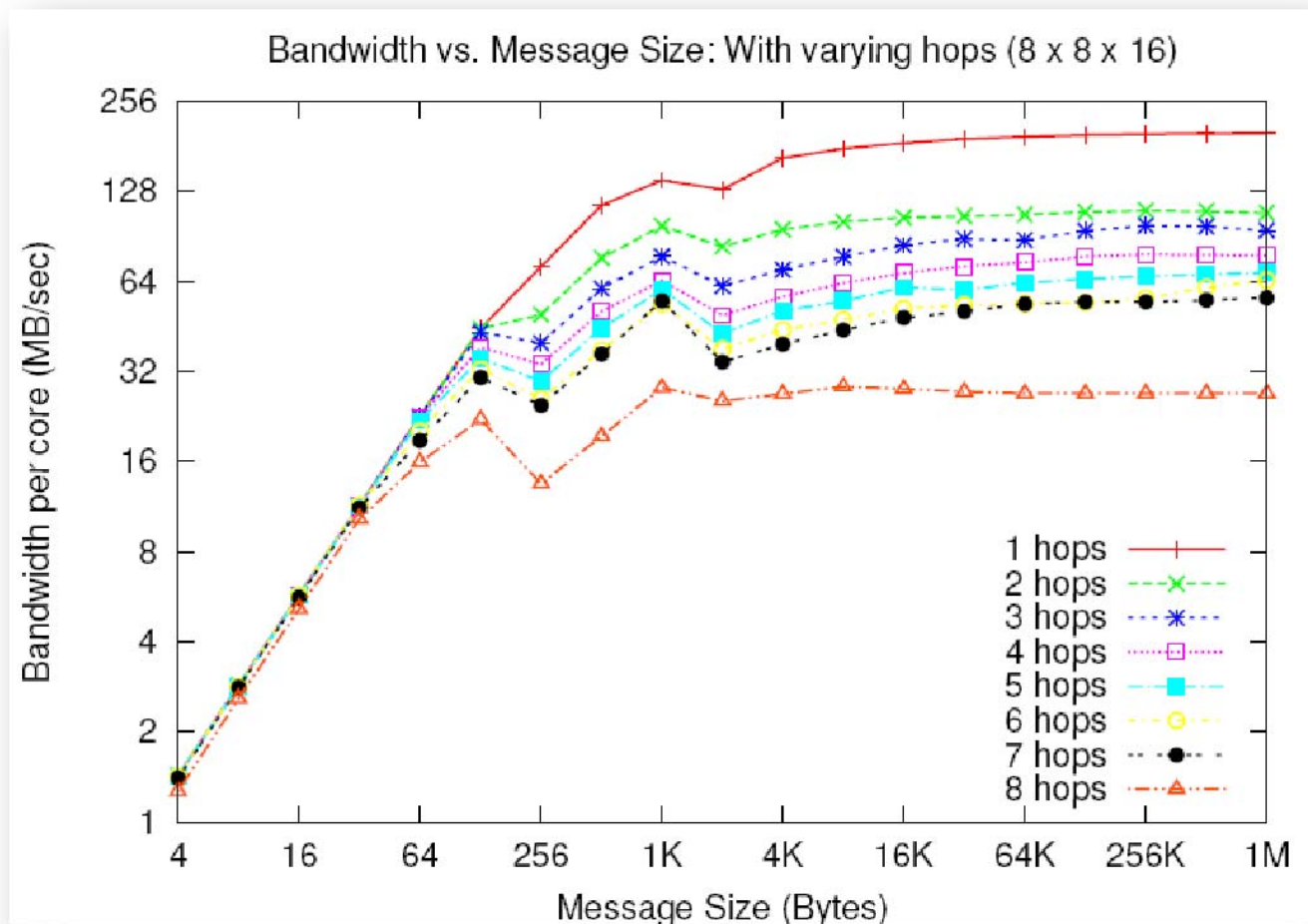
- Pair each rank with a partner which is 'n' hops away





Results





Contention Studies

- Further tests to understand the interplay of routing strategies with message latencies
- Develop models which predict message latencies in presence of contention
- Possibly: Study of other interconnects

Application Characterization

- Characterization of applications through BigSim
 - Latency tolerant
 - Latency bound
 - Computation intensive
 - Communication intensive
- Topology aware mapping is useful for latency bound applications
- Use of performance counters with applications
 - Performance counters can also be used to quantify link contention with different mapping schemes

Outline

- The Mapping Problem
- Motivation
- Previous Work
- Evaluative Studies
 - Quantifying message latencies
 - Characterizing applications
- **Topology Aware Mapping**
 - Topology Manager API
 - Application-specific Mapping
 - Automatic Mapping Framework

Demonstrate that topology aware mapping has become important again for 3D mesh topologies

Study the effects of contention on message latencies for different interconnects

Develop an automatic topology-aware mapping framework useful for application developers

Topology Aware Mapping

- Broadly three requirements
 - Processor topology graph
 - Object communication graph
 - Efficient and scalable mapping algorithms

Topology Manager API†

- The application needs information such as
 - Dimensions of the partition
 - Rank to physical co-ordinates and vice-versa
- TopoManager: a uniform API
 - On BG/L and BG/P: provides a wrapper for system calls
 - On XT3 and XT4, there are no such system calls
 - Provides a clean and uniform interface to the application

† <http://charm.cs.uiuc.edu/~bhatele/phd/topomgr.htm>

Communication Scenarios

- Arbitrary Object Graph: MD codes, Unstructured Mesh codes (possibly with AMR)

NAMD

- Regular Object Graph: 3D Stencil, Matrix Multiplication, Structured Mesh codes

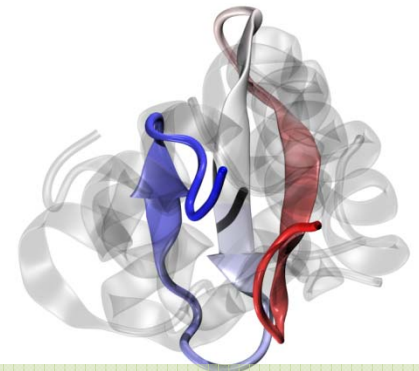
OpenAtom

Molecular Dynamics

- A system of [charged] atoms with bonds
- Use Newtonian Mechanics to find the positions and velocities of atoms
- Each time-step is typically in femto-seconds
- At each time step
 - calculate the forces on all atoms
 - calculate the velocities and move atoms around

