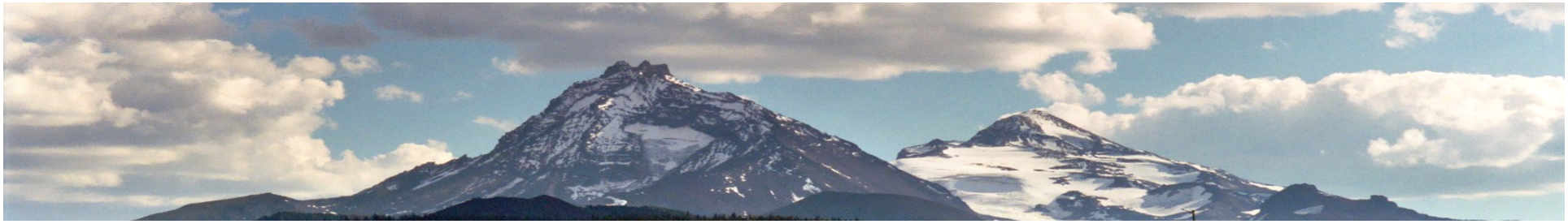


Lecture 7: TAU Applications

Allen D. Malony
Department of Computer and Information Science

UNIVERSITY OF OREGON

Short Course Outline

- Lecture 1: Introduction and Fundamentals
- Lecture 2: Methodology
- Lecture 3: Tools Technology
- Lecture 4: Tools Landscape – Part 1
- Lecture 5: Tools Landscape – Part 2
- Lecture 6: TAU Performance System
- Lecture 7: TAU Applications
- Lecture 8: Advances in TAU
- Lecture 9: Future Directions

Tuesday

Wednesday

Thursday

Outline

- Performance problem solving objectives
- Applications
  - S3D: turbulent combustion
  - NWChem: computational chemistry
  - MPAS: atmospheric simulation
  - XGC: gyrokinetic microturbulence
  - IRMHD: magneto hydrodynamics
  - CESM: earth simulation

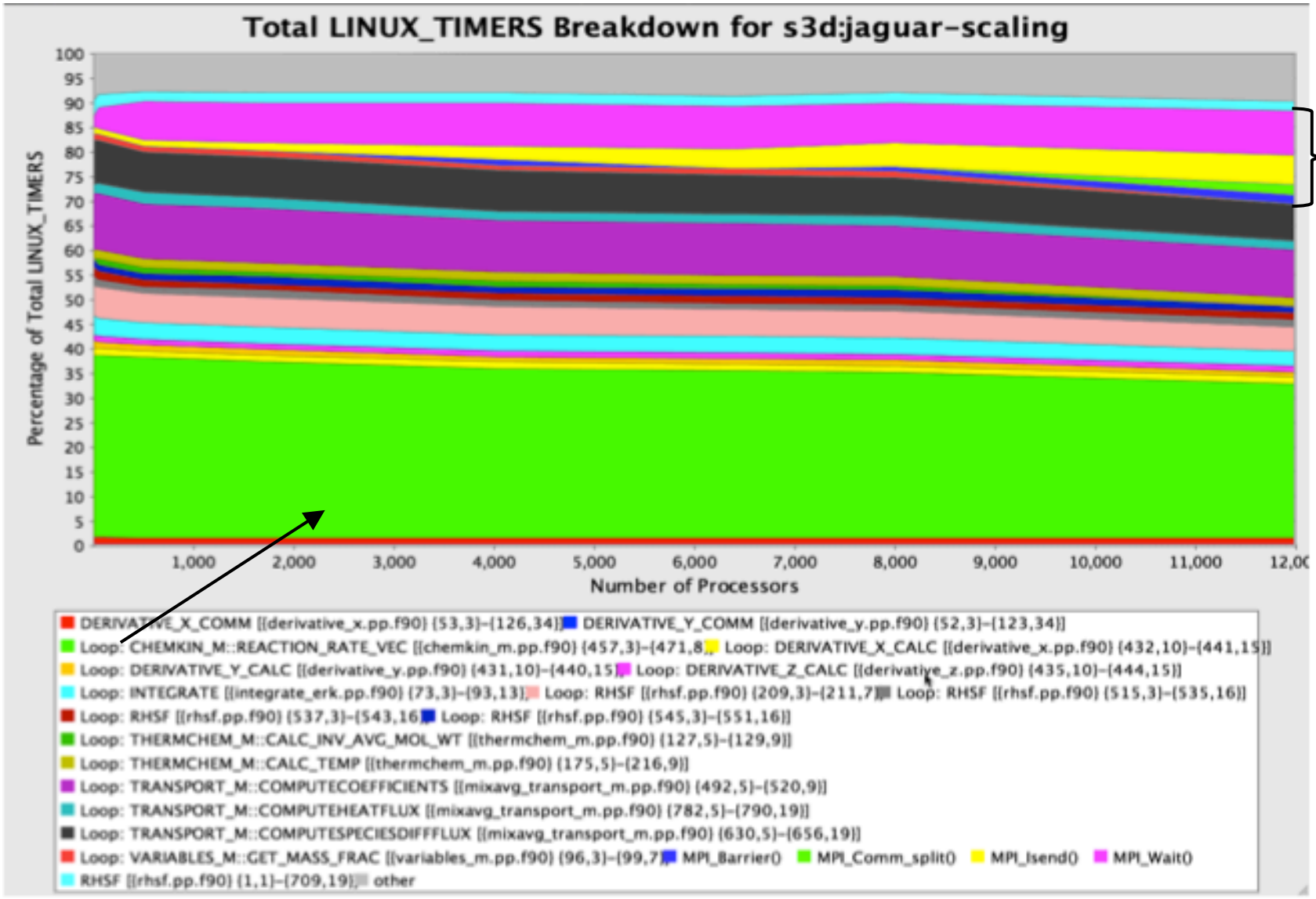
Performance Problem Solving Goals

- Answer questions at multiple levels of interest
  - Performance experiments to identify problems
    - characterization from performance data measurements
  - High-level performance across dimensions
    - machine, applications, code revisions, data sets
    - examine broad performance trends
- Discover general correlations
  - Performance features between experiments
  - Identify primary performance behavior across parameters
  - Conduct meta analysis to uncover principal factors
- Feedback performance problem identification and knowledge about features/factors to optimize

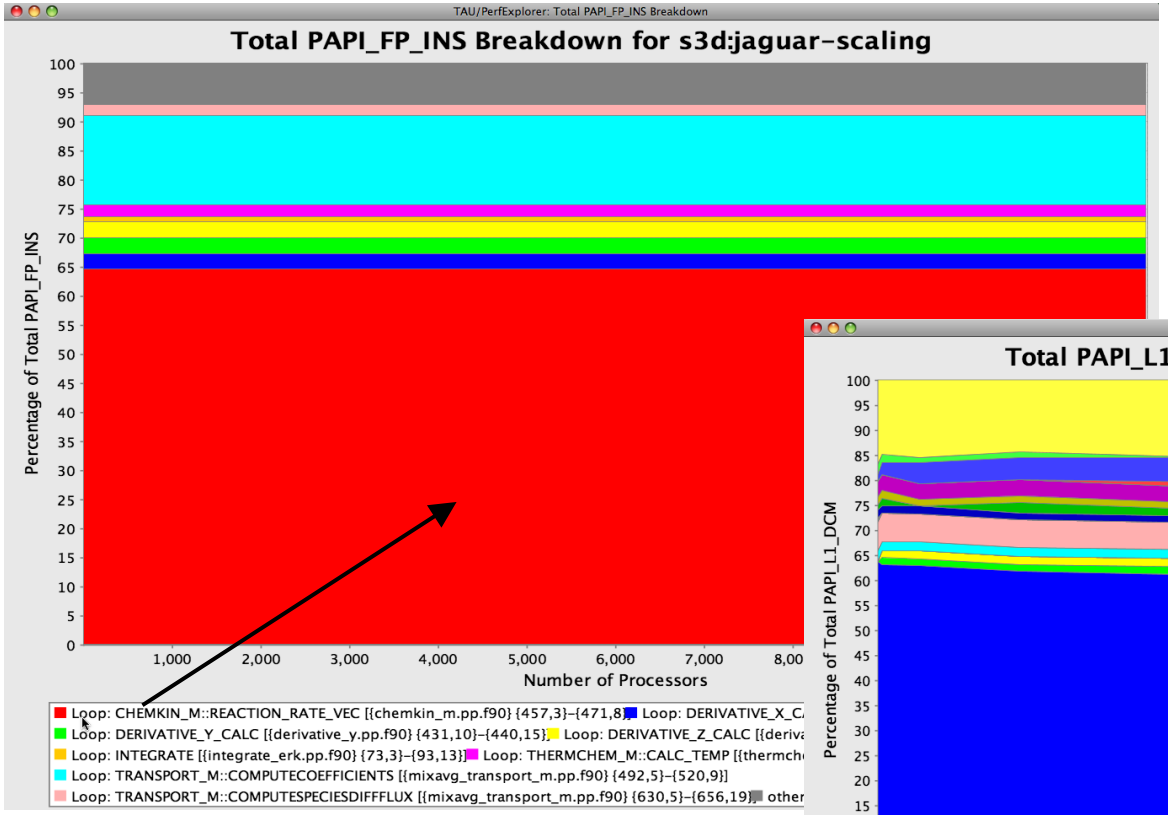
S3D Scalability Study

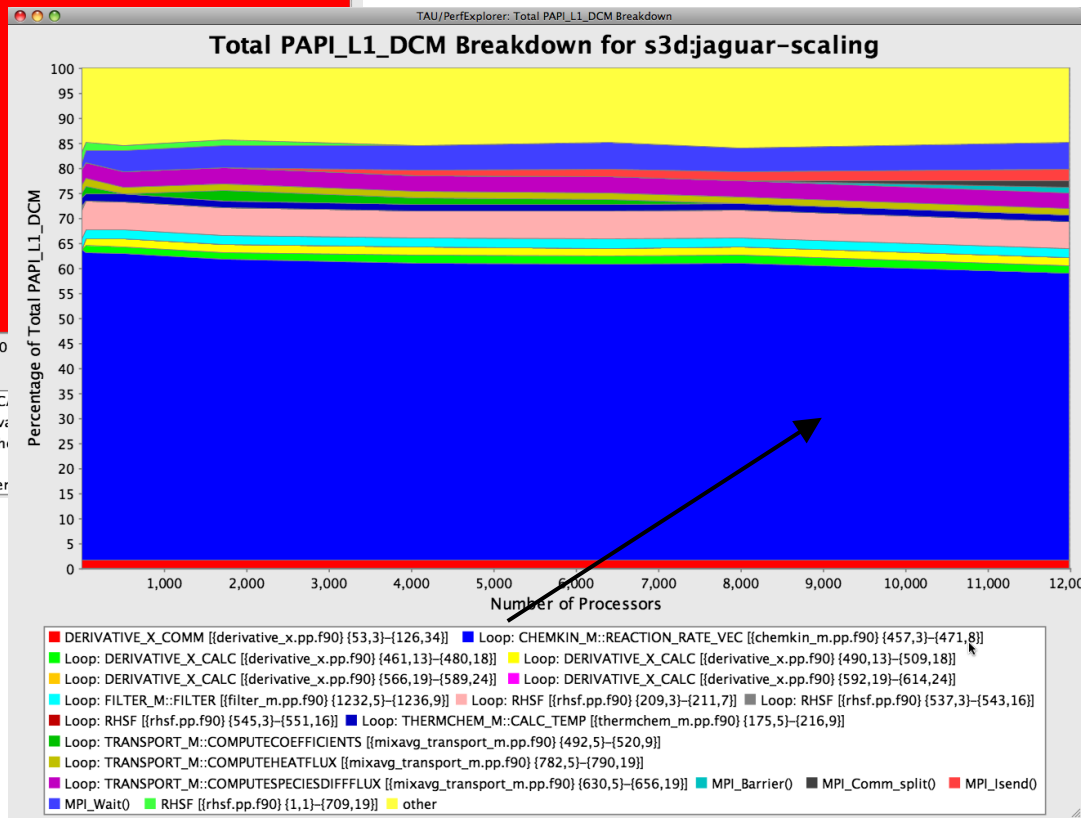
- S3D flow solver for turbulent combustion simulation
- Performance scaling study (C2H4 benchmark)
  - Cray XT3+XT4 hybrid, XT4 (ORNL, Jaguar)
  - IBM BG/P (ANL, Intrepid)
  - Weak scaling (1 to 12000 cores)
  - Evaluate scaling of code regions and MPI
- Demonstration of scalable performance measurement, analysis, and visualization
- Understanding scalability
  - Requires environment for performance investigation
  - Performance factors relative to main events

S3D Total Runtime Breakdown by Events (Jaguar)

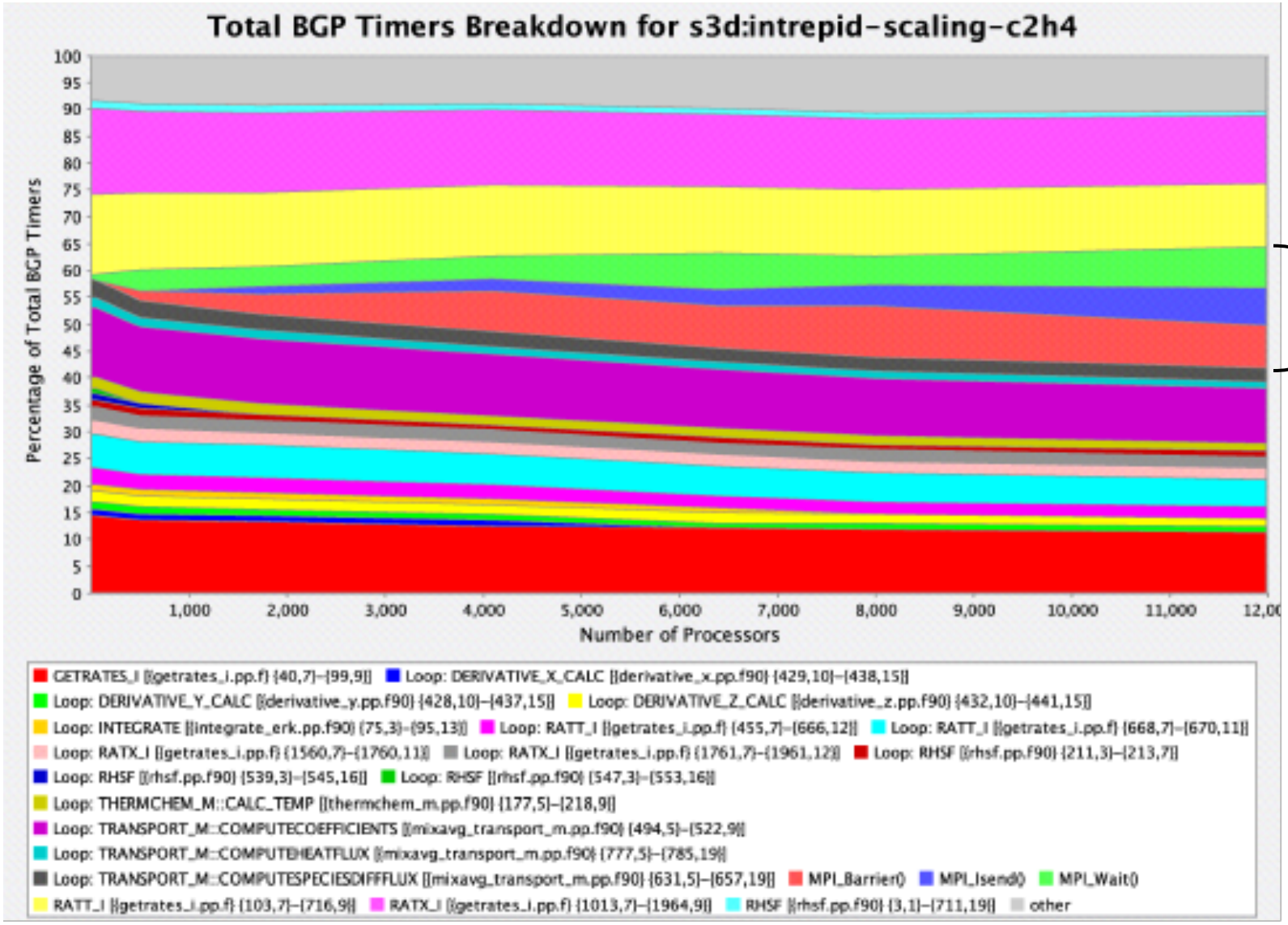


S3D FP Instructions/L1 Data Cache Miss (Jaguar)

