Analyzing the Effect of Different Programming Models Upon Performance and Memory Usage on Cray XT5 Platorms

Hongzhang Shan,
Berkeley Lab/NERSC

Haoqiang Jin
NASA Arms Research Center

Karl Fuerlinger
U.C. Berkeley

Alice Koniges, Nicholas J. Wright
NERSC

Despite continued "packing" of transistors, performance is flatlining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended
- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!

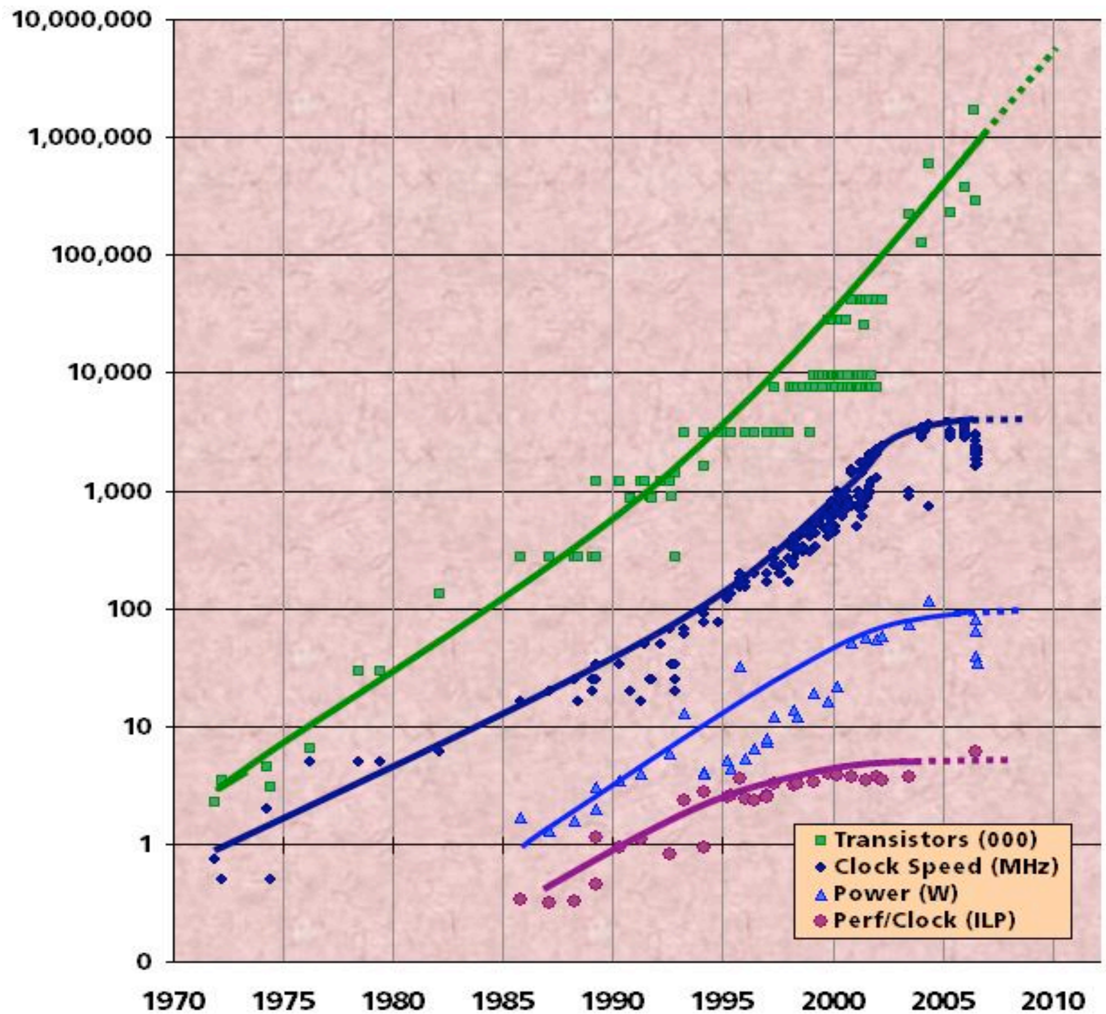
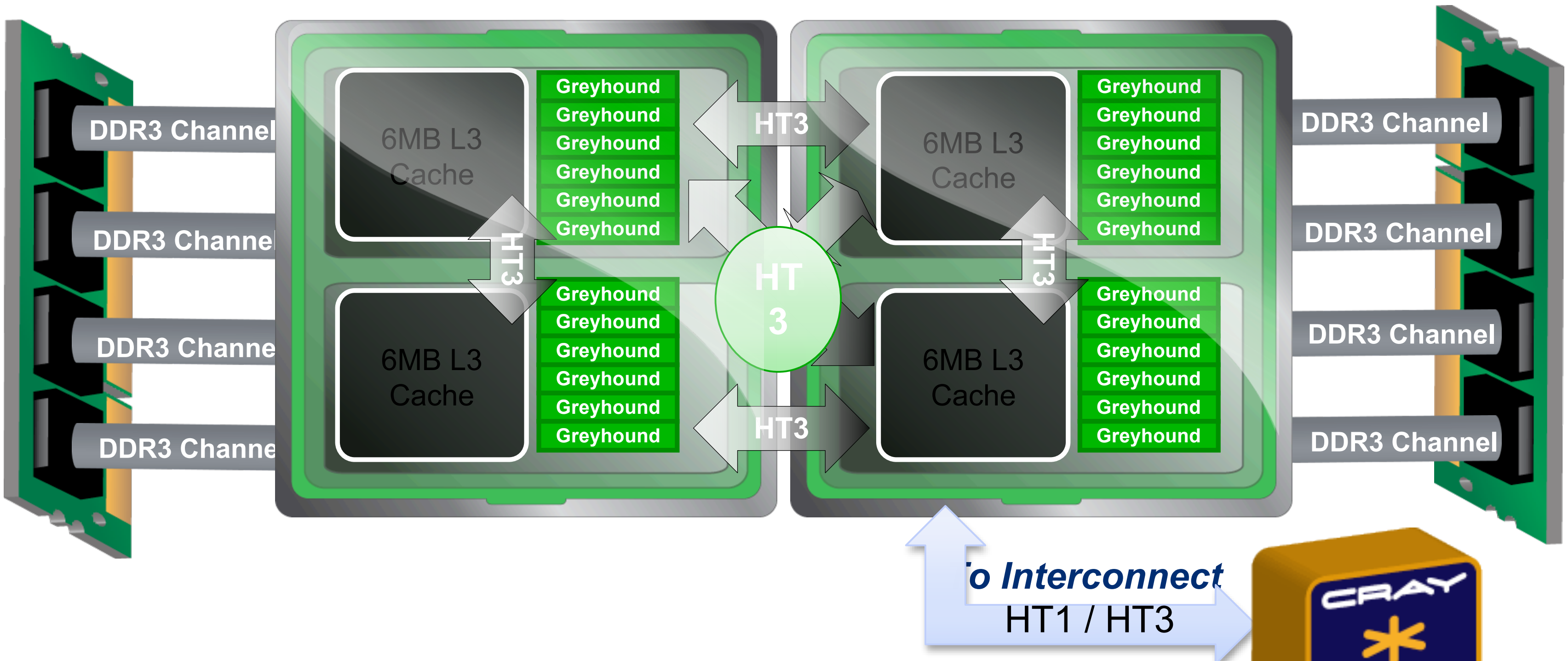