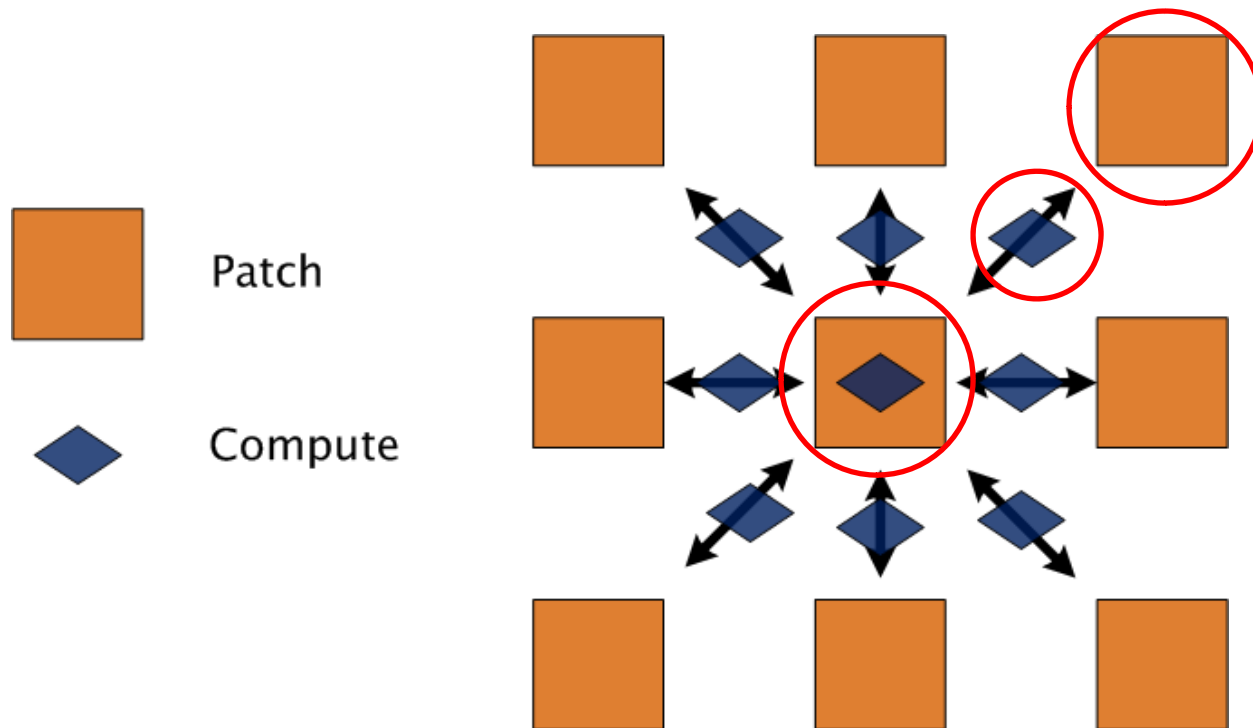
NAMD: NAnoscale Molecular Dynamics

- Naïve force calculation is $O(N^2)$
- Reduced to $O(N \log N)$ by calculating
 - Bonded forces
 - Non-bonded: using a cutoff radius
 - Short-range: calculated every time step
 - Long-range: calculated every fourth time-step (PME)