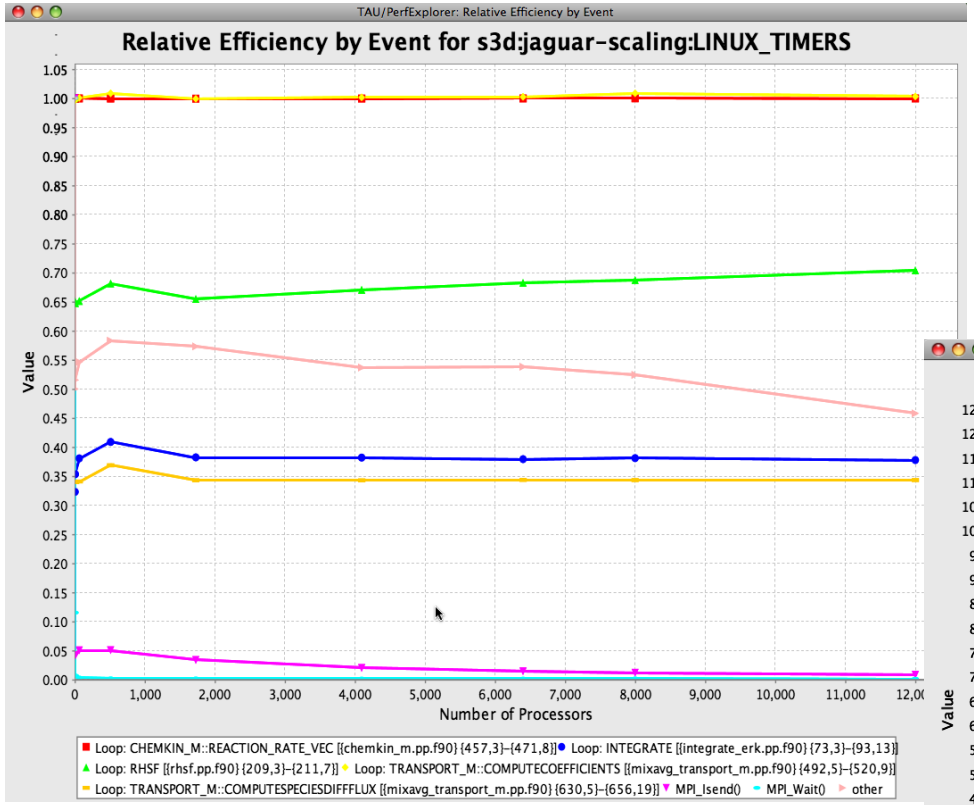


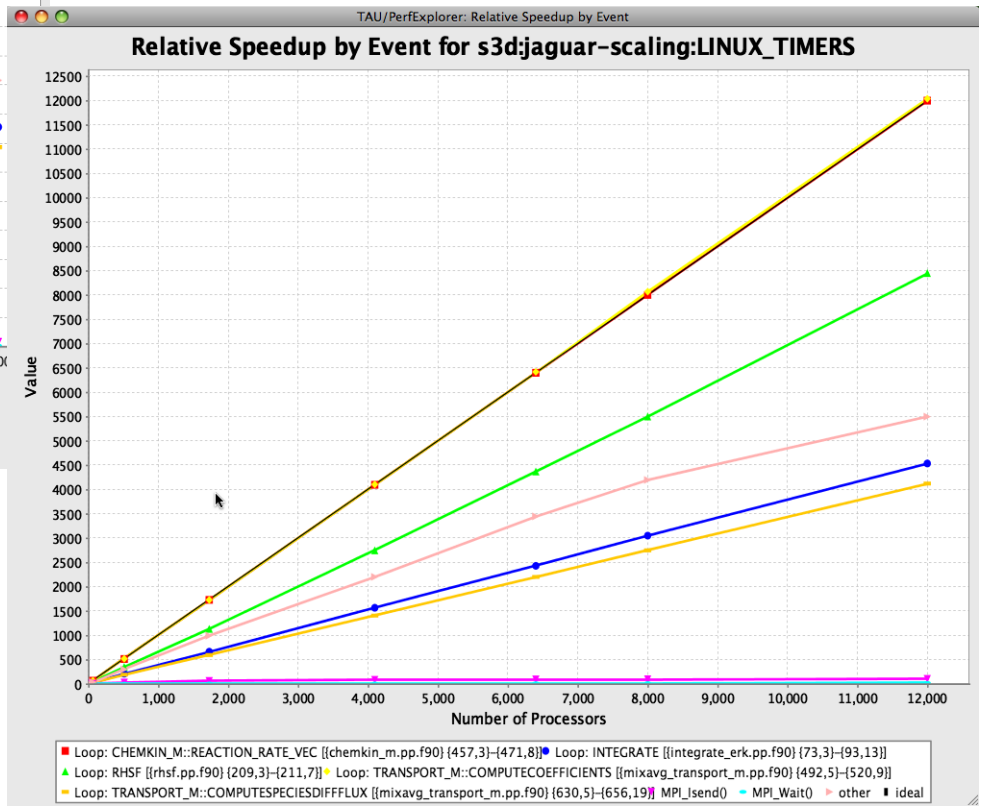


S3D Total Runtime Breakdown by Events (Intrepid)

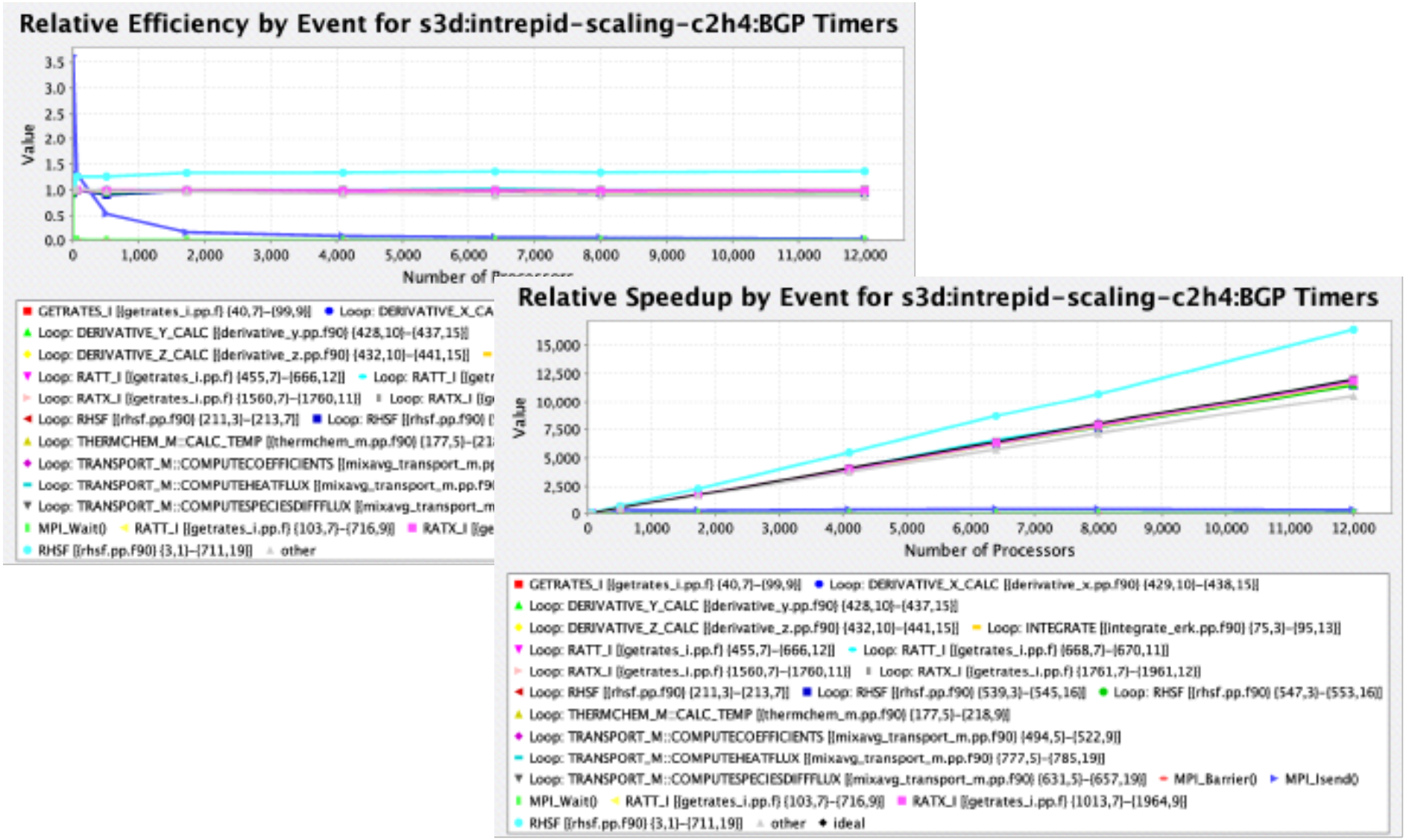


S3D Relative Efficiency/Speedup by Event (Jaguar)