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Computer Centers and Vendors are Responding with Multi-core Designs



- Baker Node Details : 24-core “Magny Cours”
- 2 Multi-Chip Modules, 4 Opteron Dies
- 8 Channels of DDR3 Bandwidth to 8 DIMMs
- 24 (or 16) Computational Cores, 24 MB of L3 cache

What's Wrong with MPI with Multi-core

- **We can run 1 MPI process per core (flat model for parallelism)**
  - This works now and will work for a while
  - But this is wasteful of intra-chip latency and bandwidth (100x lower latency and 100x higher bandwidth on chip than off-chip)
  - Model has diverged from reality (the machine is ***NOT*** flat)
- **How long will it continue working?**
  - 4 - 8 cores? Probably. 128 - 1024 cores? Probably not.
  - Depends on performance expectations
- **What is the problem?**
  - Latency: some copying required by semantics
  - Memory utilization: partitioning data for separate address space requires some replication
    - How big is your per core subgrid? At 10x10x10, over 1/2 of the points are surface points, probably replicated
  - Memory bandwidth: extra state means extra bandwidth
  - Weak scaling: success model for the "cluster era;" will not be for the many core era -- not enough memory per core
  - Heterogeneity: MPI per CUDA thread-block?

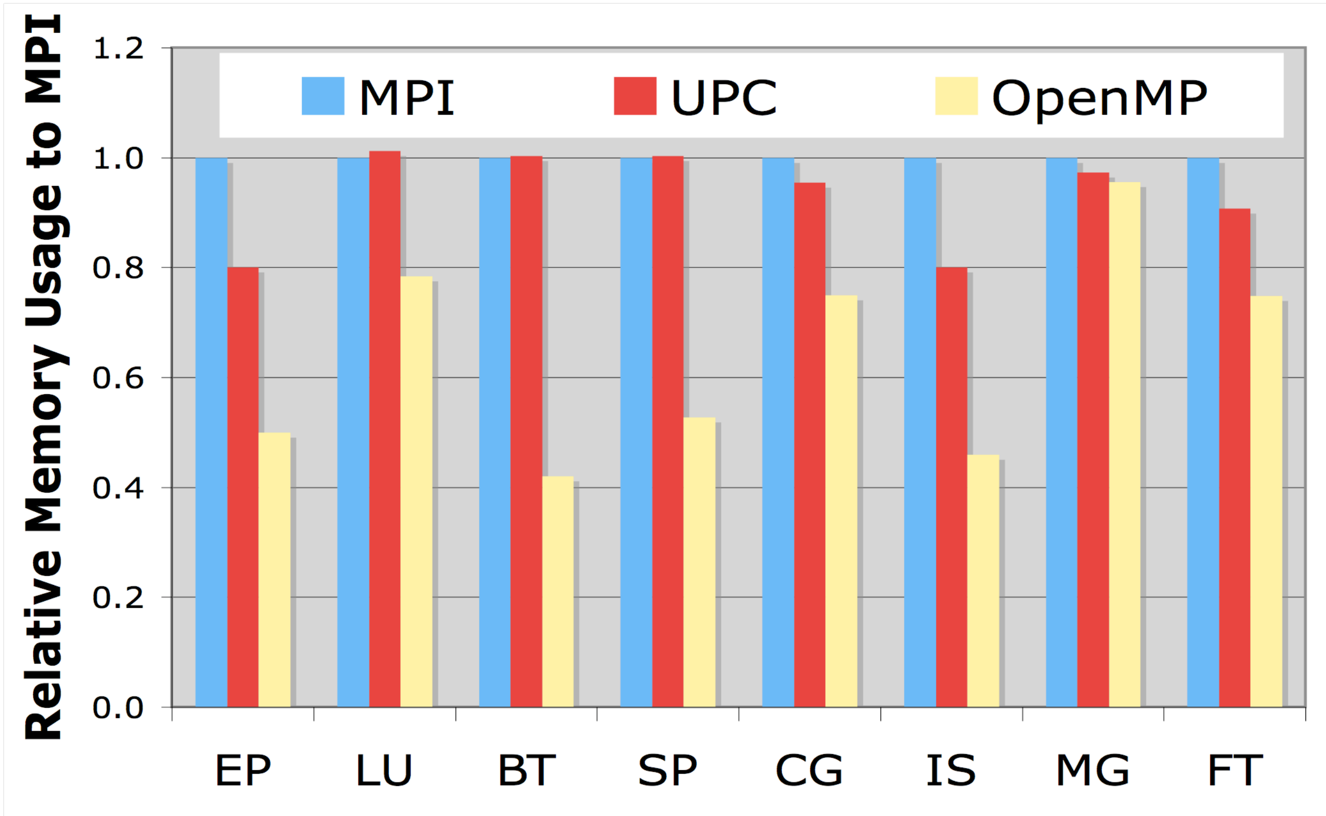
Changing Programming Models to Accommodate the Multi-core Revolution

- **We Need to research on other programming models, understand their advantages and disadvantages**
  - OpenMP
  - UPC
  - Hybrid MPI+OpenMP
  - Etc.
- **Our Work is focus on Cray XT5**