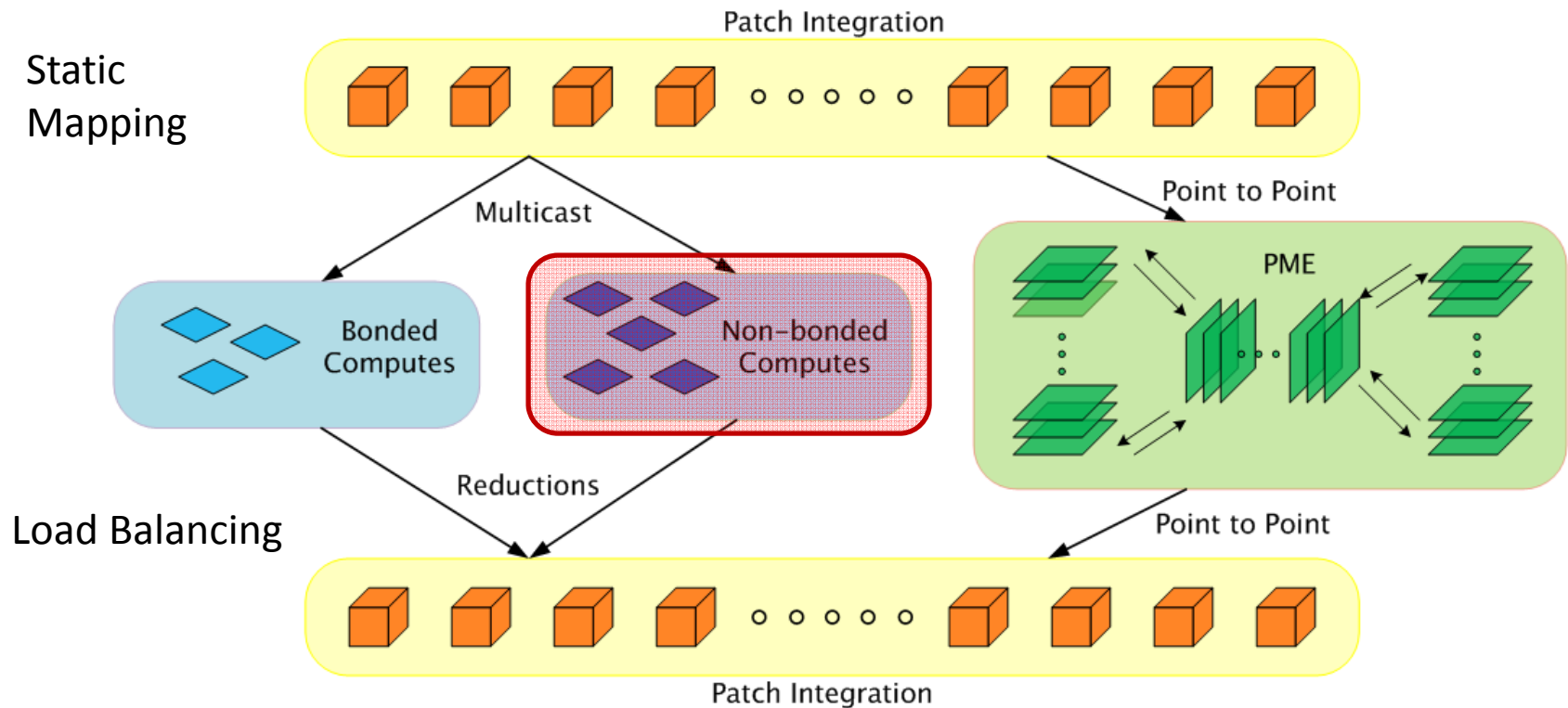


NAMD's Parallel Design

- Hybrid of spatial and force decomposition



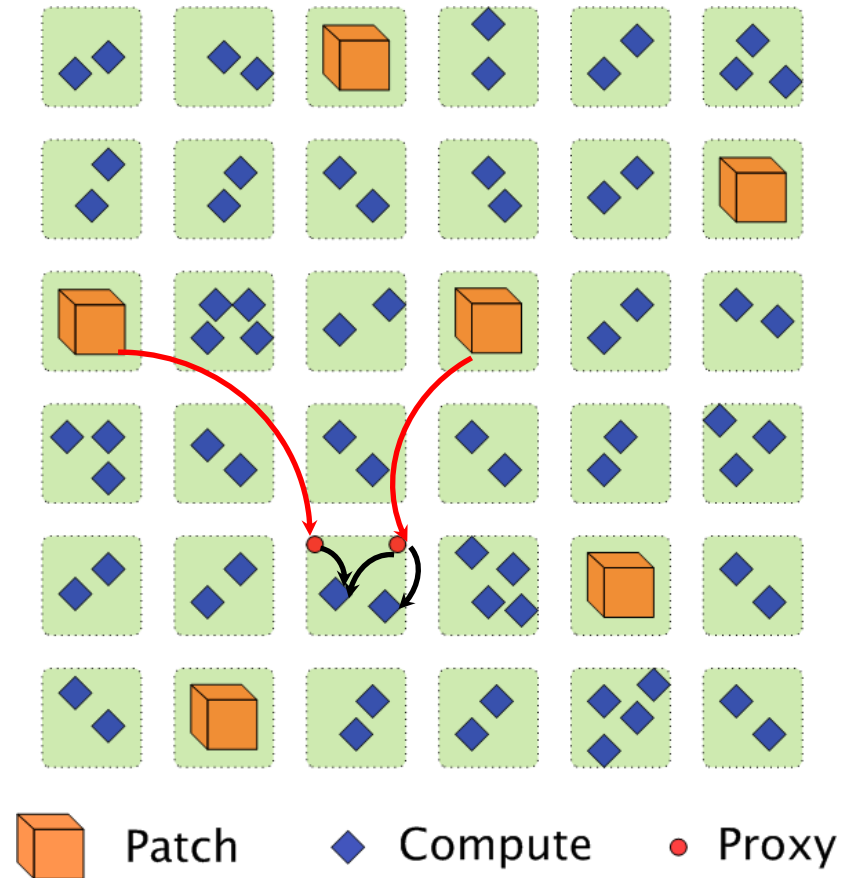
Parallelization using Charm++



[7] Bhatele, A., Kumar, S., Mei, C., Phillips, J. C., Zheng, G. & Kale, L. V., **Overcoming Scaling Challenges in Biomolecular Simulations across Multiple Platforms**. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, Miami, FL, USA, April 2008.

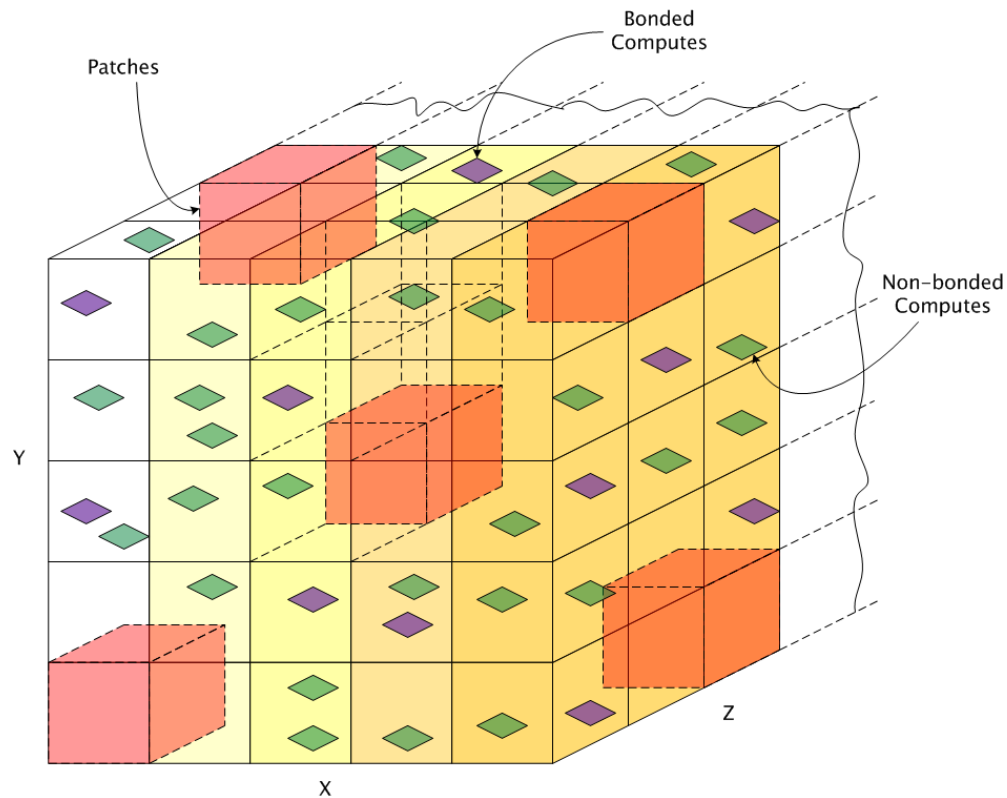
Communication in NAMD

- Each patch multicasts its information to many computes
- Each compute is a target of two multicasts only
- Use 'Proxies' to send data to different computes on the same processor



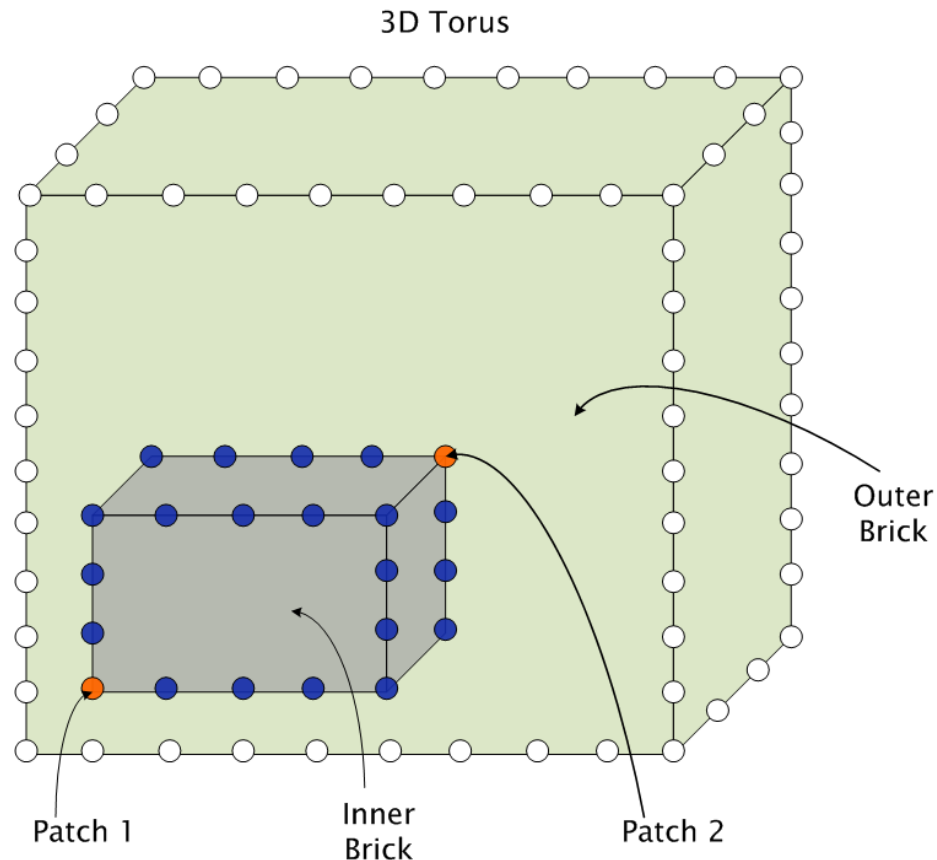
Topology Aware Techniques

- Static Placement of Patches



Topology Aware Techniques (contd.)