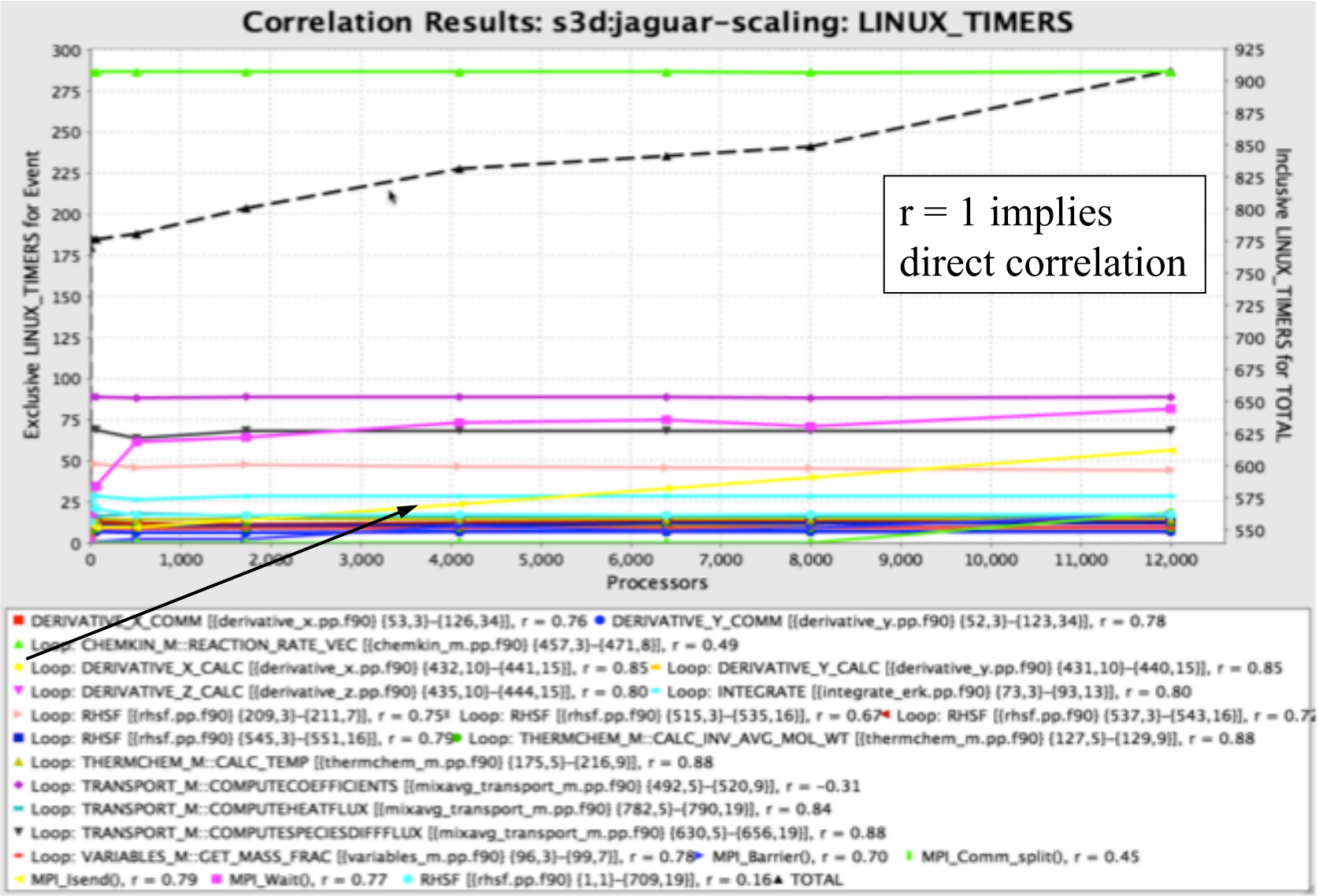




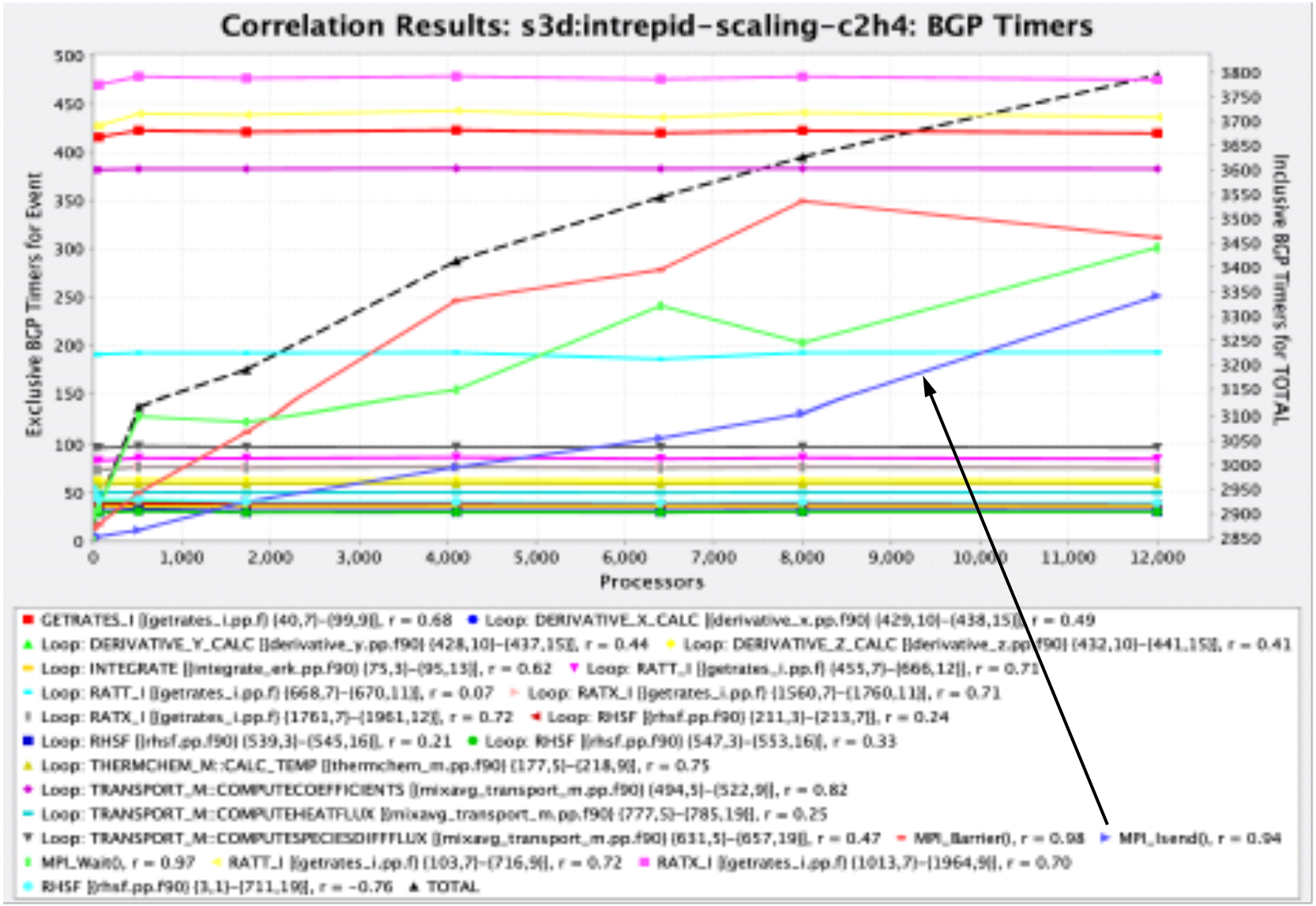
S3D Relative Efficiency by Event (Intrepid)



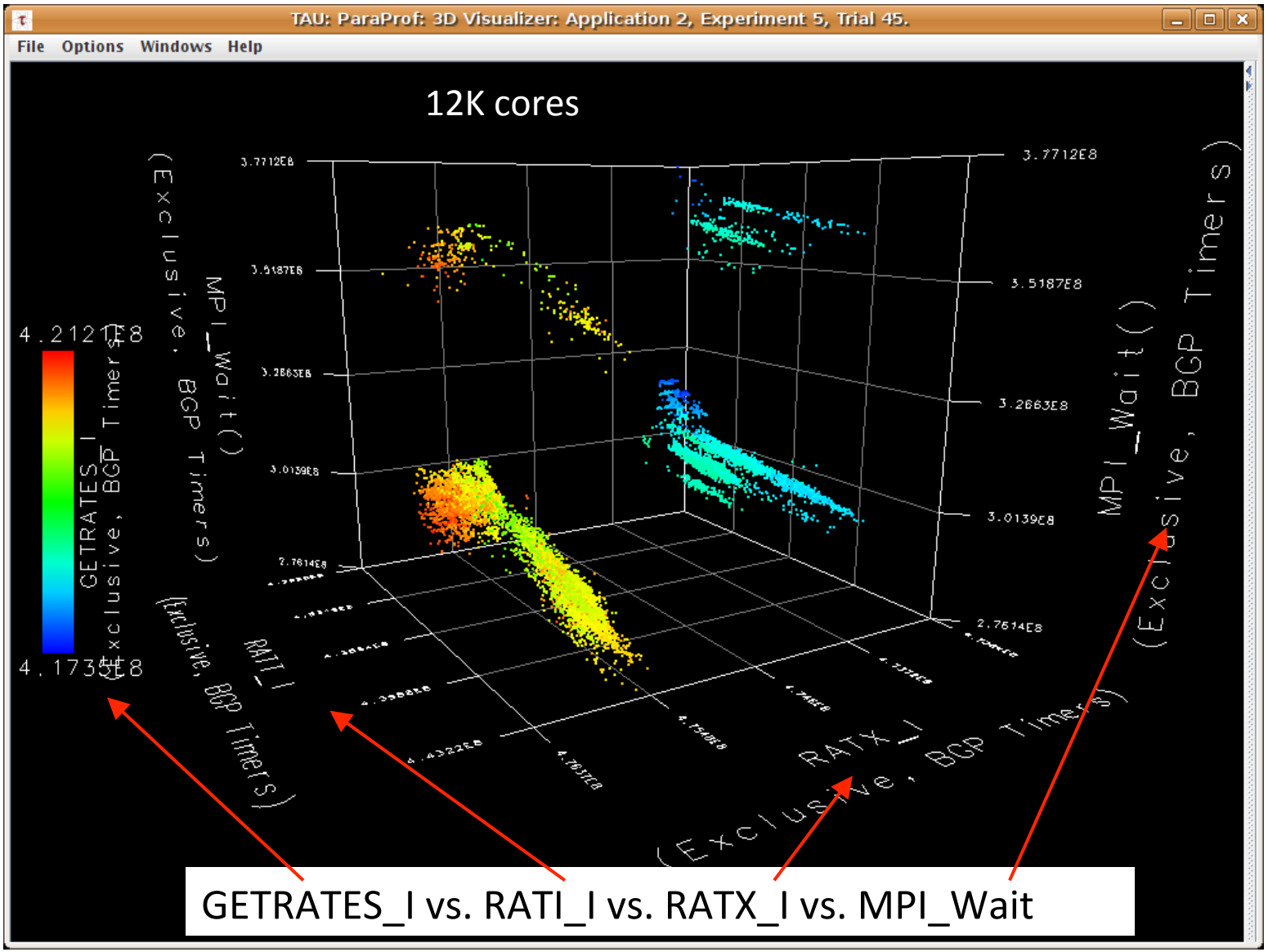
S3D Event Correlation to Total Time (Jaguar)



S3D Event Correlation to Total Time (Intrepid)

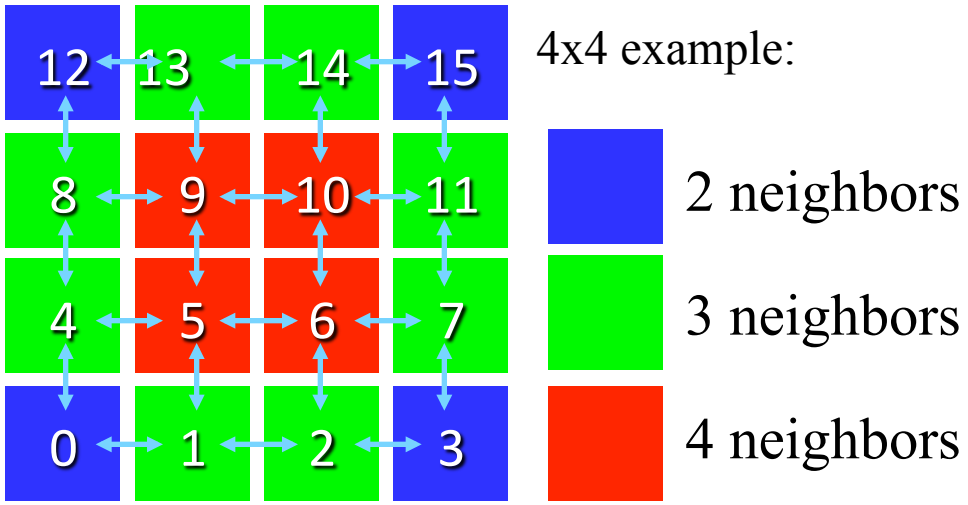


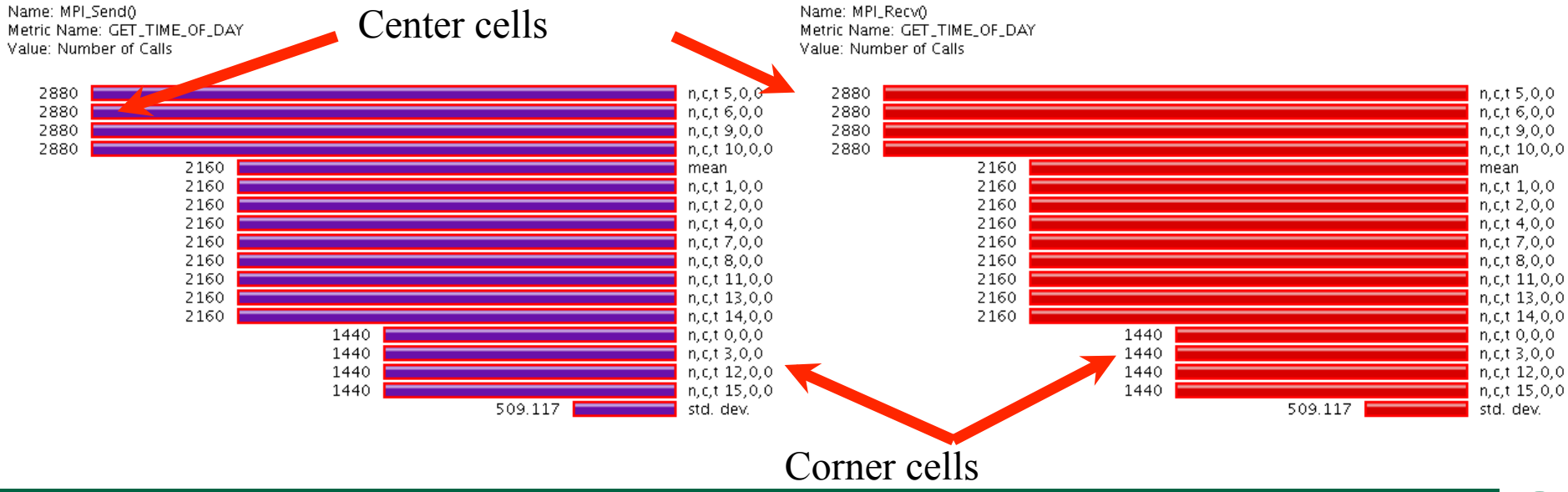
S3D 3D Correlation Cube (Intrepid, MPI_Wait)



S3D Computational Structure

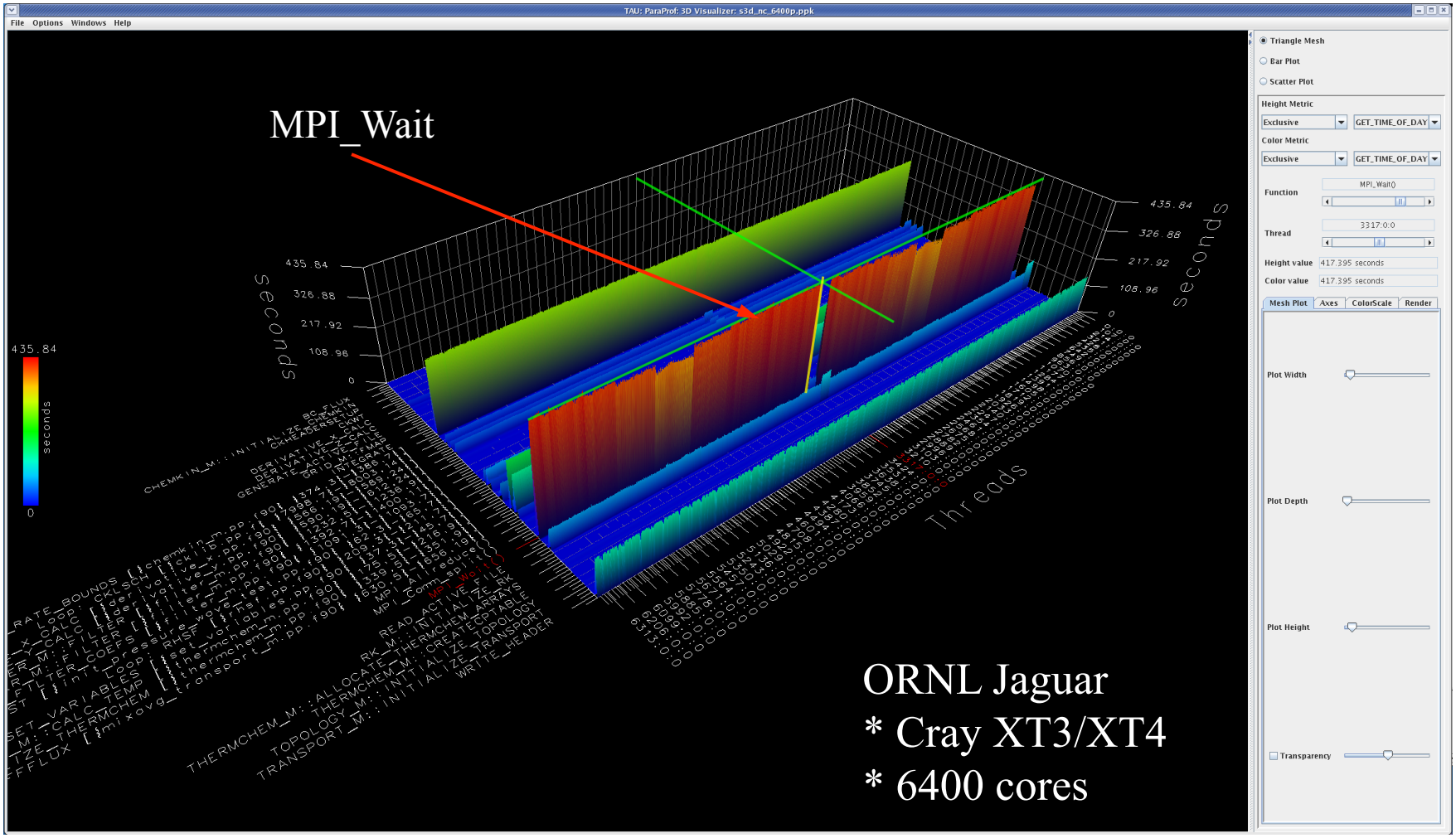
- Domain decomposition with wavefront evaluation and recursion dependences in all 3 grid directions
- Communication affected by cell location