Outline

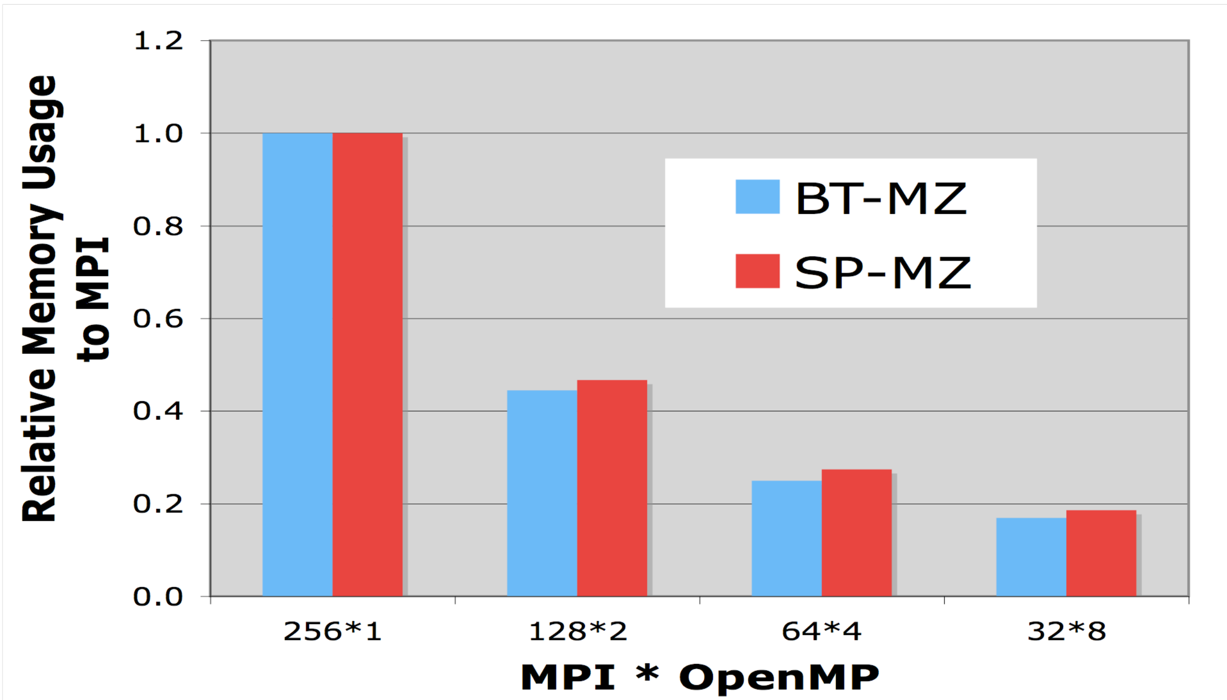
- Quantify Memory Usage for Different Programming Models
- Using Detailed Time Breakdown to Investigate Performance Effects of Different Programming Models
- Compare the Performance of Hopper and Jaguar to evaluate the hex-core and quad-core difference
- Conclusion and Future Work

Memory Usage : OpenMP, UPC, MPI



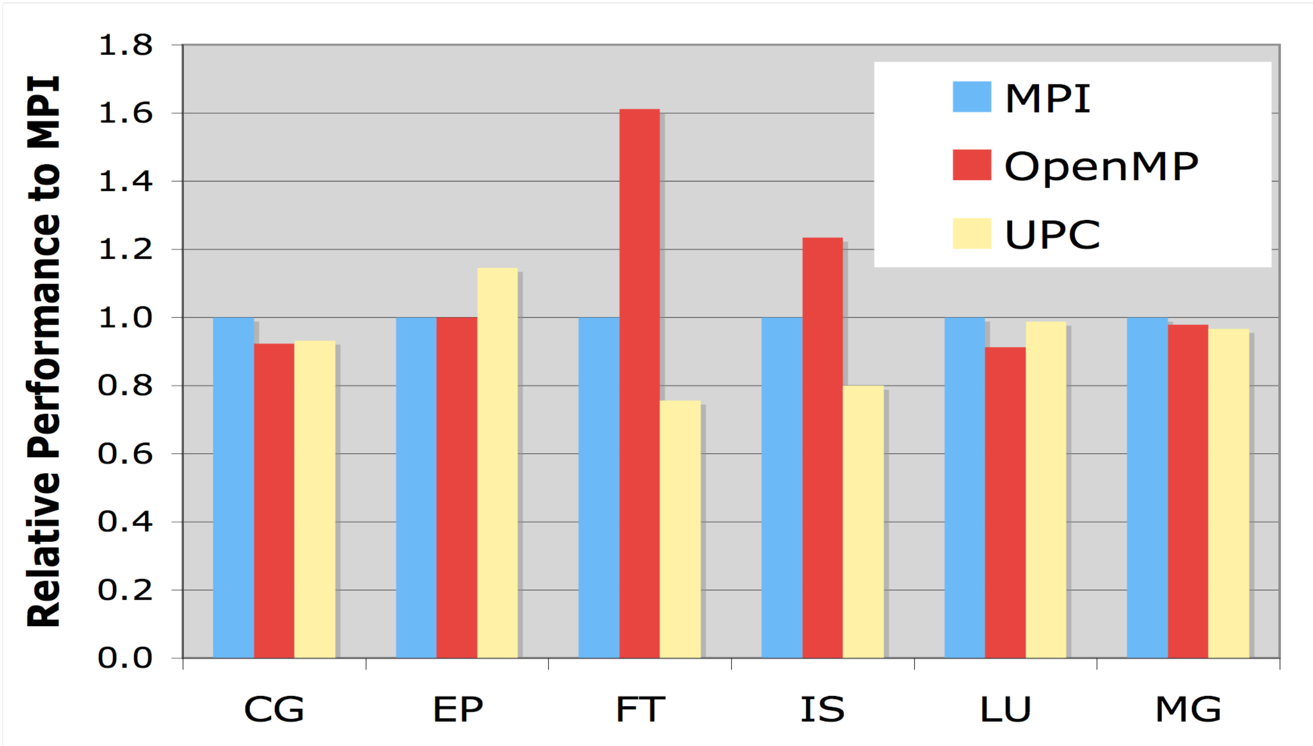
- **MPI uses most memory, UPC uses slightly less**
- **OpenMP saves great due to direct data access**