- Placement of computes



Load Balancing in Charm++

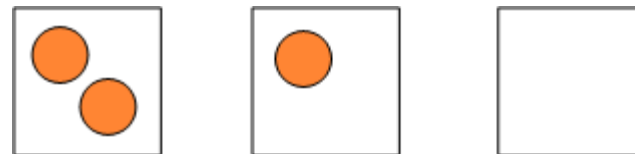
- Principle of Persistence
 - Object communication patterns and computational loads tend to persist over time
- Measurement-based Load Balancing
 - Instrument computation time and communication volume at runtime
 - Use the database to make new load balancing decisions

NAMD's Load Balancing Strategy

- NAMD uses a dynamic centralized greedy strategy
- There are two schemes in play:
 - A comprehensive strategy (called once)
 - A refinement scheme (called several times during a run)
- Algorithm:
 - Pick a compute and find a “suitable” processor to place it on

Choice of a suitable processor

- Among underloaded processors, try to:
 - Find a processor with the two patches or their proxies
 - Find a processor with one patch or a proxy
 - Pick any underloaded processor



● = Proxy or Patch

Highest
Priority



Lowest
Priority

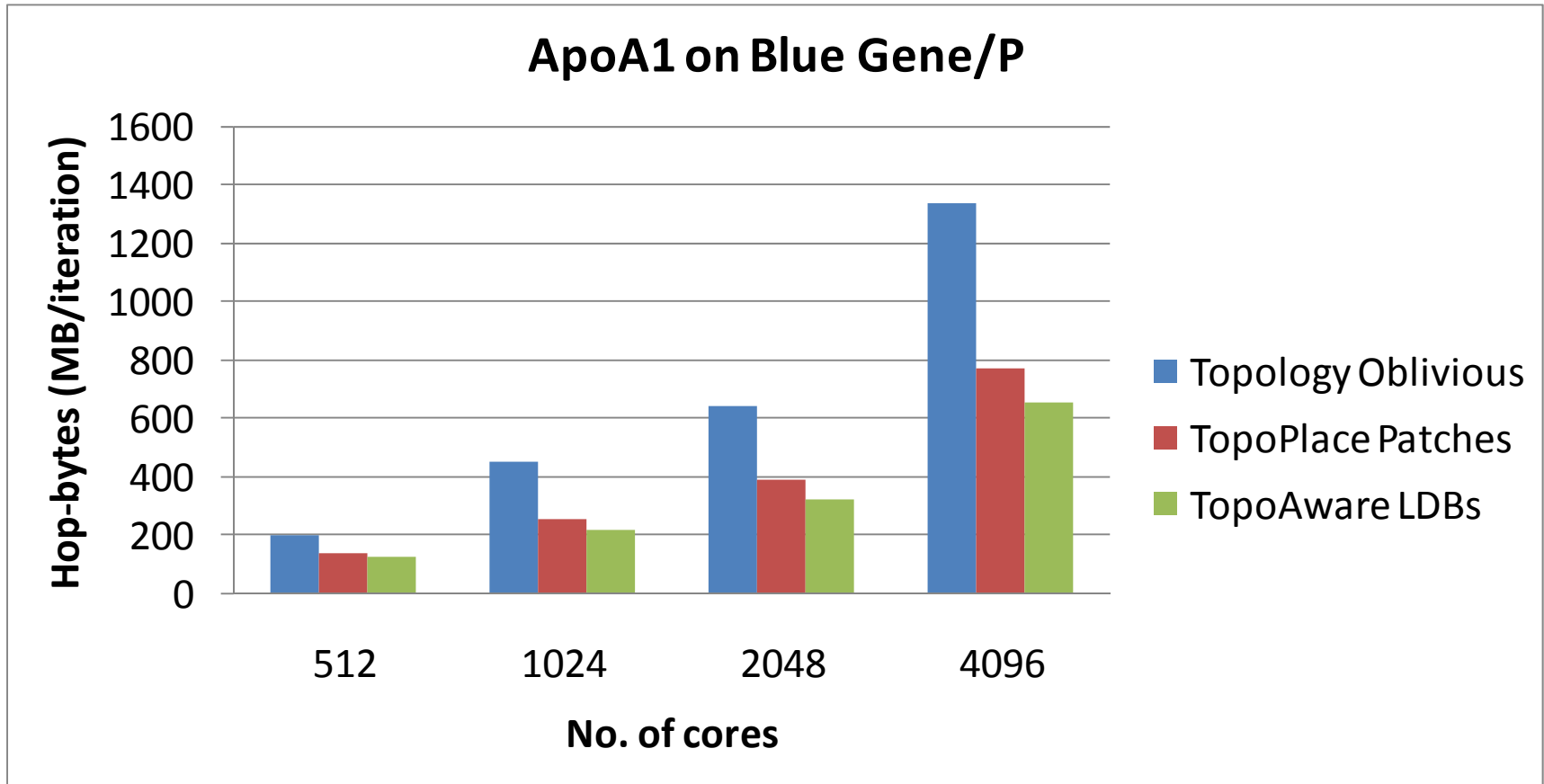
Load Balancing Metrics

- Load Balance: Bring Max-to-Avg Ratio close to 1
- Communication Volume: Minimize the number of proxies
- Communication Traffic: Minimize hop bytes