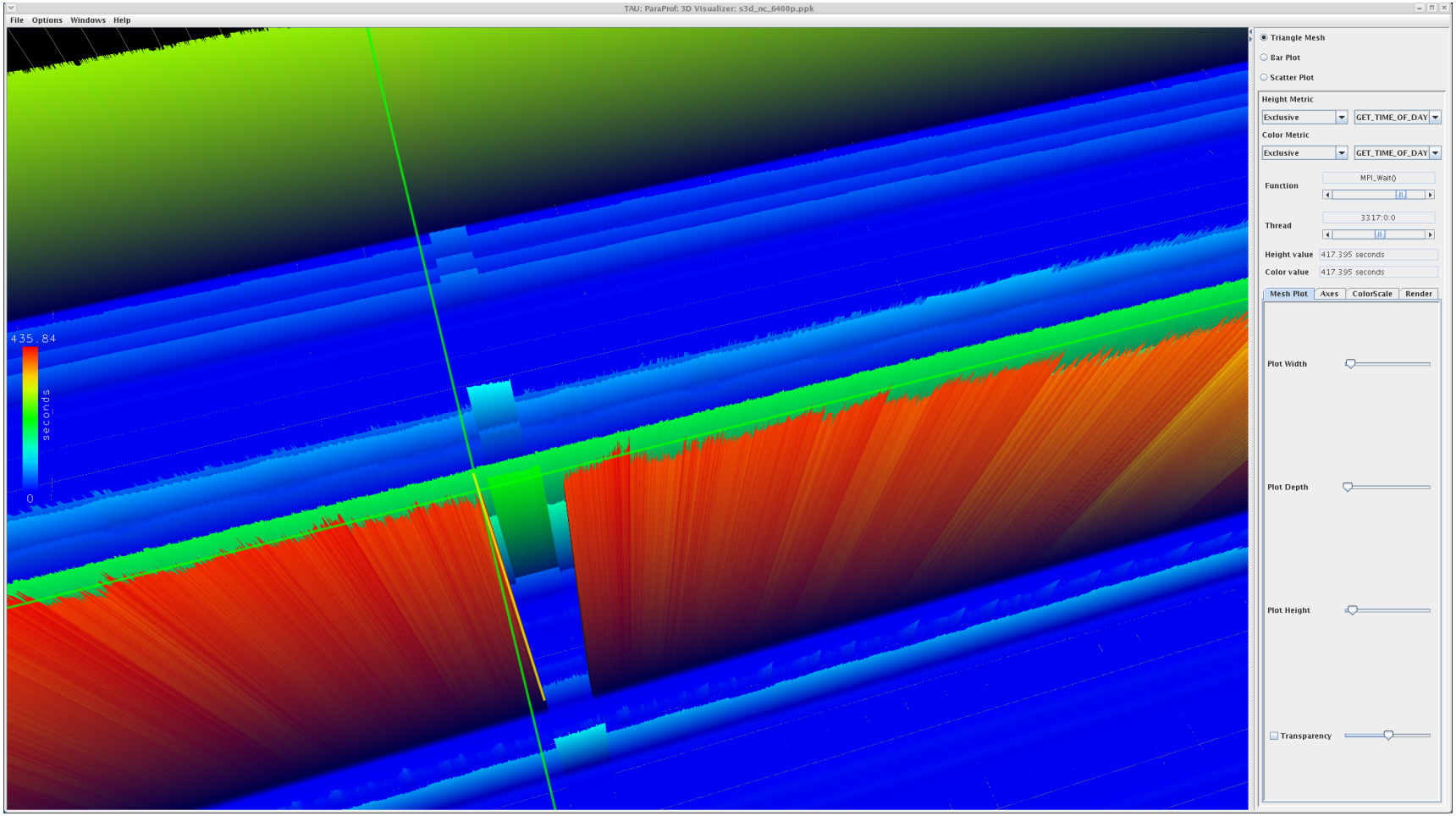


S3D on a Hybrid System (Cray XT3 + XT4)

- 6400 core execution on transitional Jaguar configuration



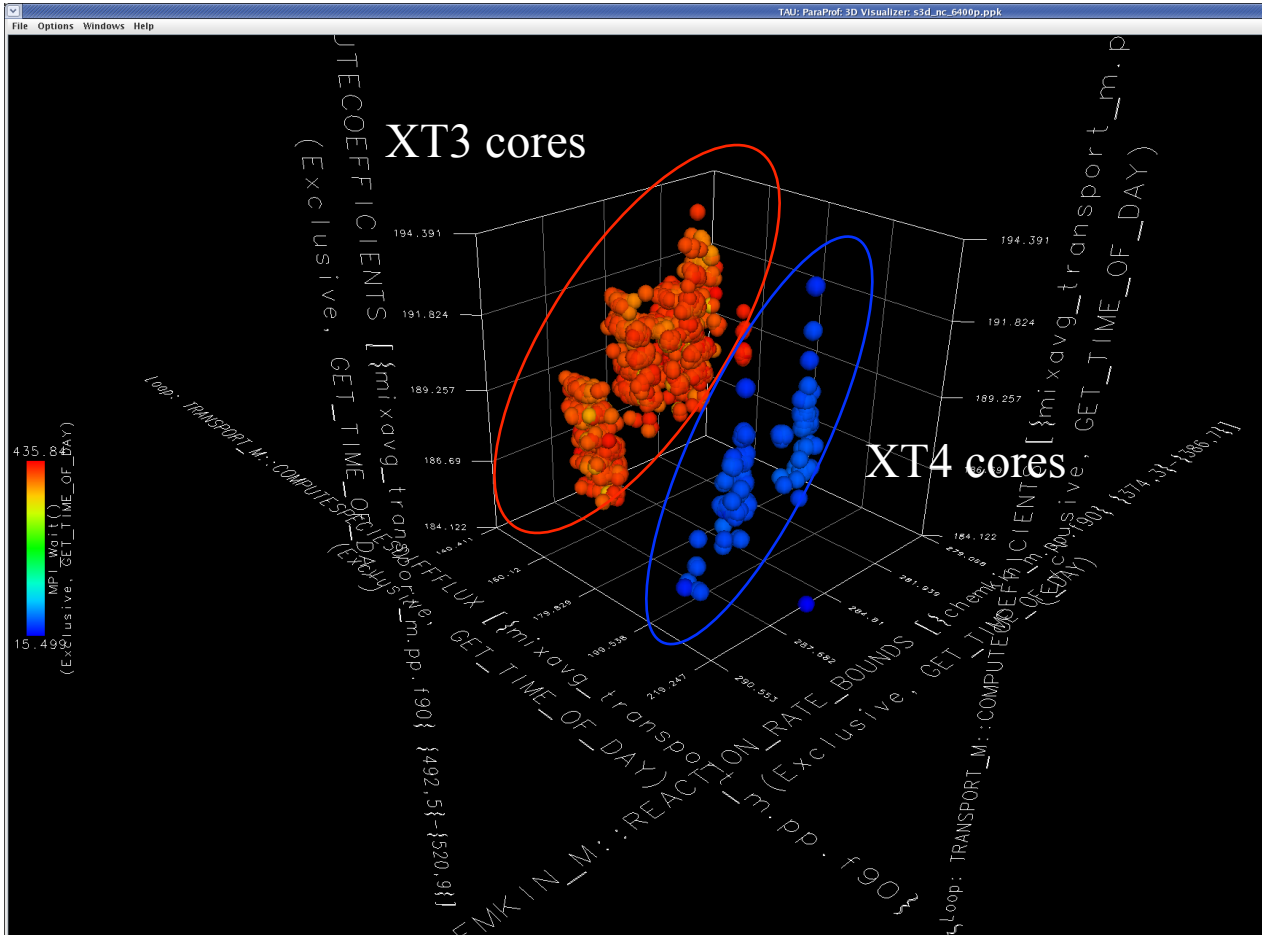
Zoom View of Parallel Profile (S3D, XT3+XT4)



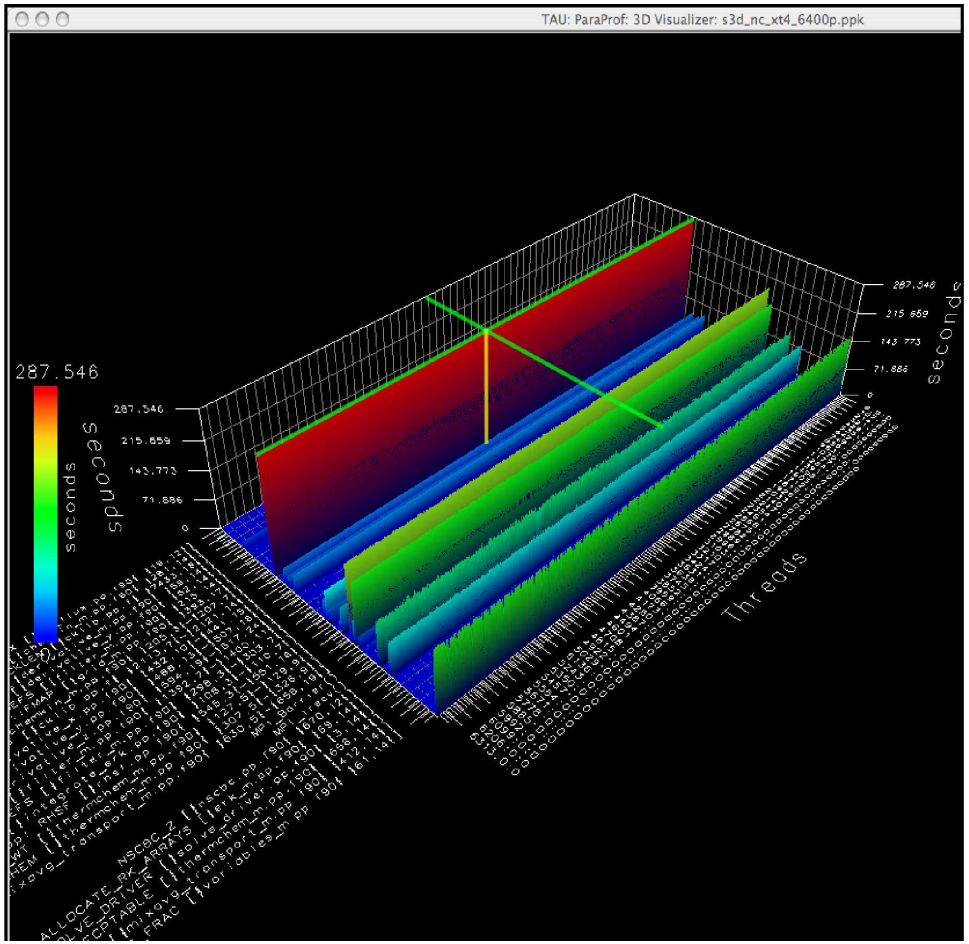
- Gap happens to be only XT3 nodes
  - *MPI_Wait* takes less time, other routines take more time

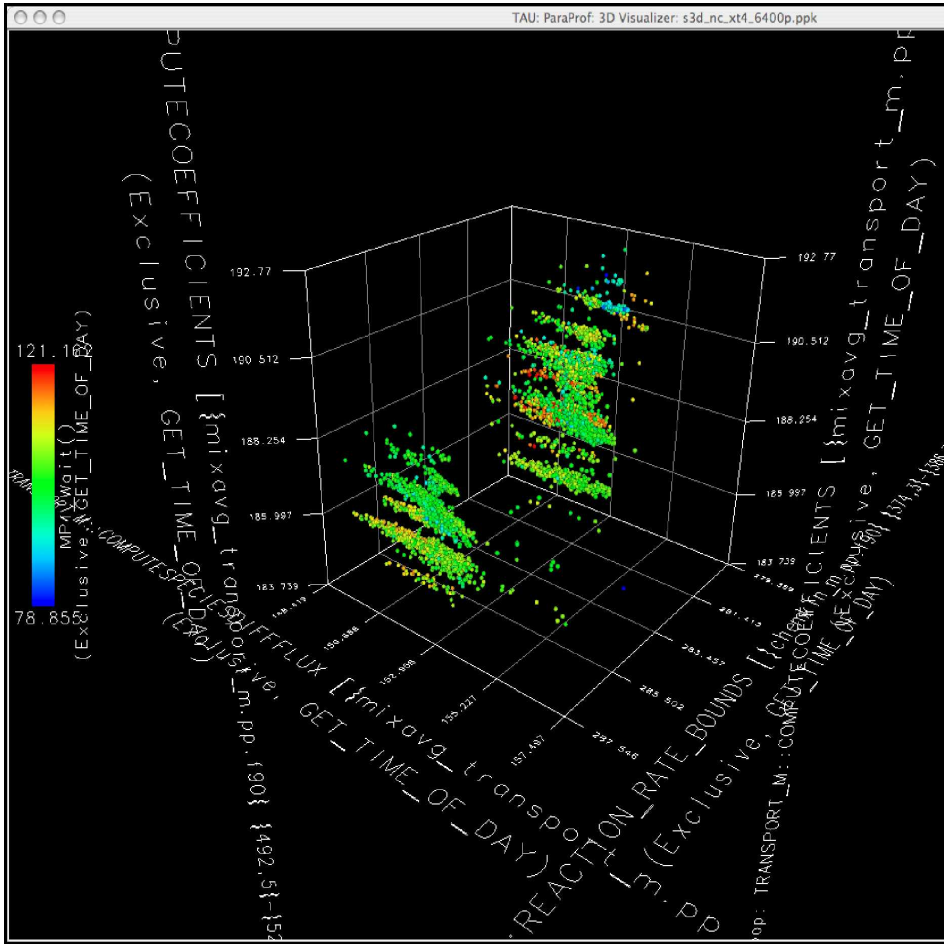
Scatterplot (S3D, XT3+XT4)

- Scatterplot of top three events colored by total time
- Two clusters
- Memory speed accounts for performance difference!!!
- Process metadata can identify the different nodes



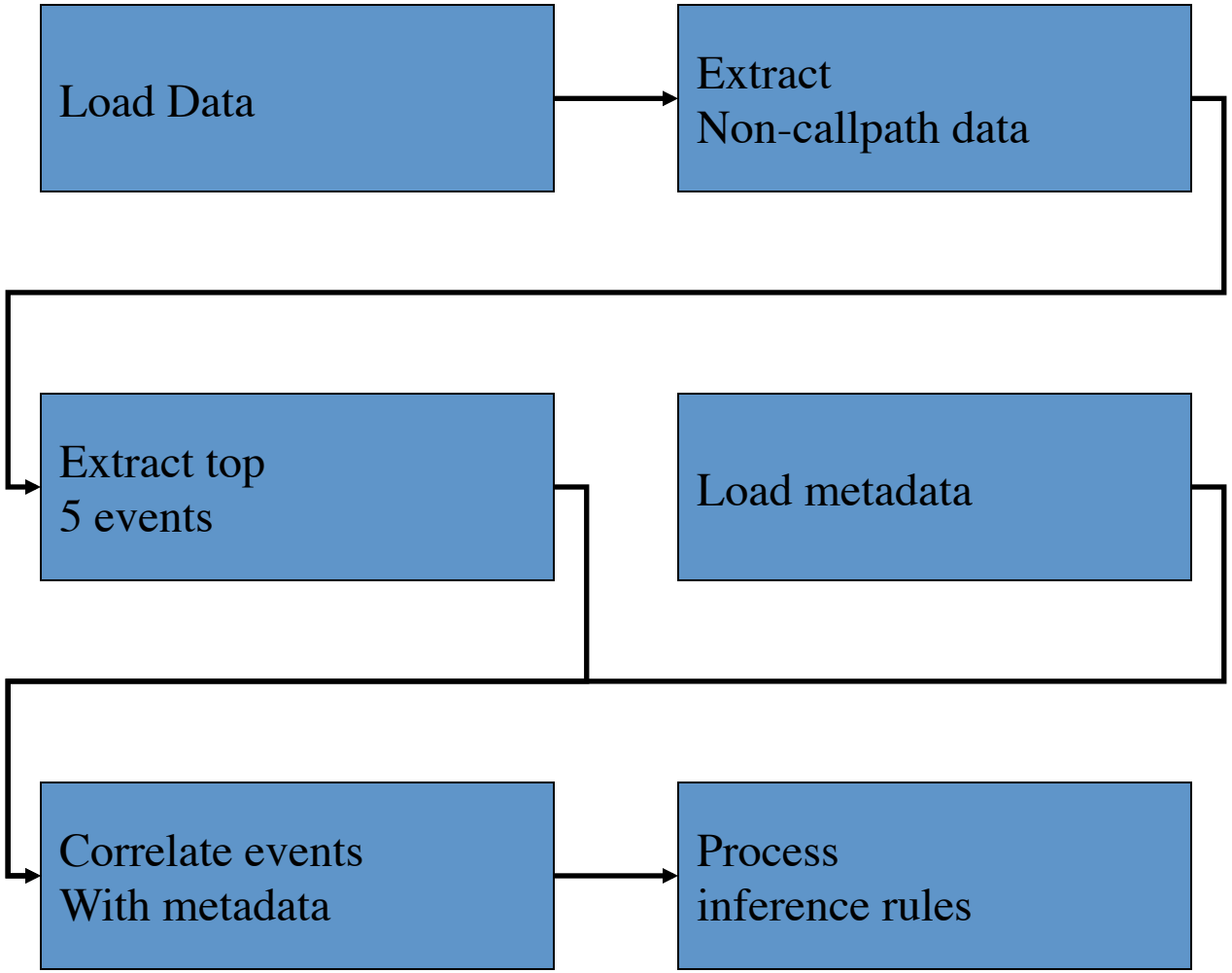
S3D Run on XT4 Only





Capturing Analysis and Inference Knowledge

- Create PerfExplorer workflow for S3D case study?