Memory Usage : MPI+OpenMP



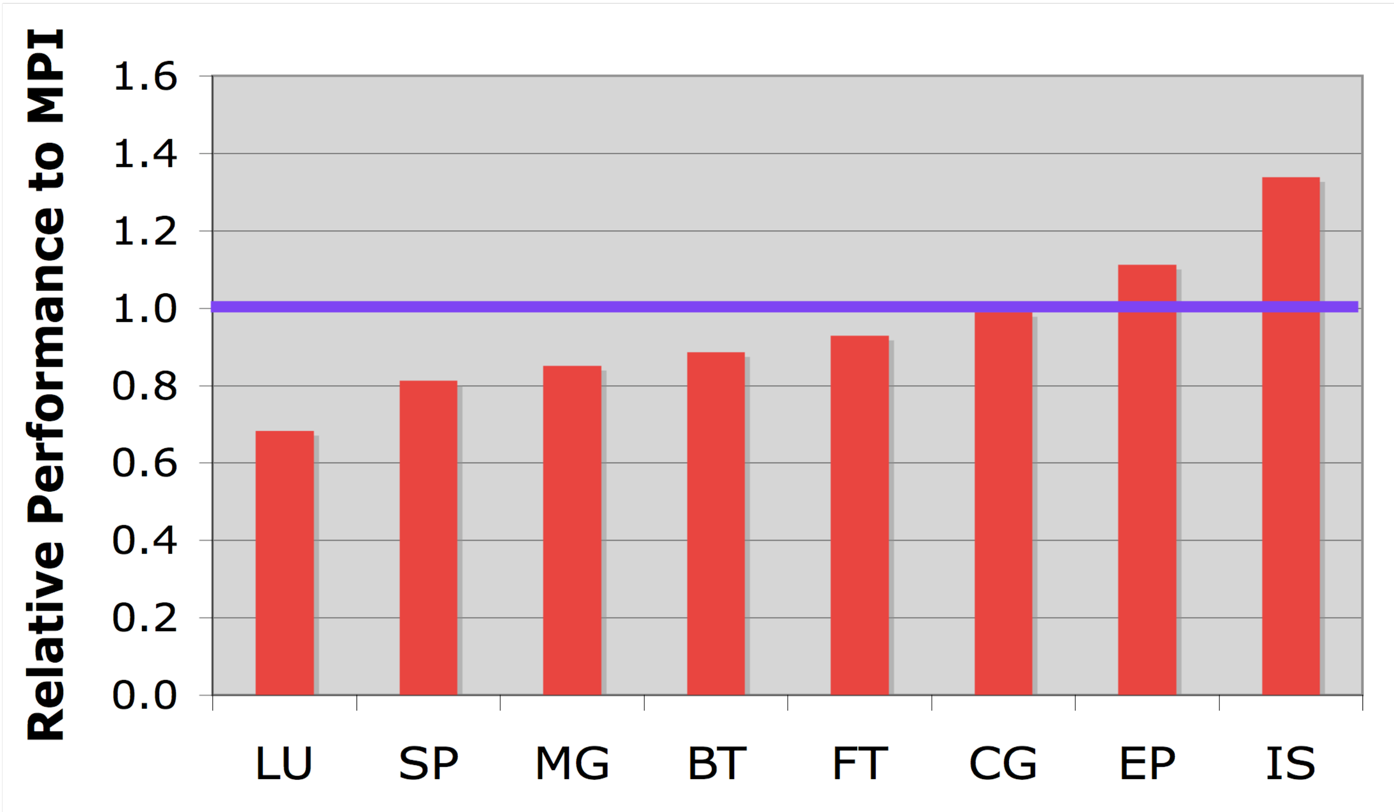
- **Using more OpenMP threads could reduce the memory usage substantially, up to five times on Hopper (eight-core nodes)**

Performance: Using One Node on Hopper



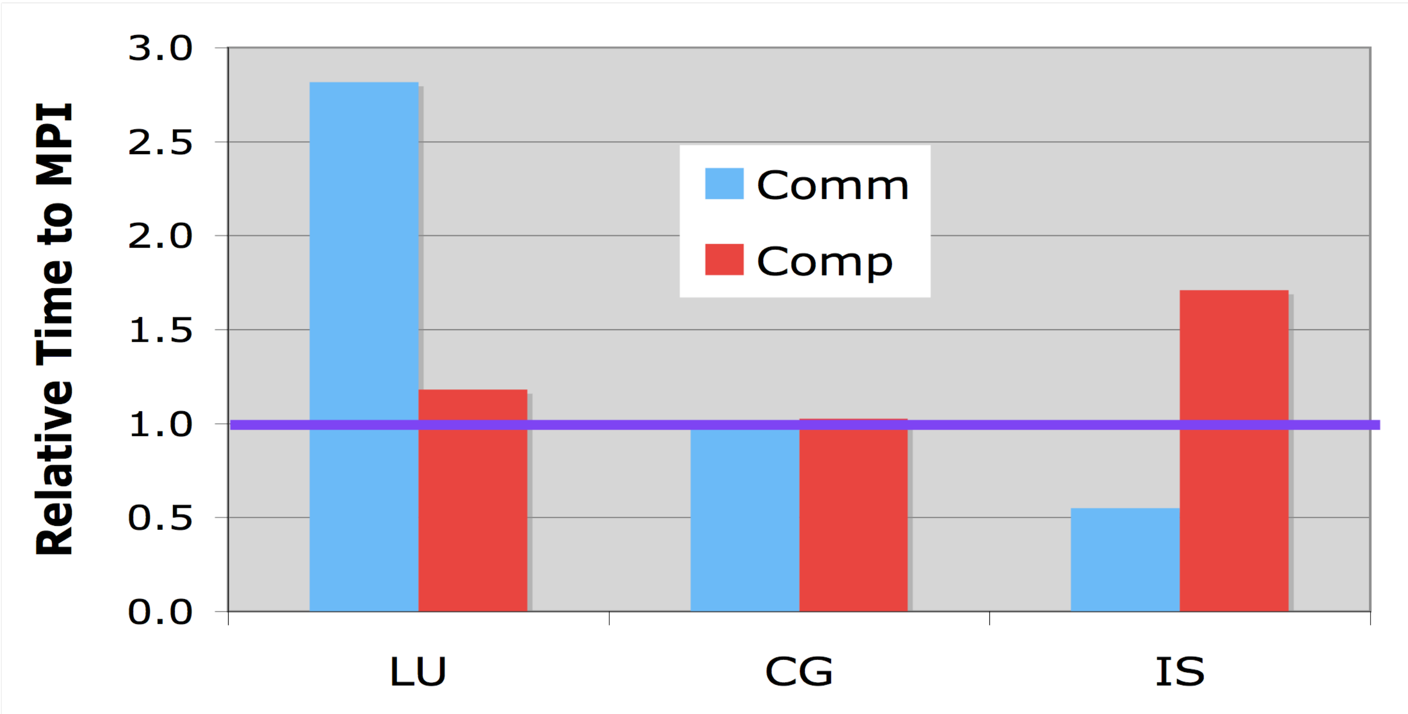
- **Similar performance for CG, EP, LU, MG**
- **For FT, IS, OpenMP delivers significantly better performance due to efficient programming**