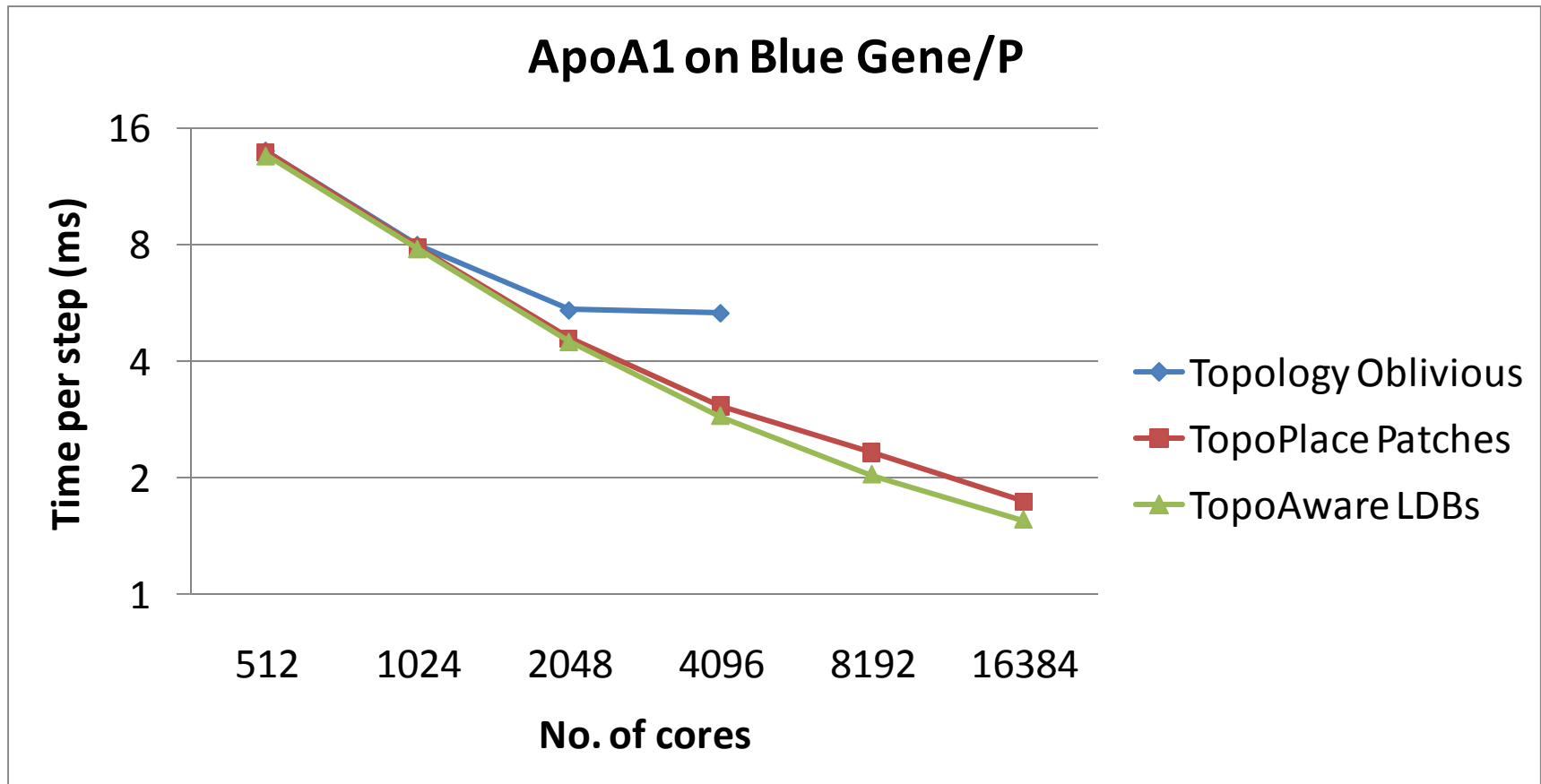
Hop-bytes = Message size X Distance traveled by message

[8] Agarwal, T., Sharma, A., Kale, L.V., **Topology-aware task mapping for reducing communication contention on large parallel machines**, In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, April 2006.

Results: Hop-bytes



Results: Performance



[9] Abhinav Bhatele and Laxmikant V. Kale. **Dynamic Topology Aware Load Balancing Algorithms for MD Applications.** *submitted to Philosophical Transactions of the Royal Society A*, 2008.

Scalable Load Balancing

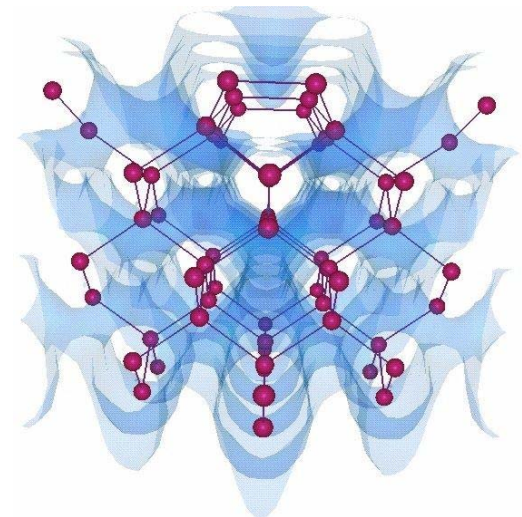
- Current centralized strategy presents scalability issues around 16K cores
- Distributed strategies
 - Have restricted view of the object graph
 - Results not as accurate as centralized strategies
- Hybrid topology-aware strategies
 - Divide the torus into sub-partitions
 - Centralized strategy within the sub-partitions
 - Refinement strategies between different sub-tori

Other Applications

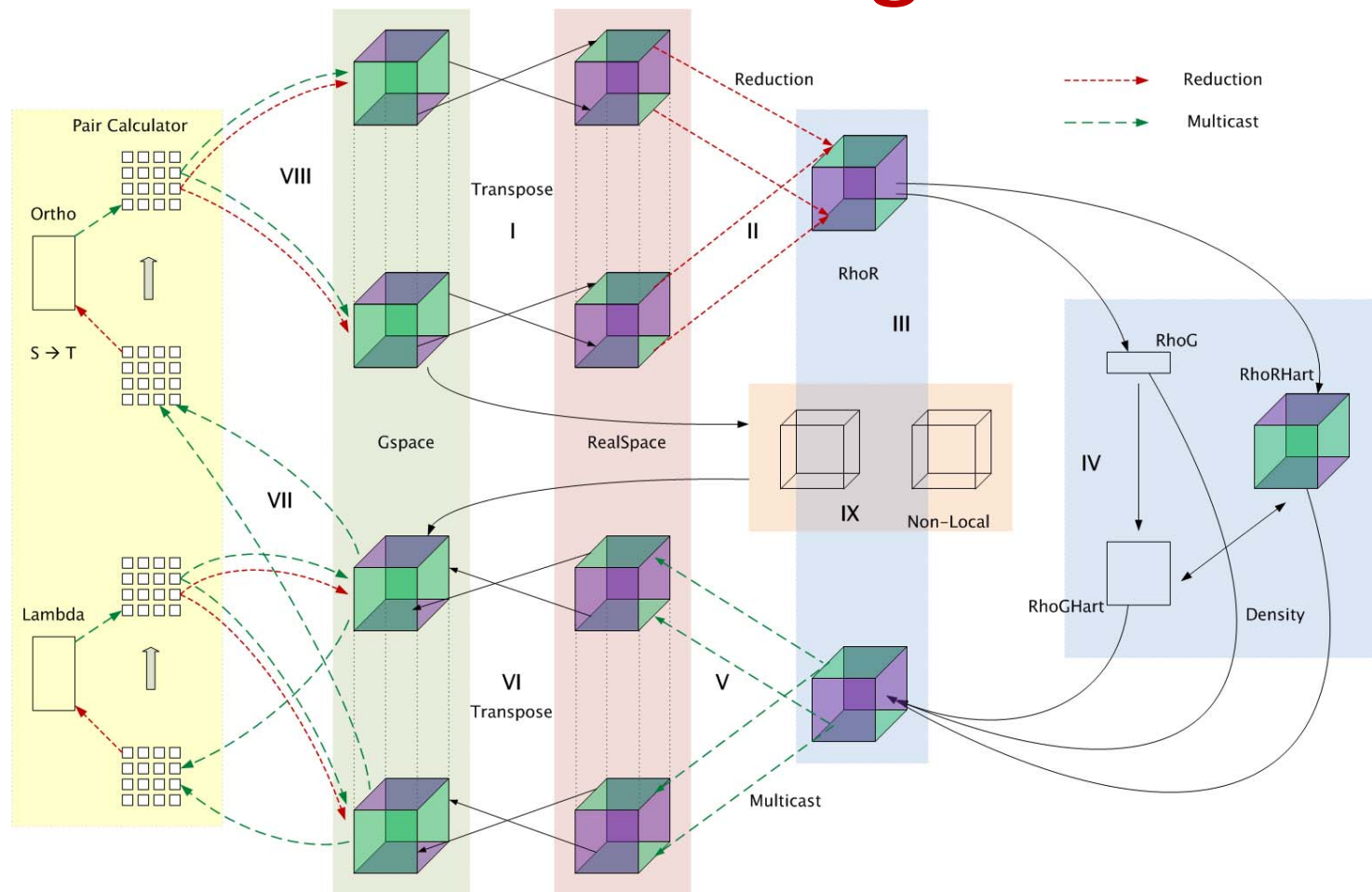
- Unstructured mesh computation
 - Presents similar challenges (arbitrary object graph)
 - Utilize coordinate information for initial mapping
 - Runtime instrumentation for load balancing
- Adaptive mesh refinement (AMR) complicates the issues
 - Topology aware load balancing can be used