PerfExplorer Workflow Applied to S3D

---------- JPython test script start ----------
doing single trial analysis for sweep3d on jaguar
Loading Rules...
Reading rules: rules/GeneralRules.drl... done.
Reading rules: rules/ApplicationRules.drl... done.
Reading rules: rules/MachineRules.drl... done.
loading the data...
Getting top 10 events (sorted by exclusive time)...
Firing rules...
MPI_Recv(): "CALLS" metric is correlated with the metadata field "total neighbors".
The correlation is 1.0 (direct).
MPI_Send(): "CALLS"metric is correlated with the metadata field "total Neighbors".
The correlation is 1.0 (direct).
MPI_Send(): "P_WALL_CLOCK_TIME:EXCLUSIVE" metric is correlated with the metadata field "total neighbors".
The correlation is 0.8596 (moderate).
SOURCE [{source.f} {2,18}]: "PAPI_FP_INS:EXCLUSIVE" metric is inversely correlated with the metadata field "Memory Speed (MB/s)".
The correlation is -0.9792 (very high).
SOURCE [{source.f} {2,18}]: "PAPI_FP_INS:EXCLUSIVE" metric is inversely correlated with the metadata field "Seastar Speed (MB/s)".
The correlation is -0.9785258663321764 (very high).
SOURCE [{source.f} {2,18}]: "PAPI_L1_TCA:EXCLUSIVE" metric is inversely correlated with the metadata field "Memory Speed (MB/s)".
The correlation is -0.9818810020169854 (very high).
SOURCE [{source.f} {2,18}]: "PAPI_L1_TCA:EXCLUSIVE" metric is inversely correlated with the metadata field "Seastar Speed (MB/s)".
The correlation is -0.9810373923601381 (very high).
SOURCE [{source.f} {2,18}]: "PAPI_L2_TCM:EXCLUSIVE" metric is inversely correlated with the metadata field "Memory Speed (MB/s)".
The correlation is 0.9985297567878844 (very high).
SOURCE [{source.f} {2,18}]: "PAPI_L2_TCM:EXCLUSIVE" metric is inversely correlated with the metadata field "Seastar Speed (MB/s)".
The correlation is 0.996415213842904 (very high).
SOURCE [{source.f} {2,18}]: "P_WALL_CLOCK_TIME:EXCLUSIVE" metric is inversely correlated with the metadata field "Memory Speed (MB/s)".
The correlation is -0.9980107779462387 (very high).
SOURCE [{source.f} {2,18}]: "P_WALL_CLOCK_TIME:EXCLUSIVE" metric is inversely correlated with the metadata field "Seastar Speed (MB/s)".
The correlation is -0.9959749668655212 (very high).
...done with rules.
---------- JPython test script end ----------

Correlated communication behavior with metadata

Identified hardware differences

Cray-specific inference

NWChem Case Study

- NWChem is a leading chemistry modeling code
- NWChem relies on Global Arrays (GA)
  - Provides a global view of a physically distributed array
  - One-sided access to arbitrary patches of data
  - Developed as a library (fully interoperable with MPI)
- Aggregate Remote Memory Copy Interface (ARMCI)
  - GA communication substrate for one-sided communication
  - Portable high-performance one-sided communication library
  - Rich set of remote memory access primitives
- Would like to better understand the performance of representative workloads for NWChem on different platforms
  - Help to create use cases for one-side programming models

NWChem One-sided Communication and Scaling

- Understand interplay between data-server and compute processes as a function of scaling
  - Data-server uses a separate thread
  - Large numerical computation per node at small scale can obscure the cost of maintaining passive-target progress
  - Larger scale decreases numerical work per node and increases the fragmentation of data, increasing messages
  - Vary #nodes, cores-per-node, and memory buffer pinning
- Understand trade-off of core allocation
  - All to computation versus some to communication

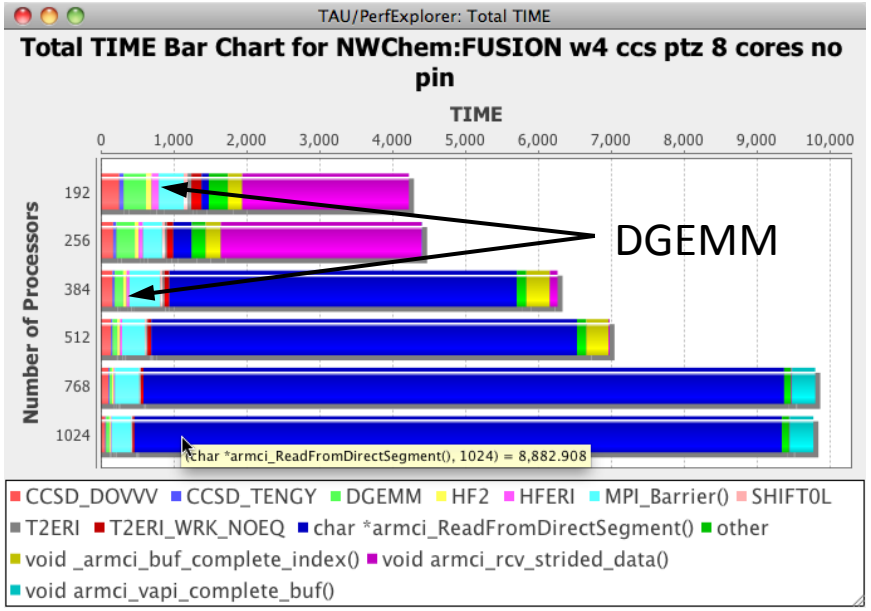
J. Hammond, S. Krishnamoorthy, S. Shende, N. Romero, A. Malony, "Performance Characterization of Global Address Space Applications: a Case Study with NWChem," Concurrency and Computation: Practice and Experience, Vol 24, No. 2, pp. 135-154, 2012.

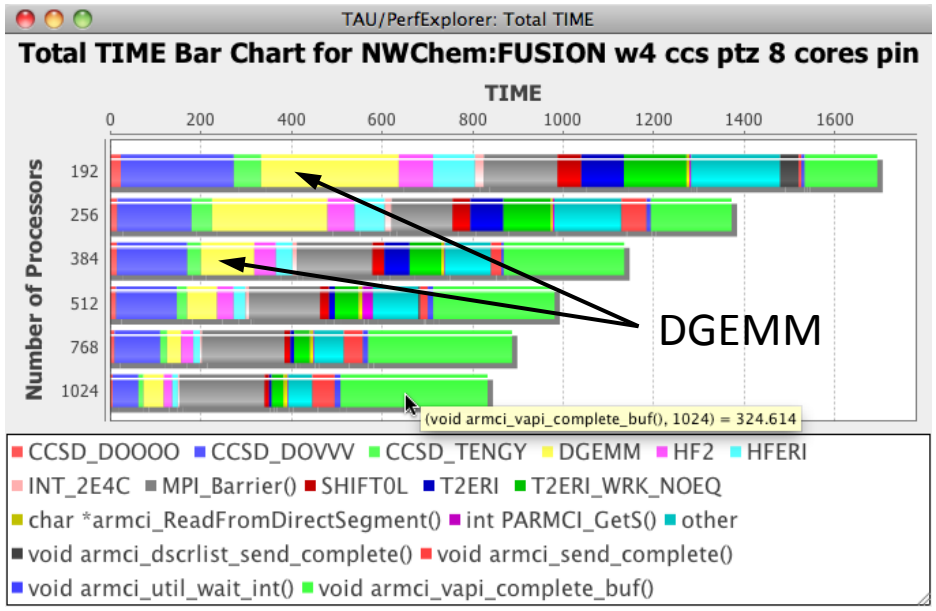
NWChem Instrumentation

- Source-base instrumentation of NWChem application
- Developed an ARMCI interposition library (PARMCI)
  - Defines weak symbols and name-shifted PARMCI interface
  - Similar to PMPI for MPI
- Developed a TAU PARMCI library
  - Intervals events around interface routines
  - Atomic events capture communication size and destination
- Wrapped external libraries
  - BLAS (DGEMM)
- Need portable instrumentation for cross-platform runs
- Systems
  - Fusion: Linux cluster, Pacific Northwest National Lab
  - Intrepid: IBM BG/P, Argonne National Lab

FUSION Tests Comparing No Pinning vs. Pinning

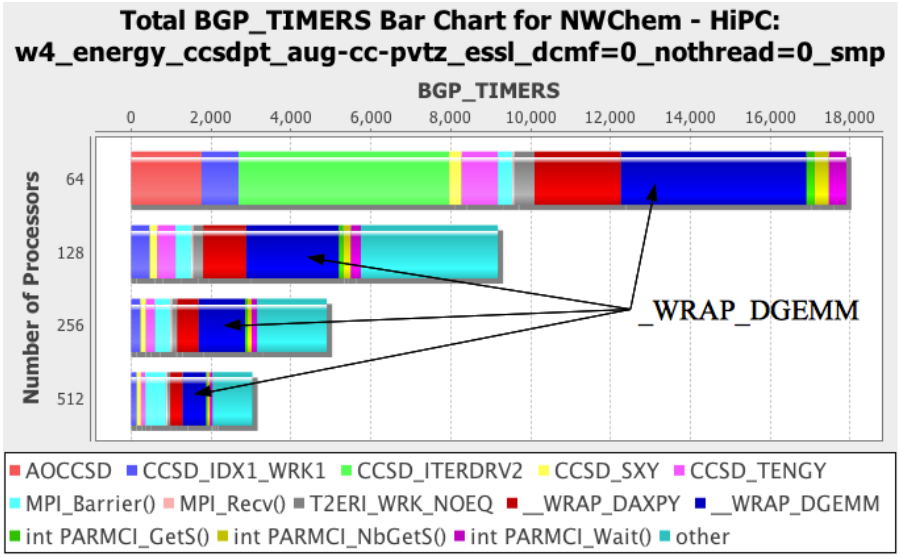
- Scaling on 24, 32, 48, 64, 96 and 128 nodes
- Test on 8 cores (no separate data server thread)
- With no pinning ARMCI communication overhead increases dramatically and no scaling is observed
- Pinning communication buffers shows dramatic effects
- Relative communication overhead increases, not dramatically

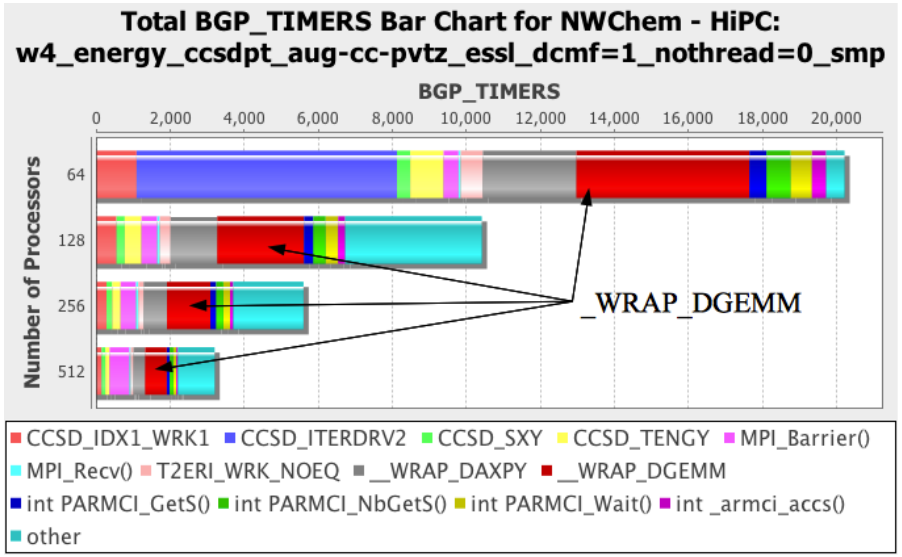




Intrepid Tests Comparing No Pinning vs. Pinning

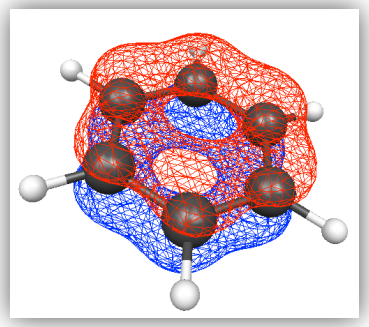
- Scaling on 64, 128, 256 and 512 nodes
- Tests with interrupt or communication helper thread (CHT)
  - CHT requires a core to be allocated
- ARMCI calls are barely noticeable
- DAXPY calculation shows up more
- CHT performs better in both SMP and DUAL modes



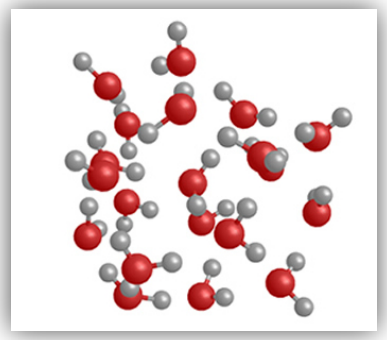


Electronic Structure in Computational Chemistry

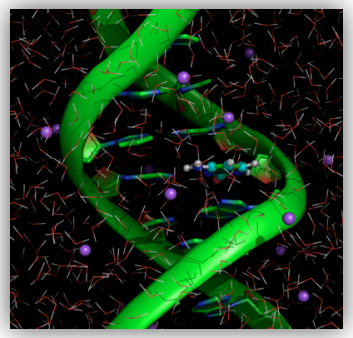
- Parallel performance is determined by:
  - How the application is design and developed
  - The nature and characteristics of the problem