Performance: MPI vs. UPC



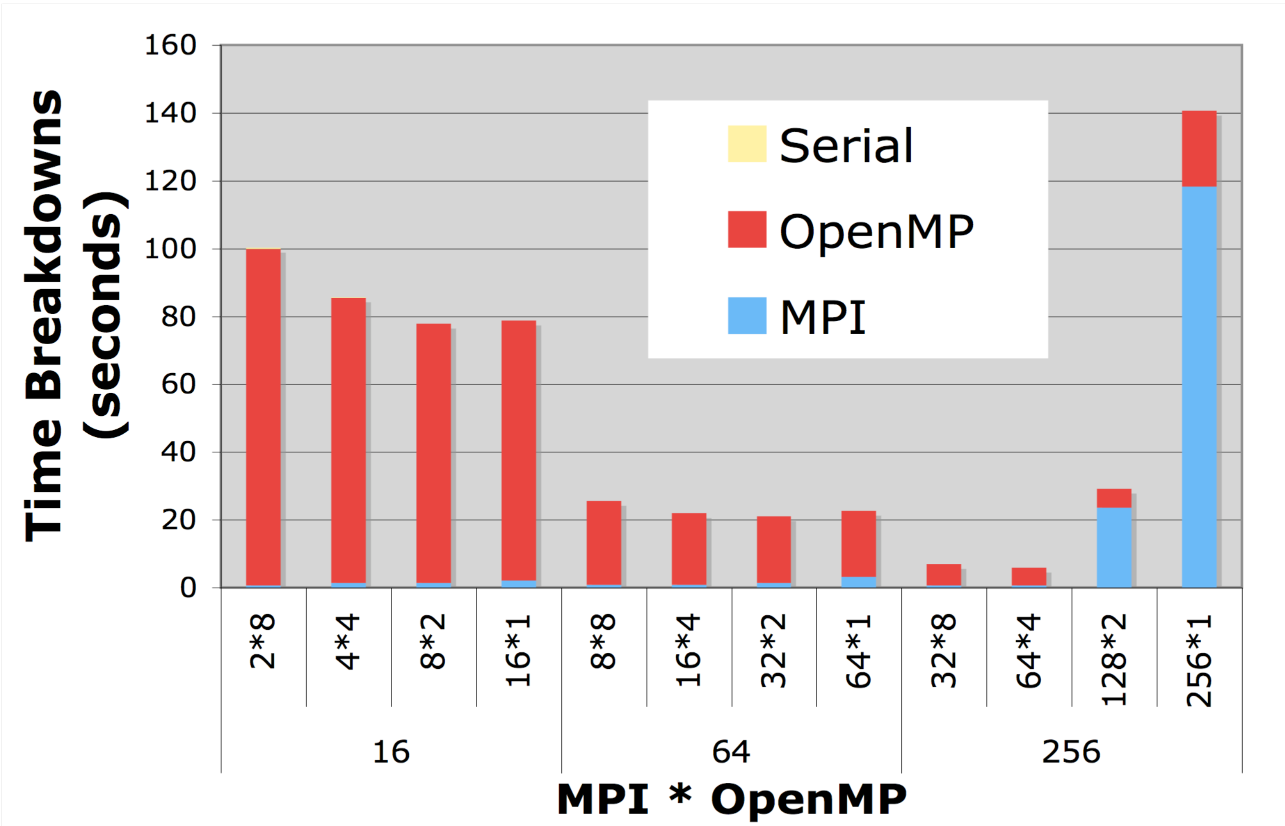
- **UPC performs better for EP and IS, close to CG, and worse for others**

Time Breakdown: MPI vs. UPC



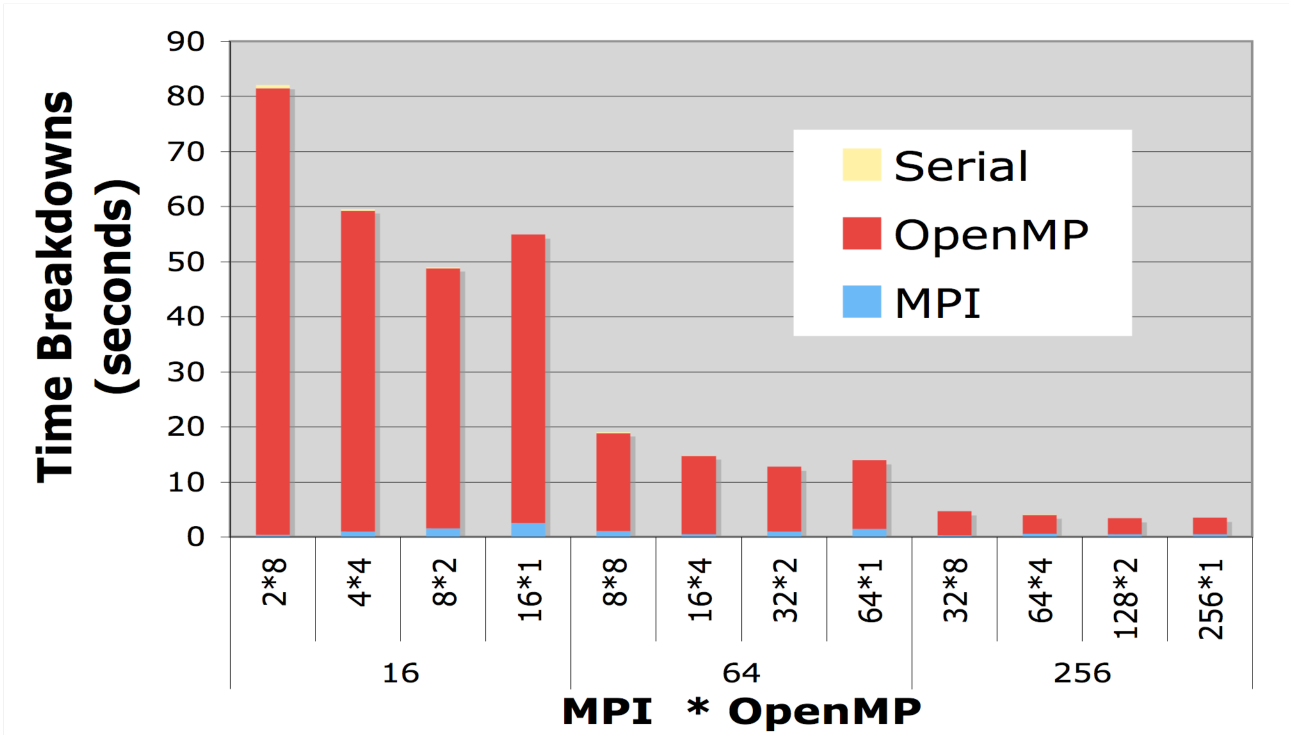
- For LU, the longer communication time for UPC is probably due to lack o efficient point-to-point synchronization
- For IS, the one-sided upc_memget/upc_memput is much more efficient than the MPI_alltoallv function

Performance: BT-MZ (MPI+OpenMP)