OpenAtom

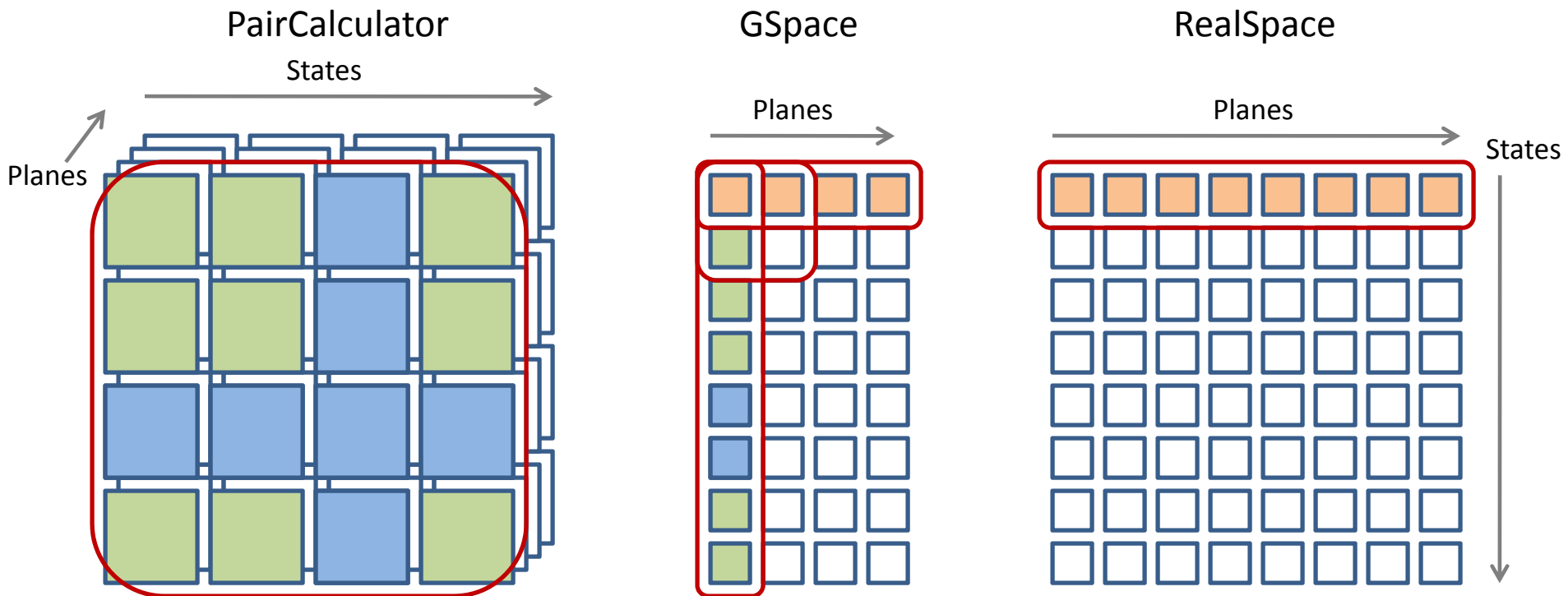
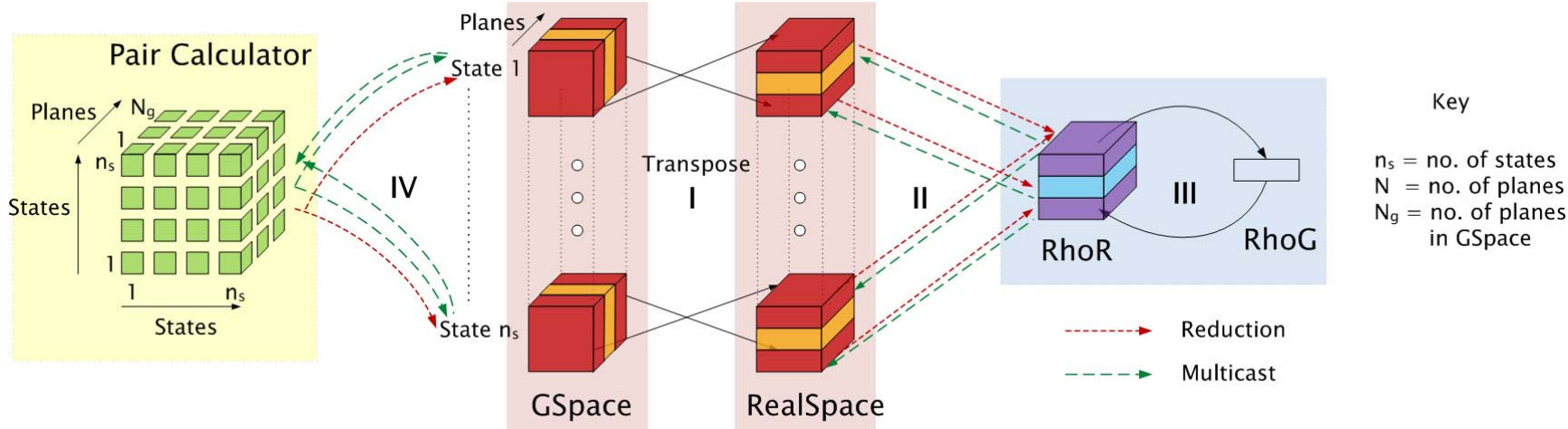
- Ab-Initio Molecular Dynamics code
- Communication is static and structured
- Challenge: Multiple groups of objects with conflicting communication patterns