Benzene



Water Clusters

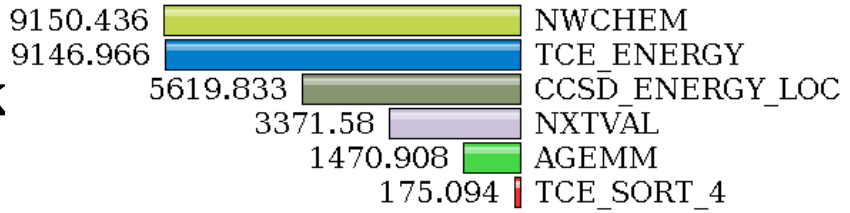


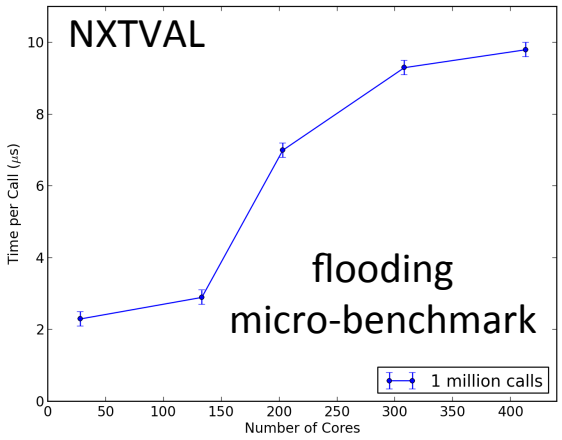
Macro-Molecules

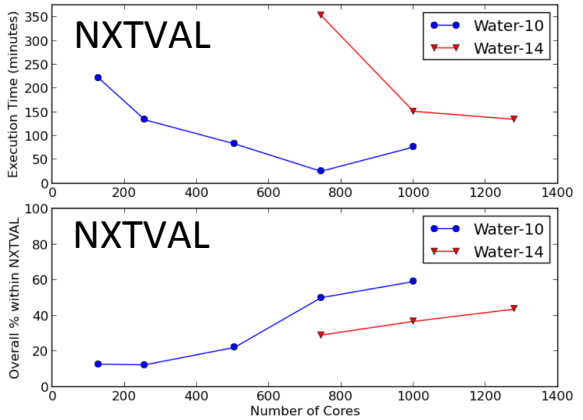
- Computational chemistry applications can exhibit:
  - Highly symmetric diverse load (e.g., Benzene)
  - Asymmetric unpredictable load (e.g., water clusters)
  - QM/MM sheer large size (e.g., macro-molecules)
- Load balance is crucially important for performance

NWChem Performance Analysis – NXTVAL

- Focus on NXTVAL
  - Atomic counter keeping track of which global tasks sent
- Strong scaling experiment
  - 14 water molecules
    - aug-cc-PVDZ dataset
  - 124 nodes on ANL Fusion
    - 8 cores per node
  - NXTVAL significant %
  - Increasing per call time
- When arrival rate exceeds processing rate, buffering and flow control must be used

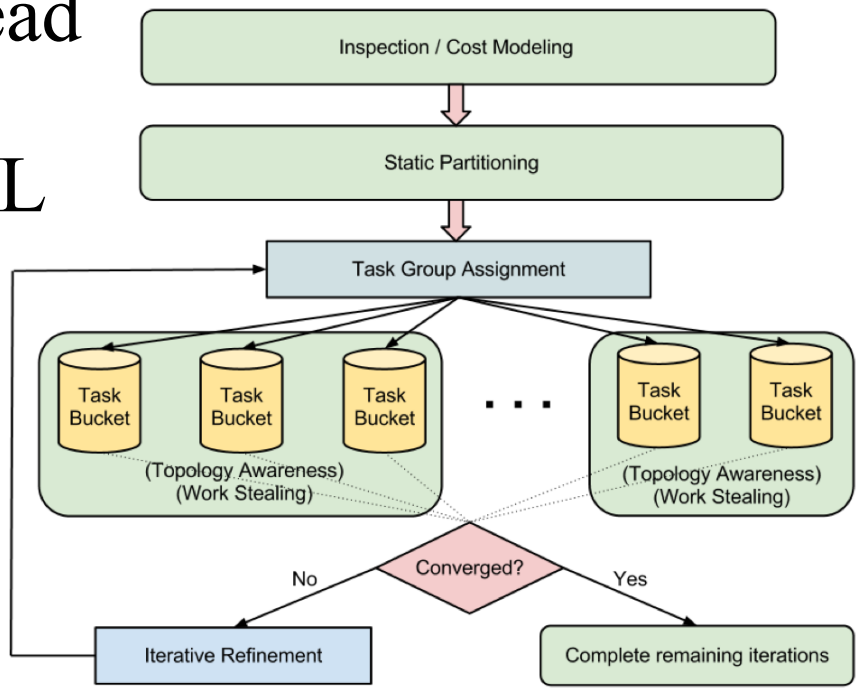


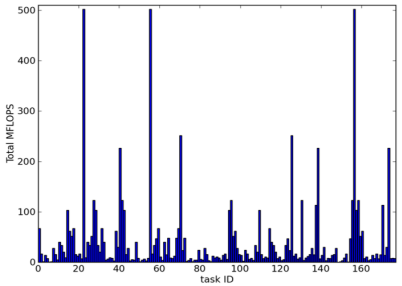




Evaluation of Inspector-Executor Algorithm

- How to eliminate the overhead of centralized load balance algorithm based on NXTVAL
- Use an inspector-executor approach to assigning tasks
  - Assess task imbalance
  - Reassign



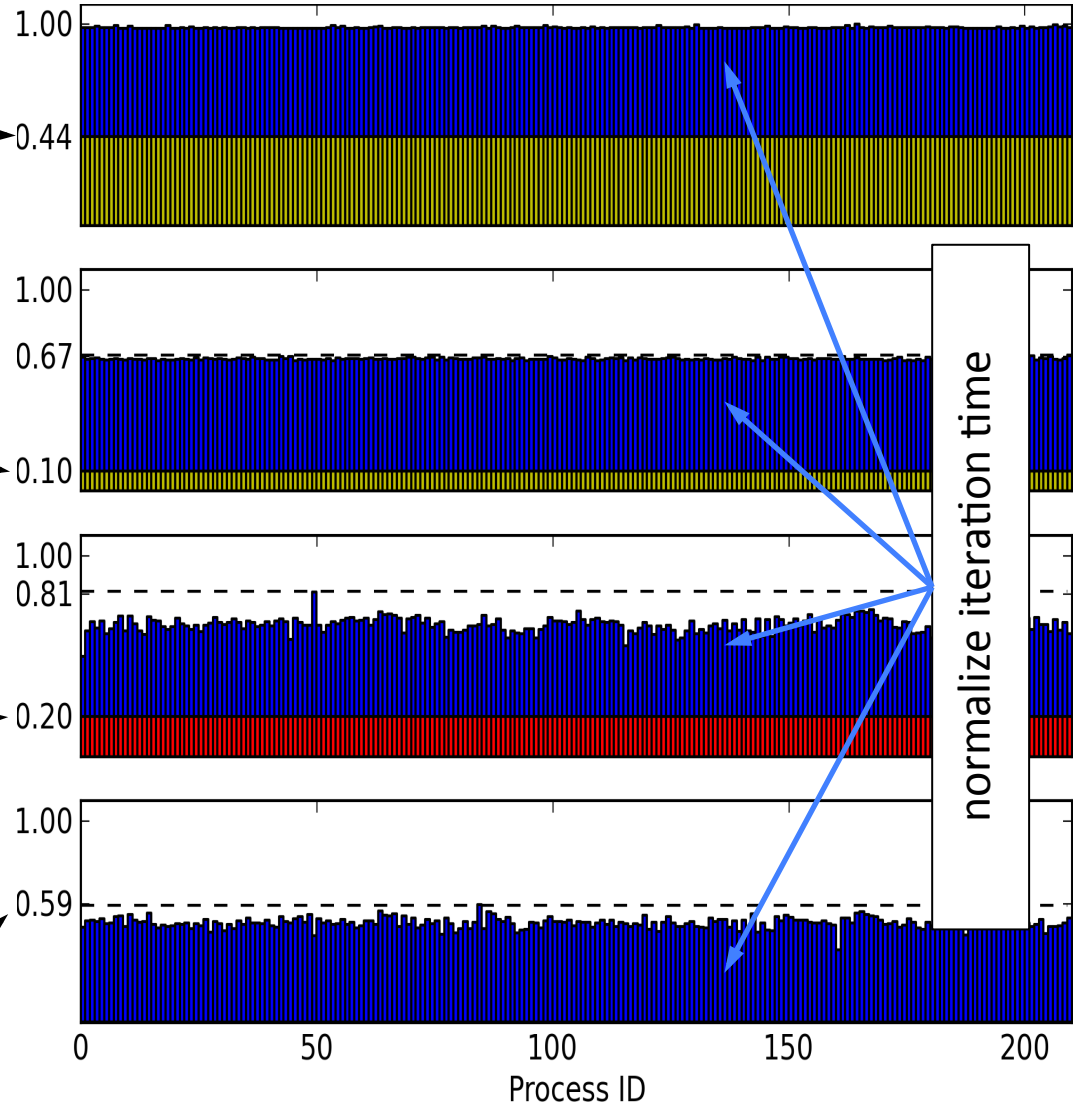


- Use TAU to evaluate performance improvement with respect to NXTVAL, overhead, task balance

D. Ozog, J. Hammond, J. Dinan, P. Balaji, S. Shende, A. Malony, "Inspector-Executor Load Balancing Algorithms for Block-Sparse Tensor Contractions," International Conference on Parallel Processing, September 2013.

Refinement from NXTVAL to Inspector/Executor

- Original NXTVAL measured
- Original NXTVAL reduced
- Inspector/Executor 1st iteration overhead
- Inspector/Executor subsequent iterations



MPAS (Model for Prediction Across Scales)

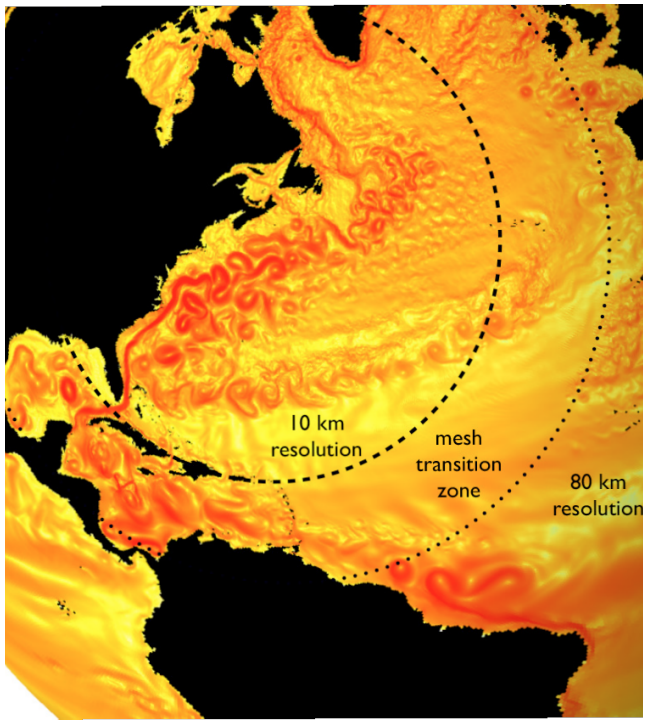
- MPAS is an unstructured grid approach to climate system modeling
  - Explore regional-scale climate change

http://mpas-dev.github.io

- MPAS supports both quasi-uniform and variable resolution meshing of the sphere
  - Quadrilaterals, triangles, or Voronoi tessellations
- MPAS is a software framework for the rapid prototyping of single components of climate system models
  - Two SciDAC earth systems codes (dynamical cores)
    - MPAS-O (ocean model)
    - CAM-SE (atmosphere model)

MPAS-Ocean (MPAS-O) Overview

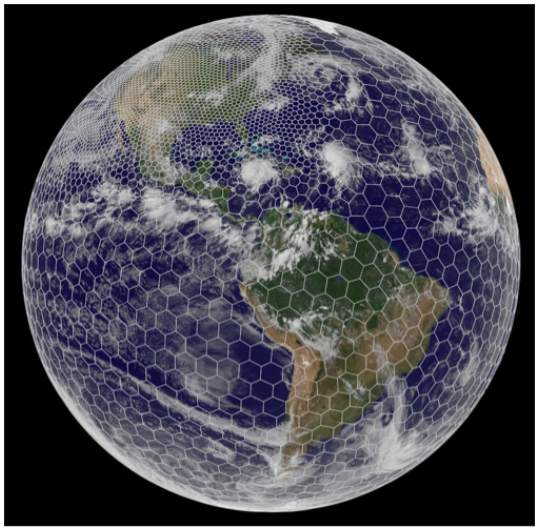
- MPAS-O is designed for the simulation of the ocean system from time scales of months to millenia and spatial scales from sub 1 km to global circulations
- MPAS-O has demonstrated the ability to accurately reproduce mesoscale ocean activity with a local mesh refinement strategy

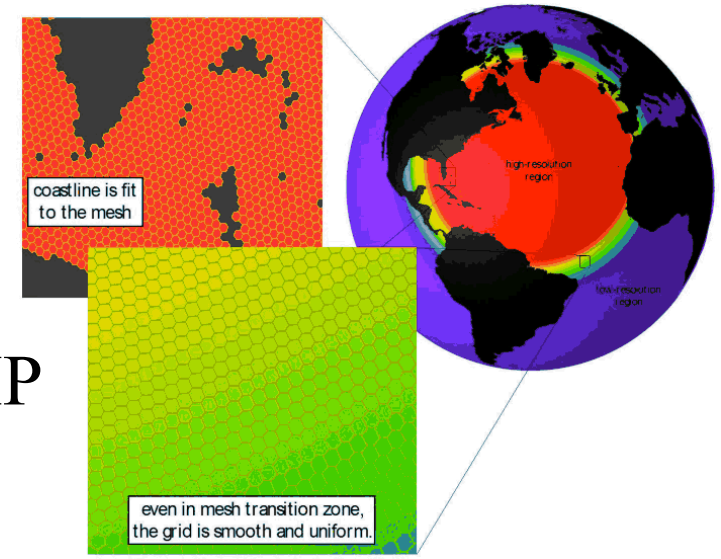


- In addition to facilitating the study of multiscale phenomena within ocean systems, MPAS-O is intended for the study of anthropogenic climate change as the ocean component of climate system models

Multiscale and MPAS-O Domain Decomposition

- Use multiscale methods for accurate, efficient, and scale-aware models of the earth system
- MPAS-O uses a variable resolution irregular mesh of hexagonal grid cells
- Cells assigned to MPI processes, grouped as "blocks"
  - Each cell has 1-40 vertical layers, depending on ocean depth
- MPAS-O has demonstrated scaling limits when using MPI alone
- Look at increasing concurrency
- Developers currently adding OpenMP
  - Both split explicit and RK4 solvers