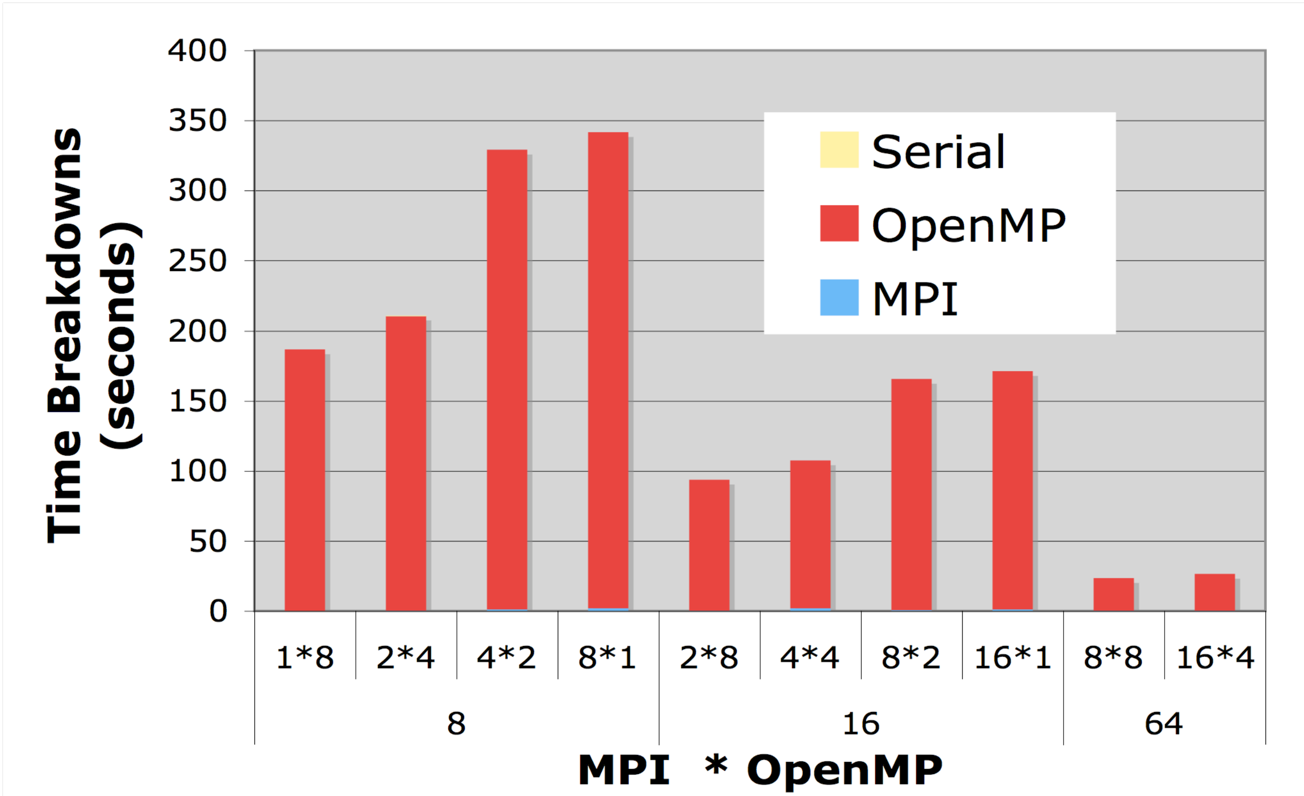
- **MPI suffers loan imbalance for higher number of MPI tasks**
- **Best performance obtained when OpenMP=2**

Performance: SP-MZ (MPI+OpenMP)



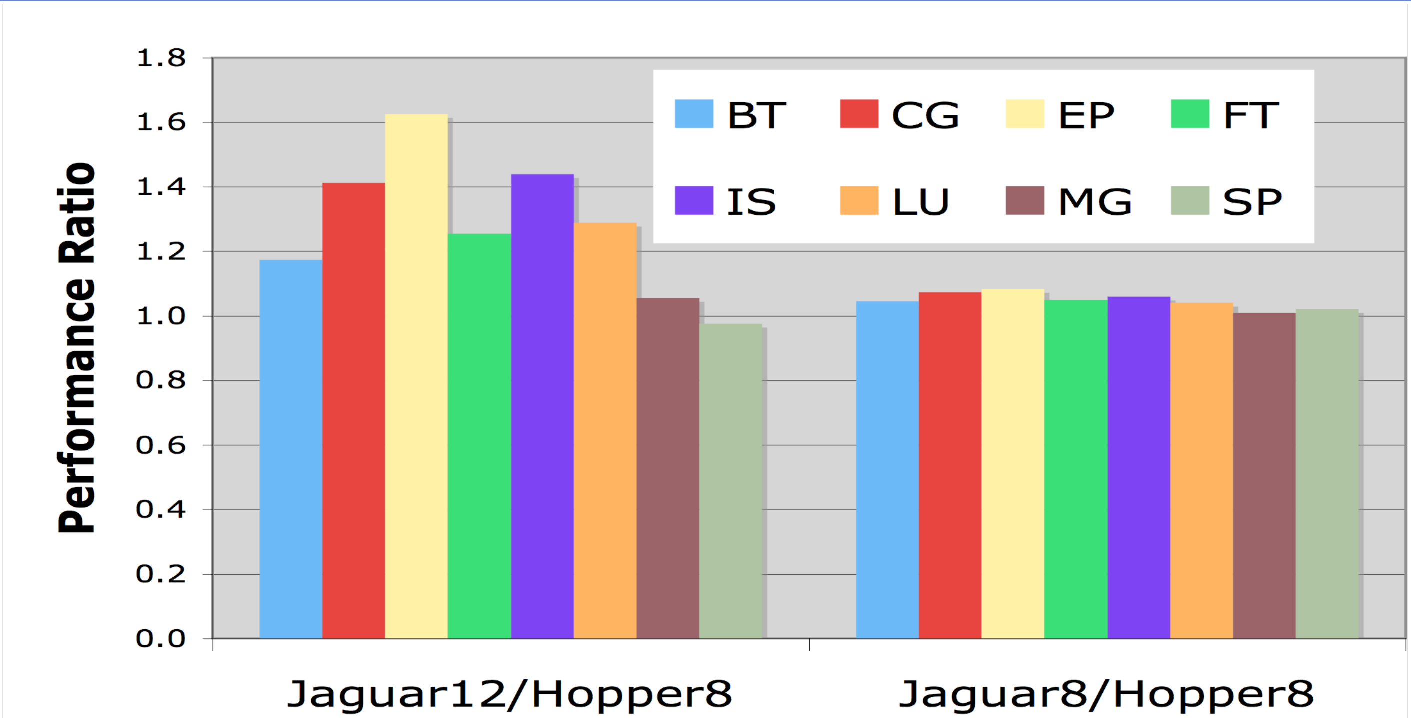
- Time is dominated by OpenMP
- Performance scales well
- Best performance obtained when OpenMP=2