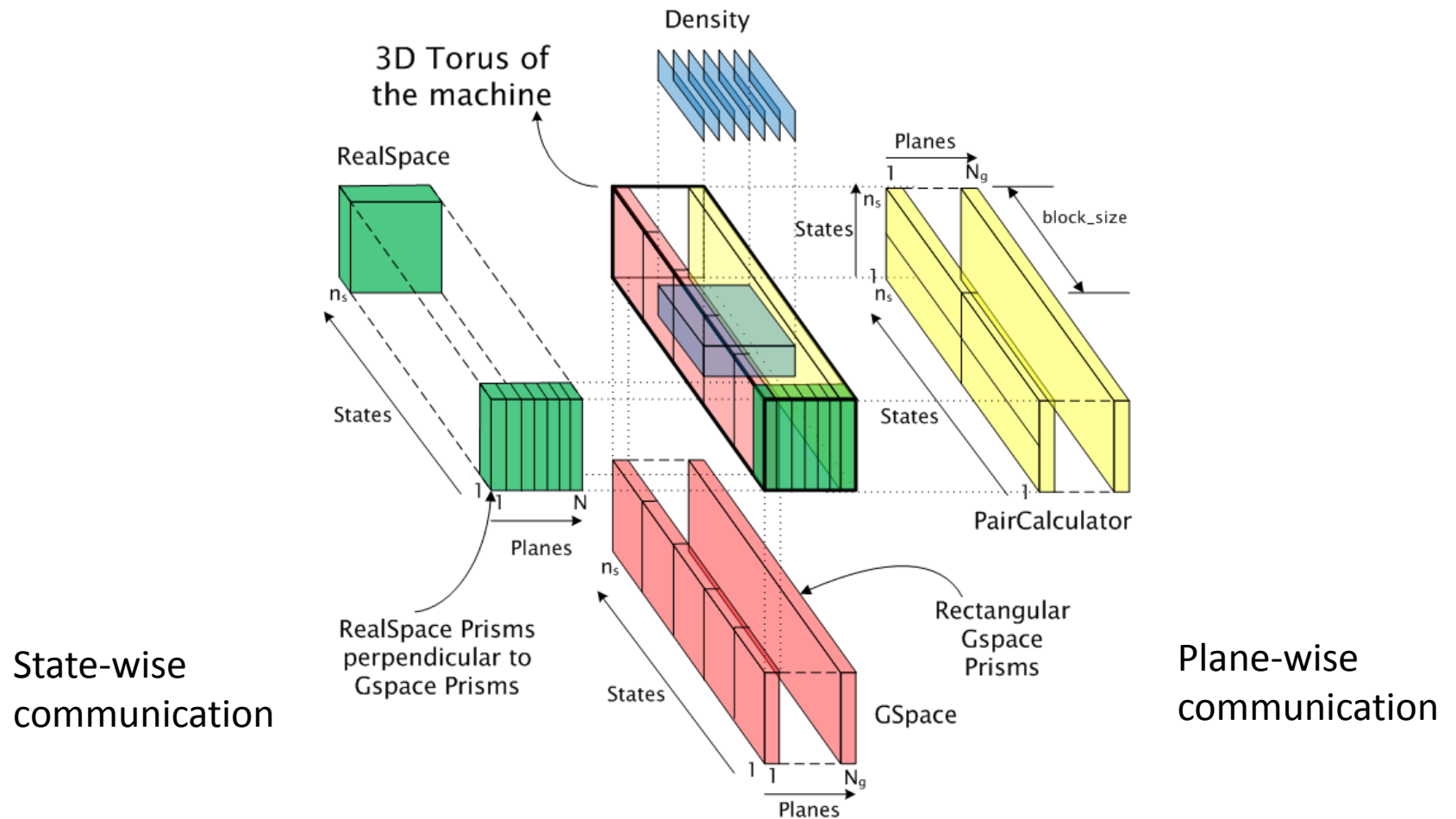
Parallelization using Charm++



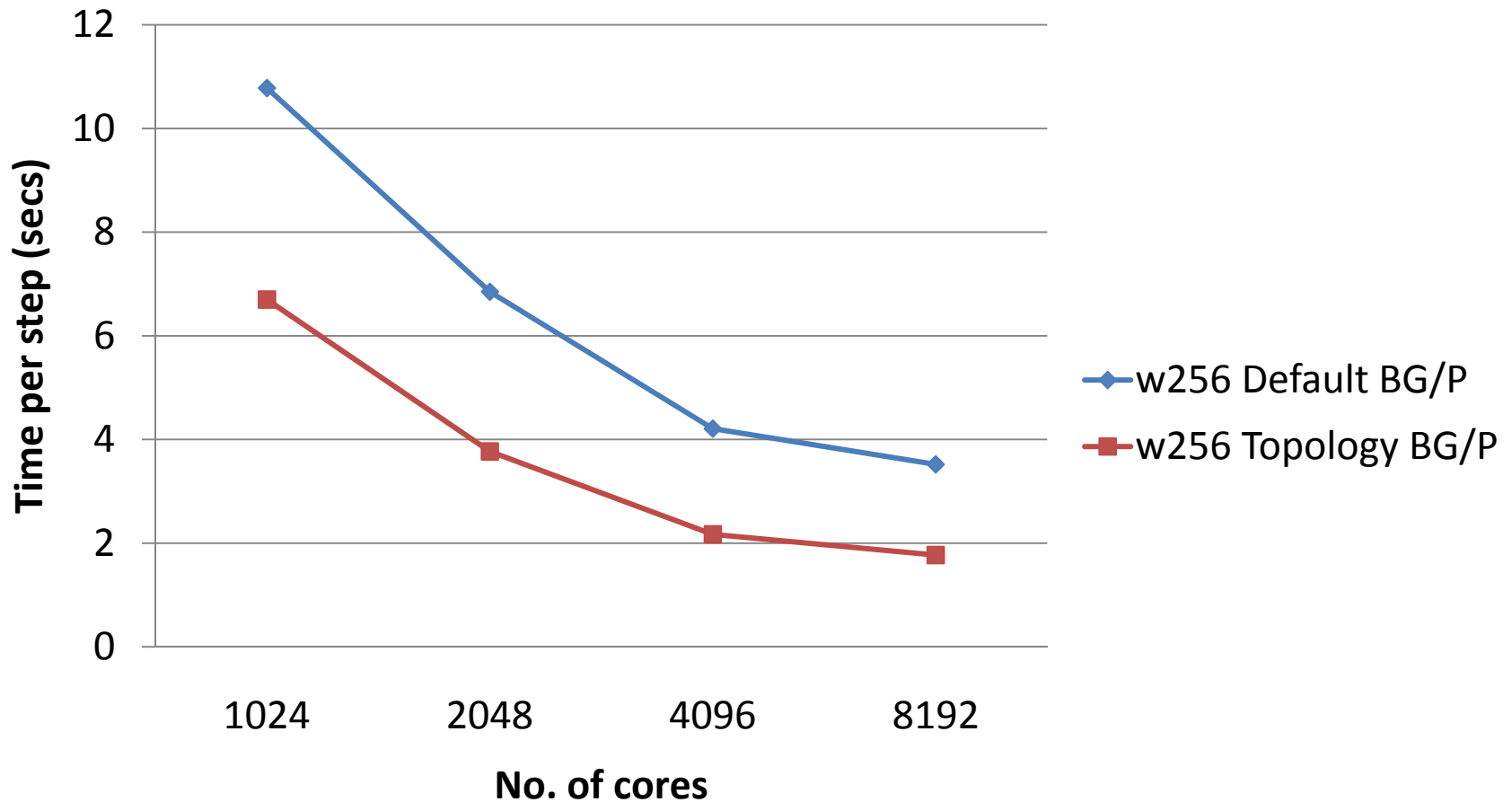
[10] Eric Bohm, Glenn J. Martyna, Abhinav Bhatele, Sameer Kumar, Laxmikant V. Kale, John A. Gunnels, and Mark E. Tuckerman. **Fine Grained Parallelization of the Car-Parrinello ab initio MD Method on Blue Gene/L.** *IBM J. of R. and D.: Applications of Massively Parallel Systems*, 52(1/2):159-174, 2008.