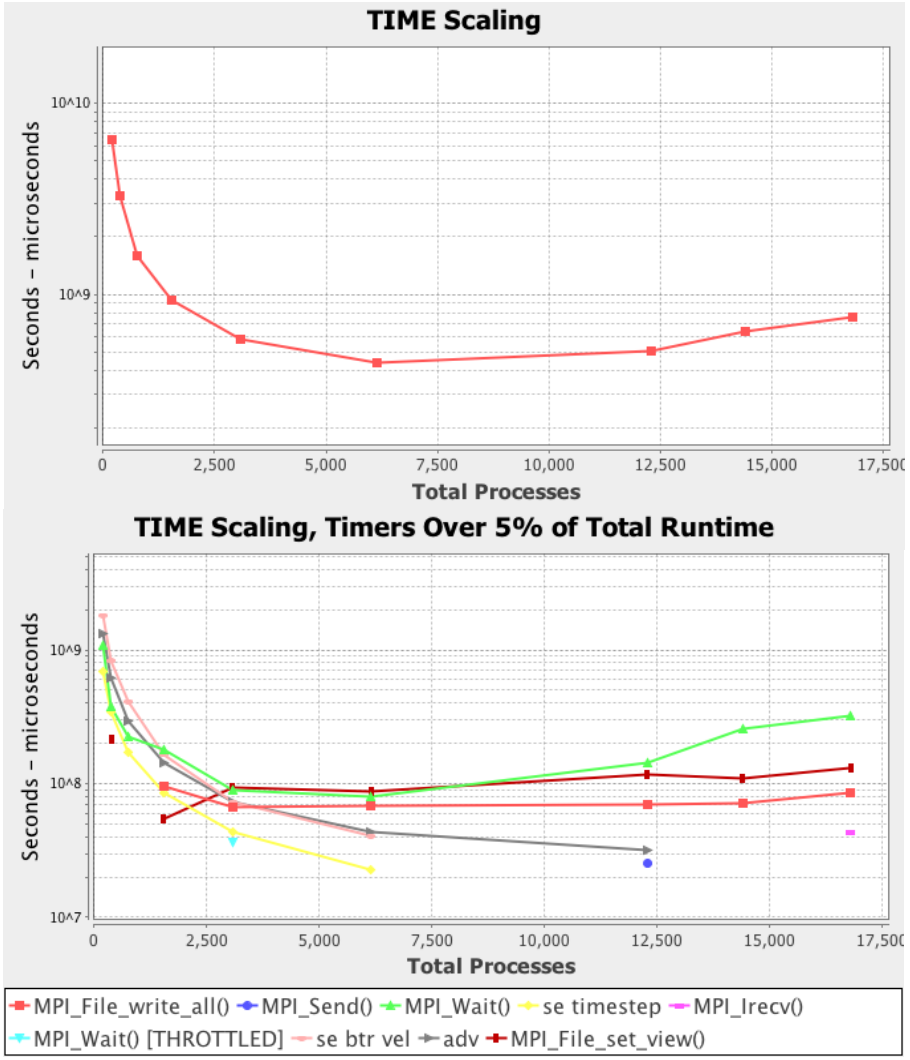


MPAS-Ocean Performance Study

- Integrate TAU into the MPAS build system
  - Evaluate the original MPI-only approach
  - Study the the new MPI+OpenMP approach
- Performance results
  - MPI block + OpenMP element decomposition
    - reduces total instructions in computational regions
    - ~10% faster than MPI alone
  - Guided OpenMP thread schedule balances work across threads
    - ~6% faster than default
  - Weighted block decomposition using vertical elements (depth) could balancing work across processes (~5% faster in some tests)
  - Overlapping communication and computation could reduce synchronization delays when exchanging halo regions (underway)
- Evaluation is ongoing and includes porting to MIC platform

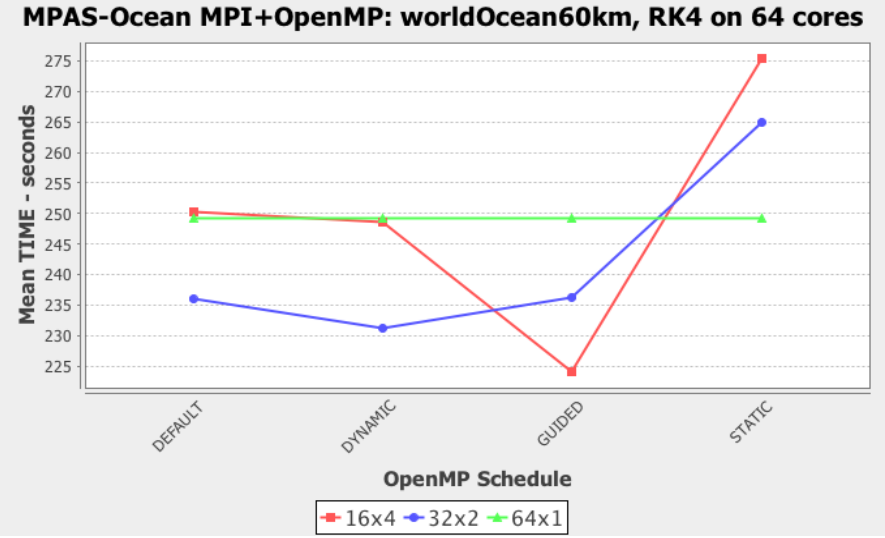
MPI Scaling Study (Hopper)

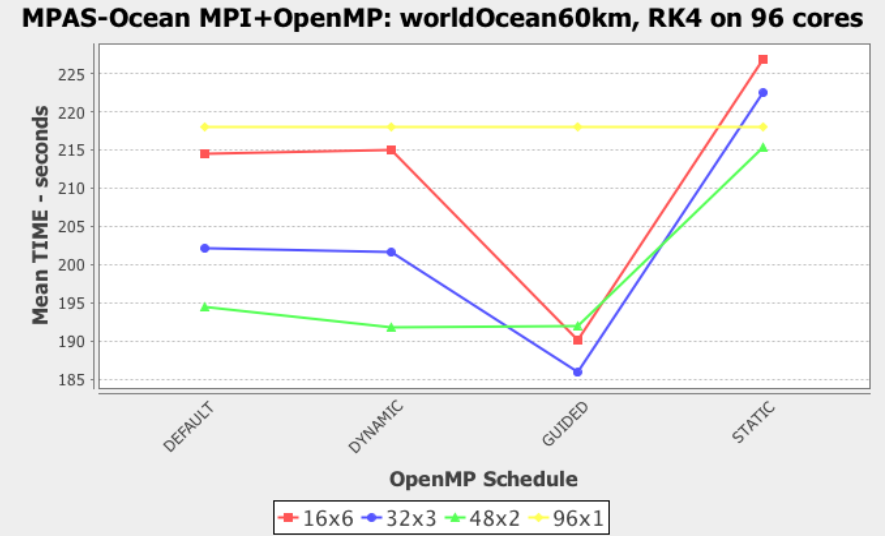
- Strong scaling
  - 192 to 16800
- Poor scaling over 6144 processes
  - Communication begins to dominate
  - Problem size might be too small
- Time profile by events
  - Only MPI events > 5% after 12K processes



Benefits of RK4 solver and OpenMP Scheduling

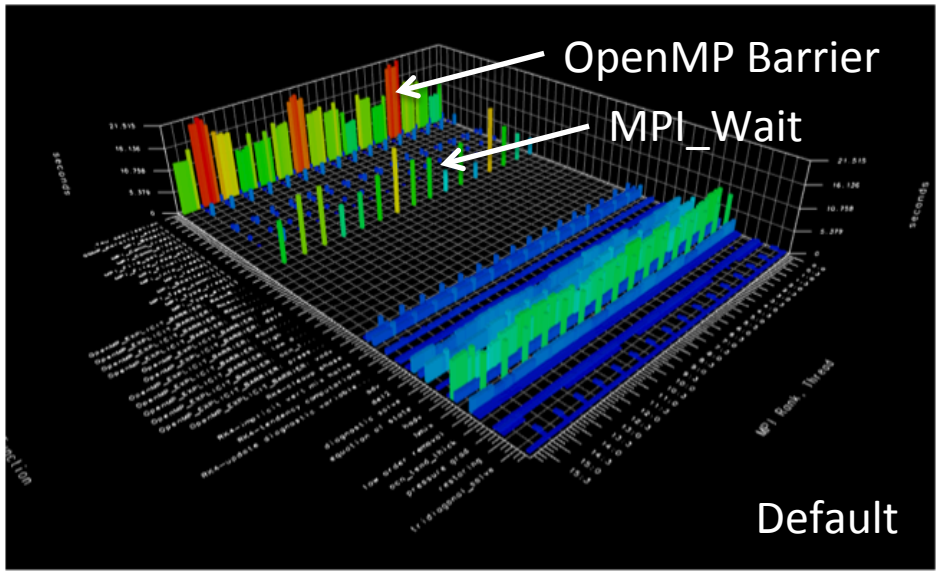
- Use modified RK4 solver versus original MPI solver
- Test cases
  - 64 cores
    - 64 processes (MPI-only)
    - 32 processes x 2 threads
    - 16 processes x 4 threads
  - 96 cores
    - 96 processes (MPI-only)
    - 48 processes x 2 threads
    - 32 processes x 3 threads
    - 16 processes x 6 threads
- OpenMP scheduling options evaluated

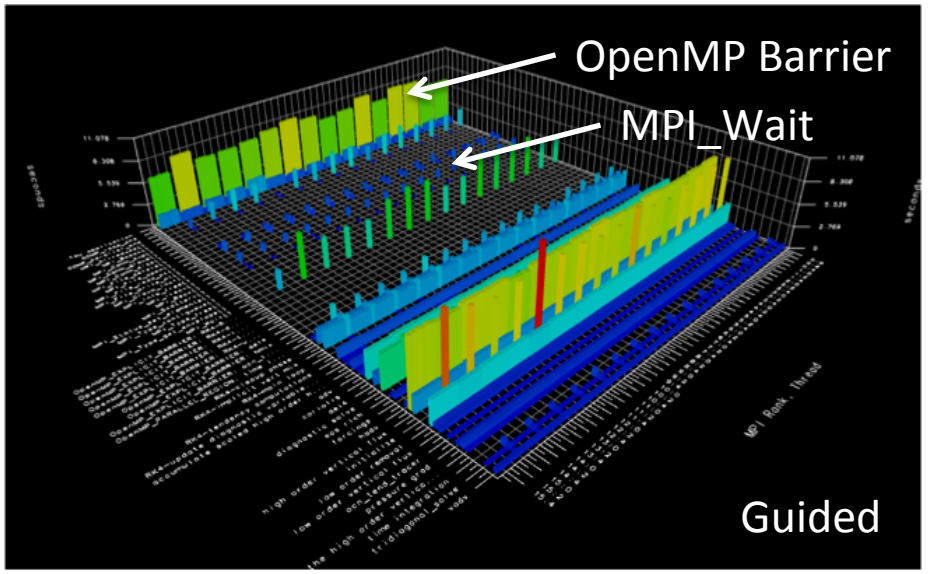




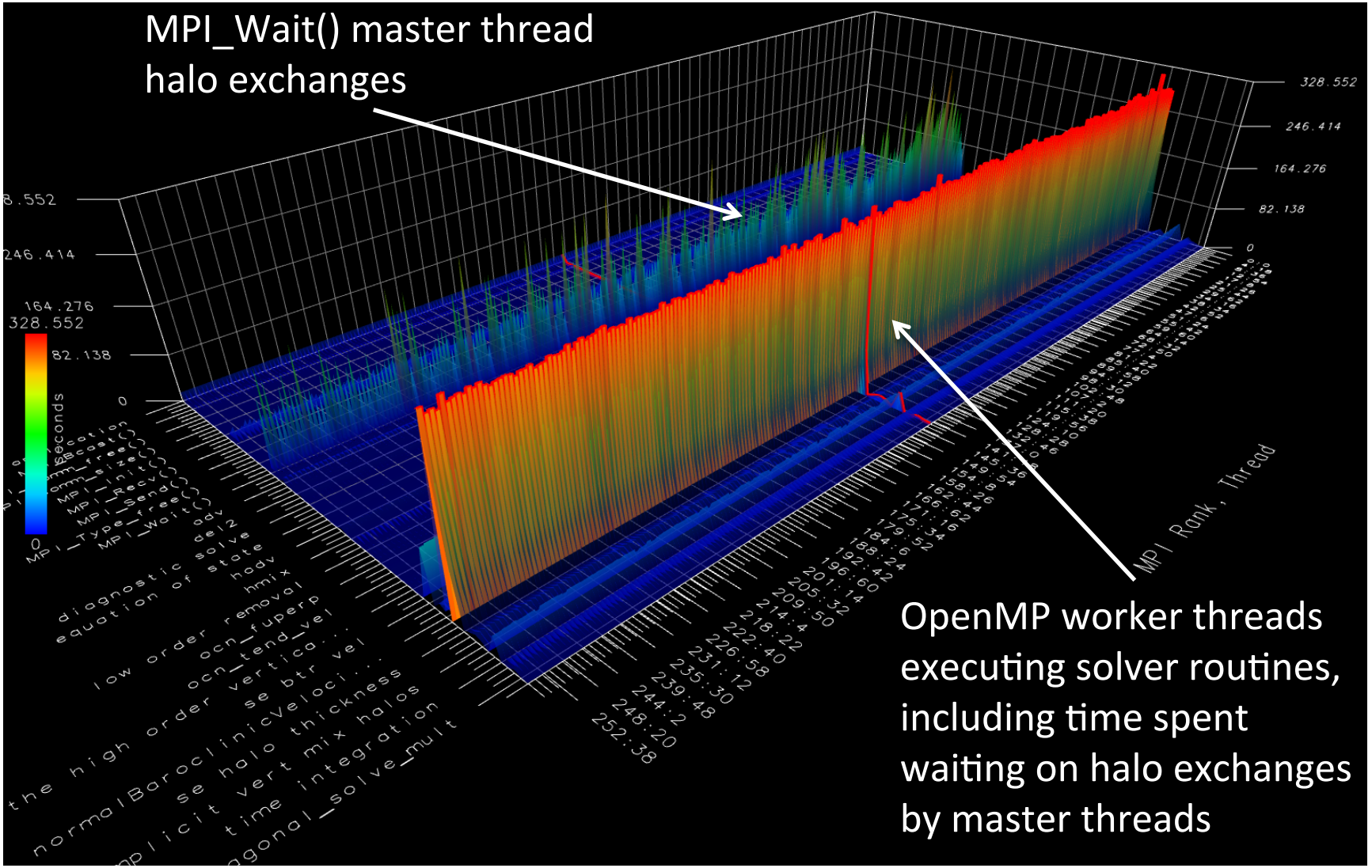
Benefits of Guided Scheduling with MPAS-O RK4

- MPI + OpenMP experiment
  - 96 cores (16 processes by 6 threads per process)
- Default scheduling generates a load imbalance
  - Threads arrive at boundary cell exchange at different times
  - Complete boundary cell exchange at different rates
    - only the master thread performs the exchange
- Guided scheduling provides better balance



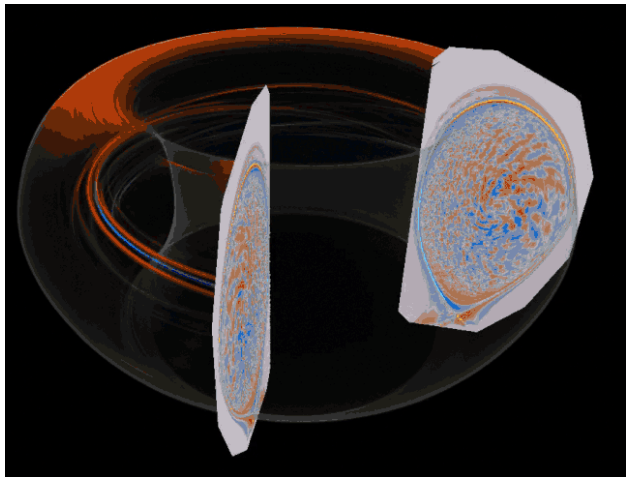


MPAS-O on BG/Q (Vesta) on 16K (256x64)