Performance: LU-MZ (MPI+OpenMP)



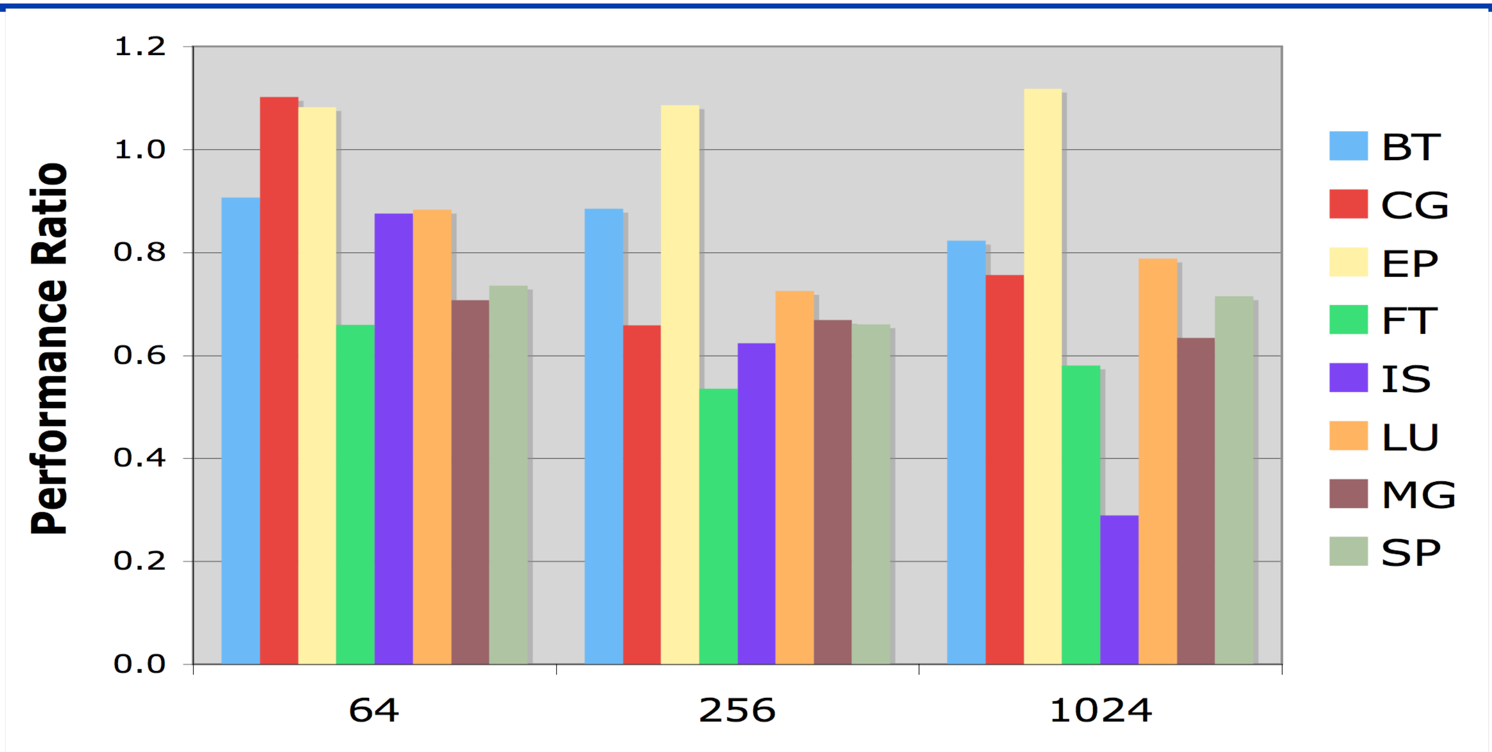
- **Best performance obtained when OpenMP=8 due to larger cache size and enough work in OpenMP region to amortize the OpenMP overhead**

Jaguar vs. Hopper: Single Node



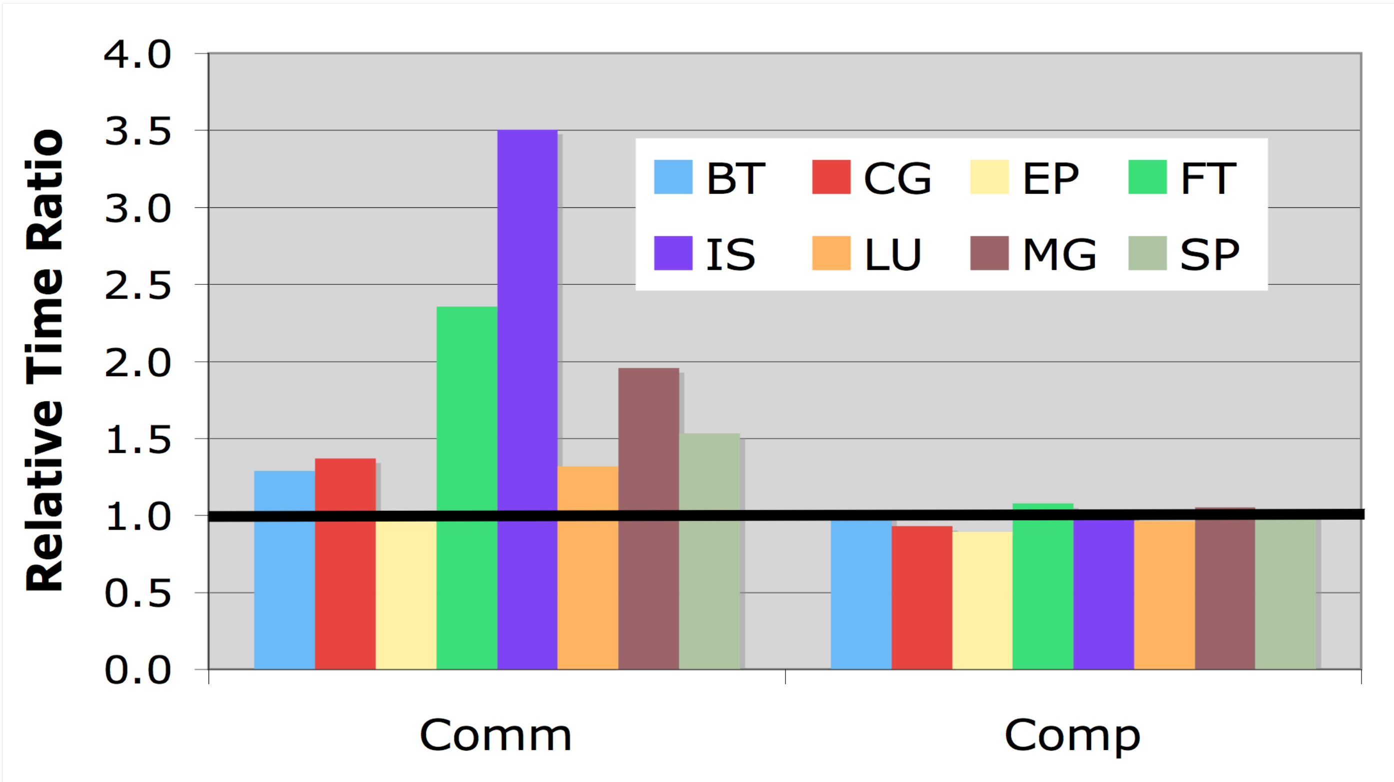
- **Using 8 cores on Jaguar, deliver similar performance**
- **Using 12 cores on Jaguar:**
  - EP 1.6 times better due to higher CPU frequency
  - CG, IS, better performance due to larger aggregate cache size
  - MG, SP worse performance due to memory contention