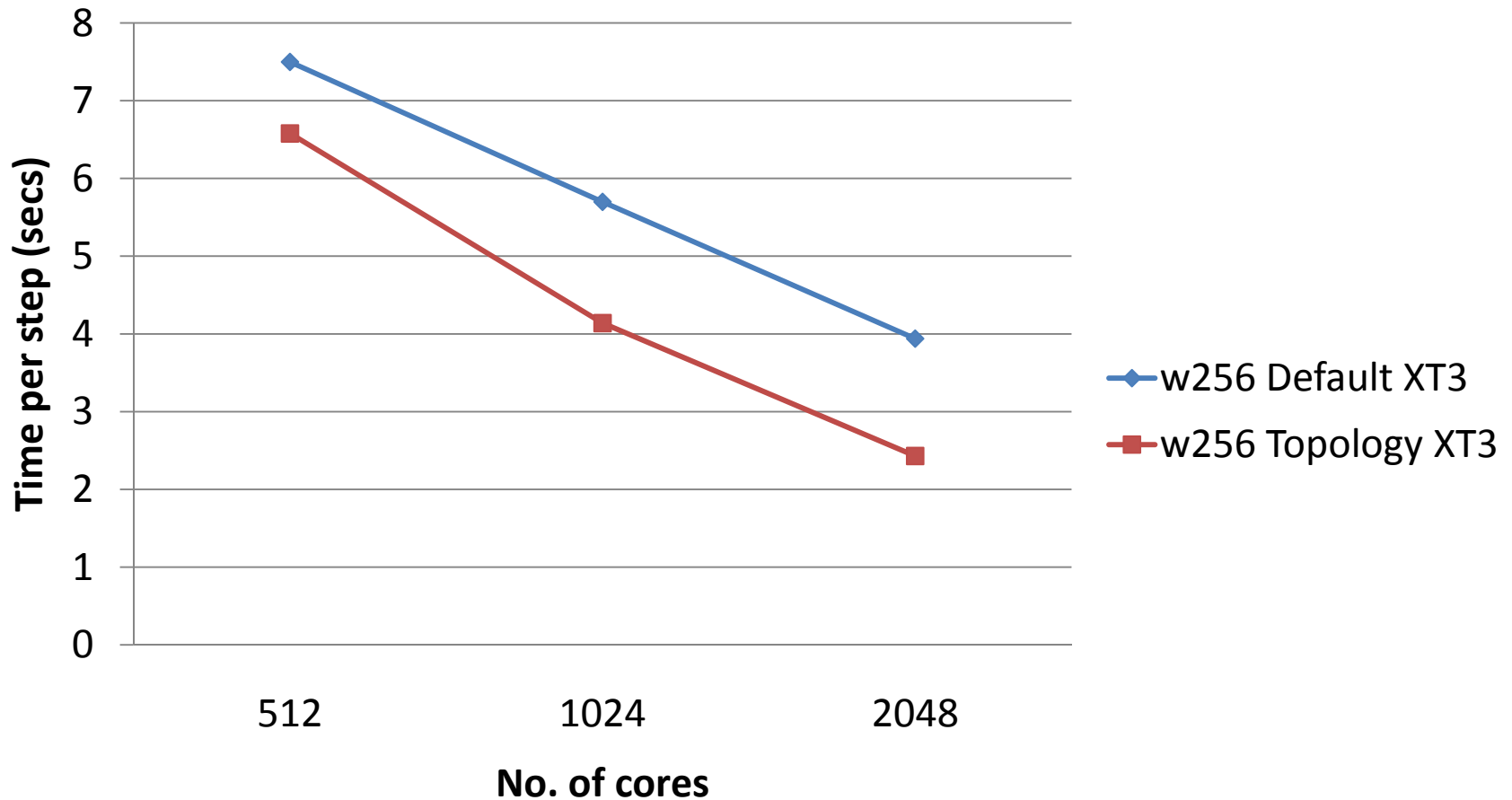
Topology Mapping of Chare Arrays



Results on Blue Gene/P (ANL)

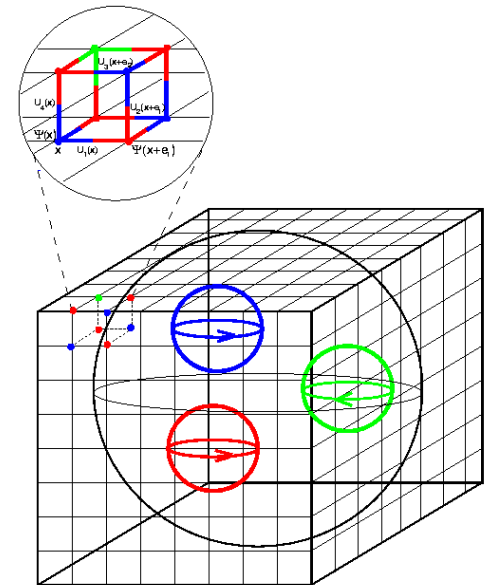


Results on XT3 (BigBen)



Regular Graph

- Matrix Multiplication
- Lattice QCD (Quantum Chromodynamics) Collaboration
 - Near-neighbor communication
- MILC (MIMD Lattice Computation)
 - Static Structured communication
 - Map 4D grid to 3D mesh



Further challenges

- Multiple object graphs communicating with each other
 - Create simpler standalone cases similar to OpenAtom
 - Automate mapping for such cases

Automatic Mapping Framework

- Obtain the communication graph (previous run, at runtime, compile time)
- Apply heuristic techniques for different scenarios to arrive at a mapping solution automatically
 - Test the strategies on different MPI and Charm++ applications

Summary

1. Topology is important again
2. Even on fast interconnects such as Cray

1. In presence of contention, bandwidth occupancy effects message latencies significantly
2. Increases with the number of hops each message travels

1. Further study and modeling of latencies in presence of contention
2. Application characterization to identify latency-bound applications

1. Topology Manager API: A uniform API for IBM and Cray machines
2. Irregular (arbitrary graph): NAMD
3. Regular graph: OpenAtom

1. Irregular graph: Unstructured mesh codes (possibly with AMR)
2. Regular graph: Matrix Multiplication, Lattice QCD, MILC
3. Automatic Topology Aware Mapping Framework

Thanks!