Jaguar vs. Hopper: MPI Across Nodes



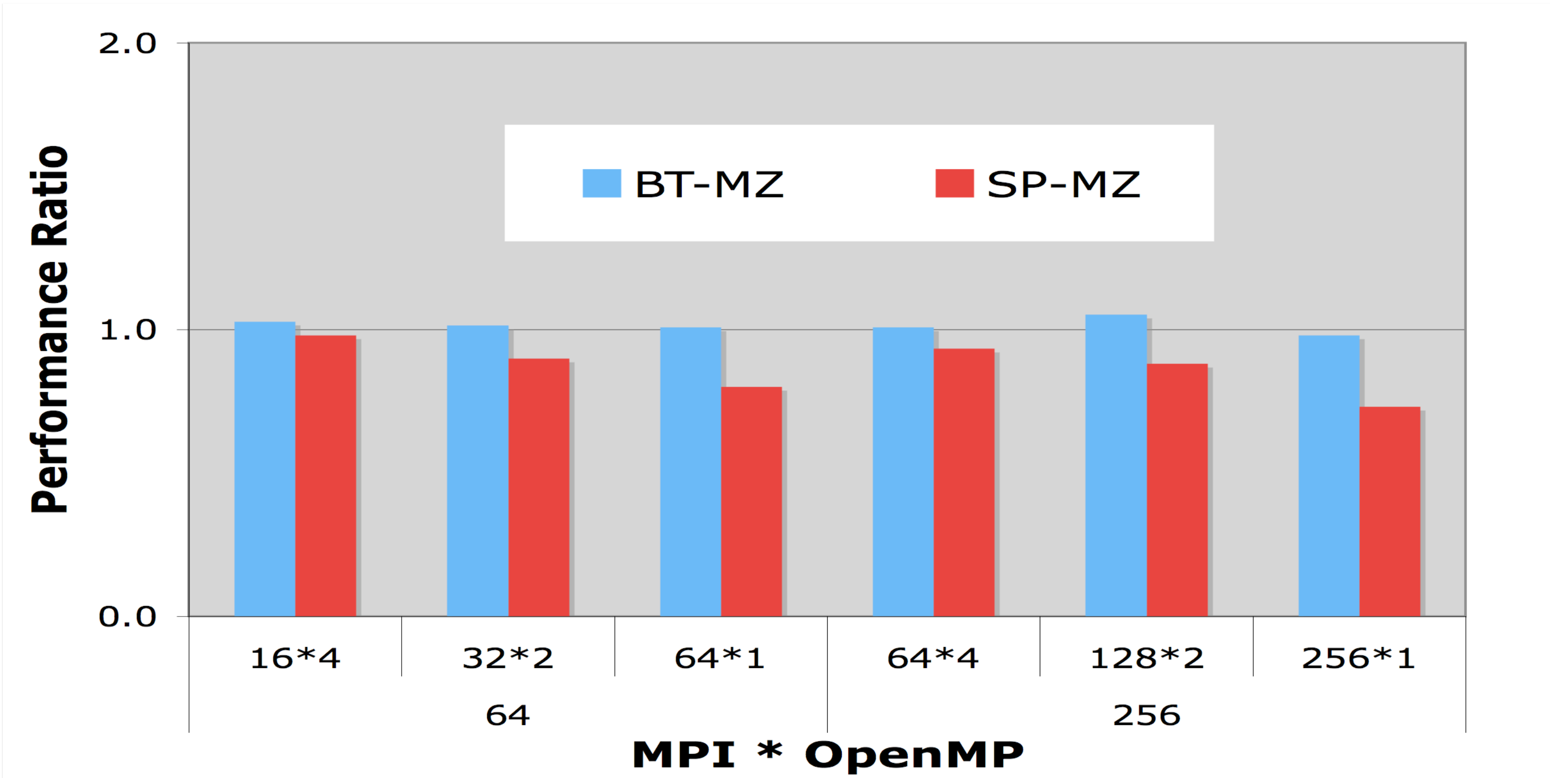
- **EP, computation intensive application, consistently better**
- **IS performs worst due to higher communication contention**

Jaguar vs. Hopper:
Time Breakdown for MPI on 1024 Cores



- **Computation time similar between Jaguar and Hopper**
- **Communication time higher on jaguar except EP**

Jaguar vs. Hopper: Hybrid (MPI+OpenMP)



- **For BT-MZ, similar performance**
- **For SP-MZ, Jaguar is worse due to higher network contention**
- **Using more OpenMP threads could reduce the performance gap**

Conclusion and Future Work (1)

- **Memory Usage**
  - MPI consumes most, UPC is slightly less, OpenMP saves greatly
  - Using more OpenMP threads could save up to several times amount of memory usage for MPI+OpenMP hybrid model
- **Performance**
  - On single node, OpenMP performs best due to its efficient programming and direct data access
  - Across nodes, overall, UPC performs slightly worse now, but delivers much better performance for IS, the communication intensive application
  - Hybrid MPI+OpenMP codes in favor of using more OpenMP threads, the best performance depends on the tradeoff between OpenMP overhead and larger cache effects

Conclusion and Future Work (2)

- **Jaguar vs. Hopper**
  - Using hex-core may cause more memory contention, slowing down the performance
  - Using hex-core may cause more network contention, degrading the performance, hurting the scalability
  - Only favors computation intensive applications, such as EP.
- **Future Work**
  - Examine on larger node architectures
  - New Programming Models or MPI + x or ?