XGC for Studying Critical Edge Physics

- XGC is a set of codes to modelgyrokinetic microturbulence in the simulation of fusion reactors
  - DOE SciDAC for Edge Physics Simulation (EPSi)
  - Particle-in-cell (PIC) codes:
    - XGC0, XGC1, XGC1a, XGC2, …
- XGC1
  - ODE-based PIC approach on space grid using unstructured triangular grid
  - 5D gyrokinetic equations
    - ODE: time advance of marker particles
    - finite difference: partial integro-differential Fokker-Planck collision operator discretized on rectangular v-space grid
    - PDE (PETSc): Maxwell's equations on unstructured triangular x-space grid
  - Usual interpolation issue exists



XGC1 Development and Performance

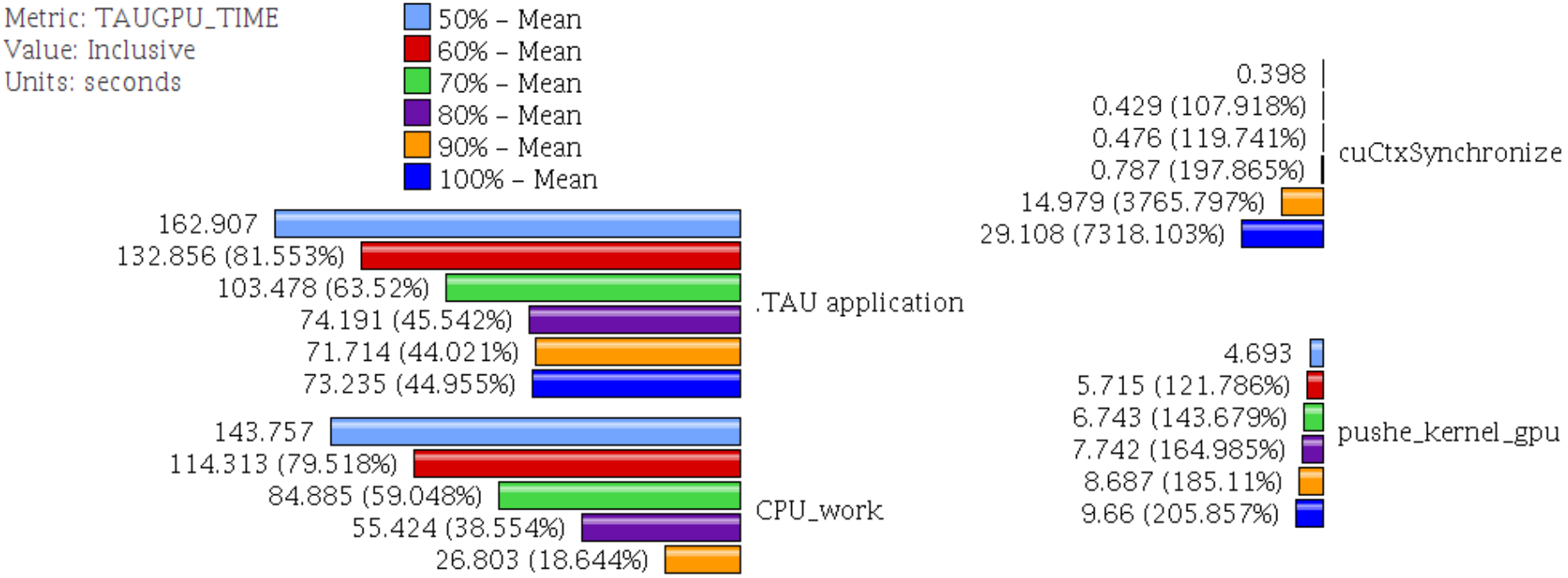
- Development of electromagnetic turbulence capability
  - Requires more accurate calculation of electrical current
  - More particles requires greater scaling
- Challenge to run on heterogeneous systems (Titan)
  - Electron subcycling time-advance takes ~85% of total time
    - designed to be without external communication in each cycle
    - ideal for occupying GPUs while other routines use CPUs
    - solver spend <5% of total computing time
  - Weak scaling in number of particles
    - XGC1 has not reach the MPI communication limit
    - #particles per grid-node is fixed
    - #grid-nodes memory determined by compute-node memory
    - need more compute nodes

XGC1 Performance Study (Titan, ACISS)

- EPSi with DOE SUPER institute is optimizing XGC1
  - Computational kernel including port to GPU
  - OpenMP parallelism
  - MPI communication
- Use TAU to investigate XGC1 performance questions:
  - How to efficiently split work between the CPU and GPU?
  - How to improve the cache performance in the GPU?
  - Effects of memory copying with multiple GPUs per node
- Experiments done on ACISS and Titan
  - ACISS: 12-core 2.67GHz Xeon X5650 / Tesla M2070 (3x)
  - Titan: 16-core 2.2GHz Opteron 6274 / Tesla K20X

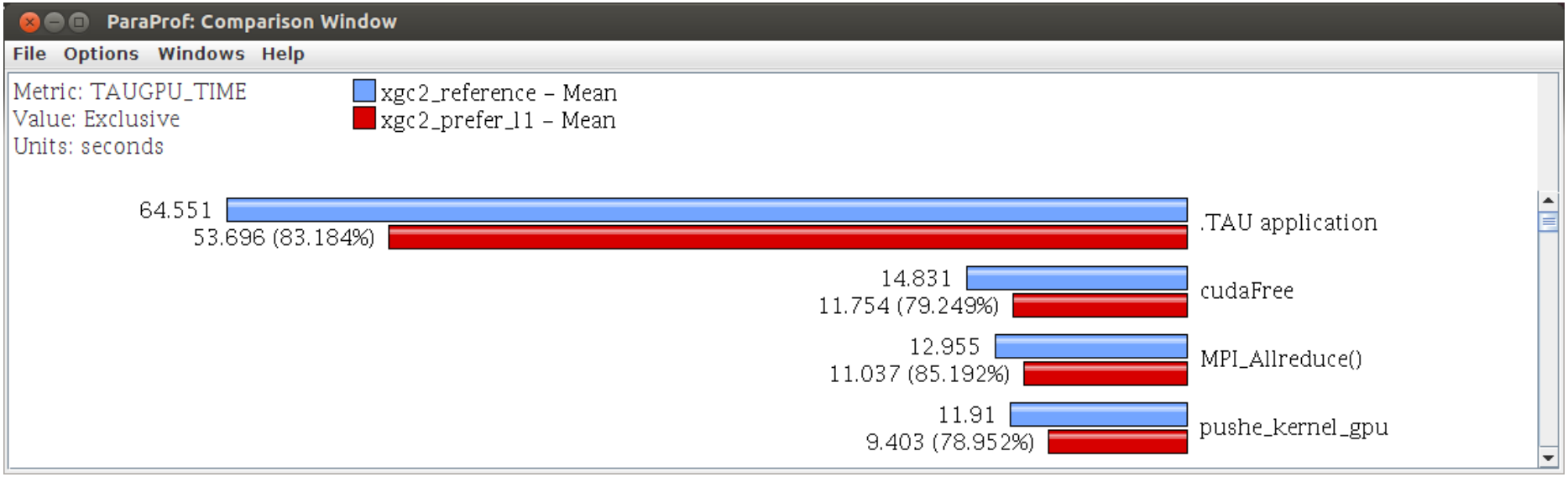
XGC Work Sharing Between CPU/GPU (ACISS)

- Look at CPU+GPU worksharing execution scenario
- Electron 'push' takes 85% of the computation time
  - Kernel (*pushe2*) was ported to the GPU
- Assigning more work to the GPU can shorten the execution since it can compute the kernel faster



XGC Cache Optimization (ACISS)

- Shared memory capacity
  - L1 cache and device shared memory share physical storage
  - Default: 48KB shared memory / 16KB L1 cache
  - CUDA allows you to swap this allocation
  - Improved the performance of the pushe2 kernel (~20%)
  - Improved the performance of XGC application (~17%)

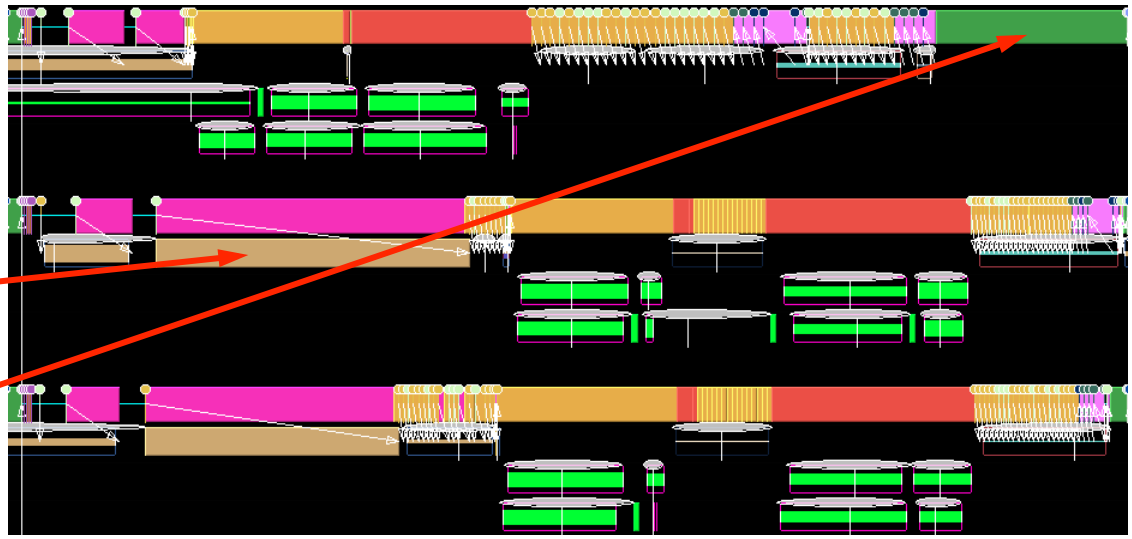


Memory Copies with Multiple GPUs in XGC

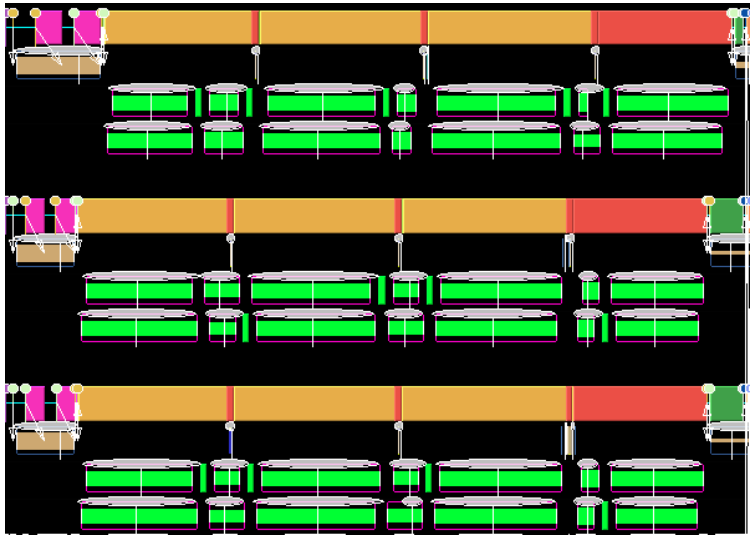
- Given multiple GPUs per node in ACISS
  - XGC could be run with multiple MPI ranks per node
  - Each rank would be assigned 1 GPU
- Performace effect of CPU-GPU memory copies
  - Compare
    - 1 MPI rank per node with 1 GPU
    - 3 MPI ranks per node with 3 GPUs
  - Slow memory copies result with multiple ranks+GPUs
- Slow memory copies can lead to MPI waiting
  - Leads to a slow down in MPI collectives

XGC Performance with Different # GPUs (ACISS)

- 3 MPI ranks
  - 3 GPUs per node (6 seconds)
  - Slow memory copies …
  - … cause faster MPI ranks to stall at MPI_Reduce

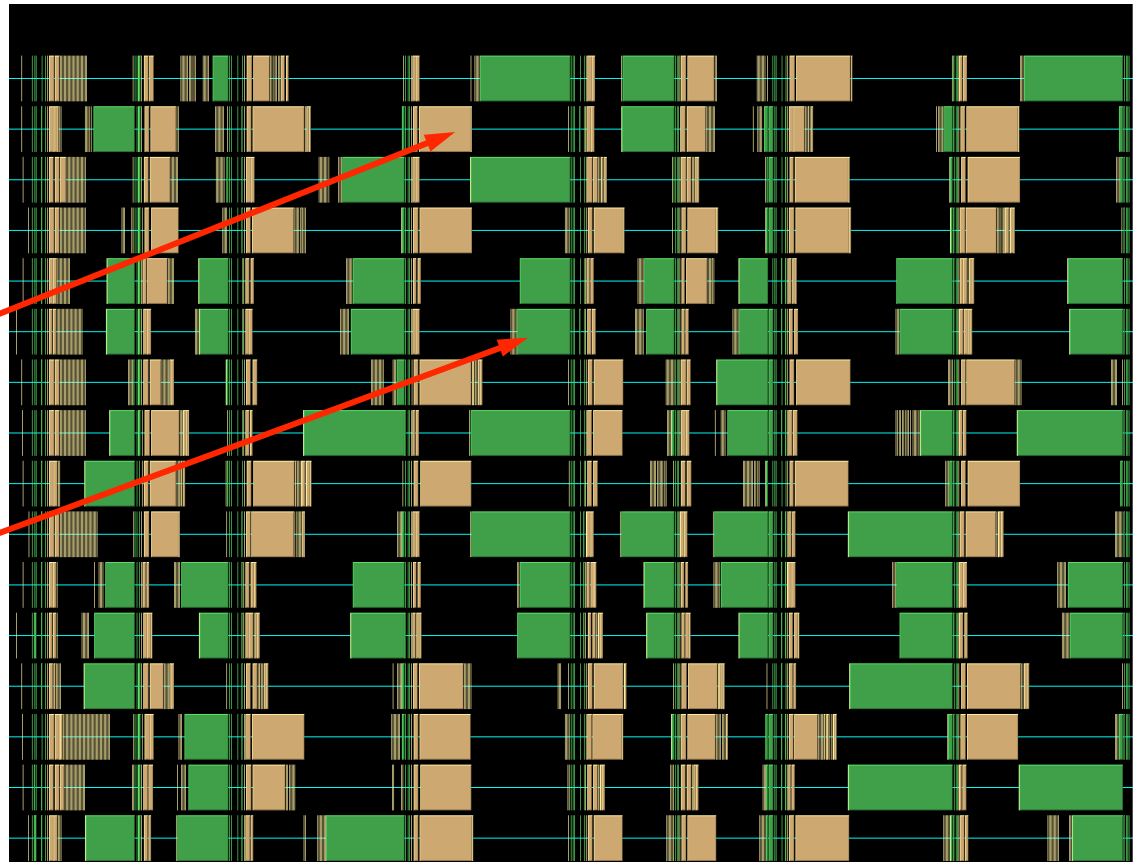


- 3 MPI ranks (different nodes)
  - 1 GPU per node (4 seconds)
  - Fast memory copies better aligns processes and reduces MPI stalls



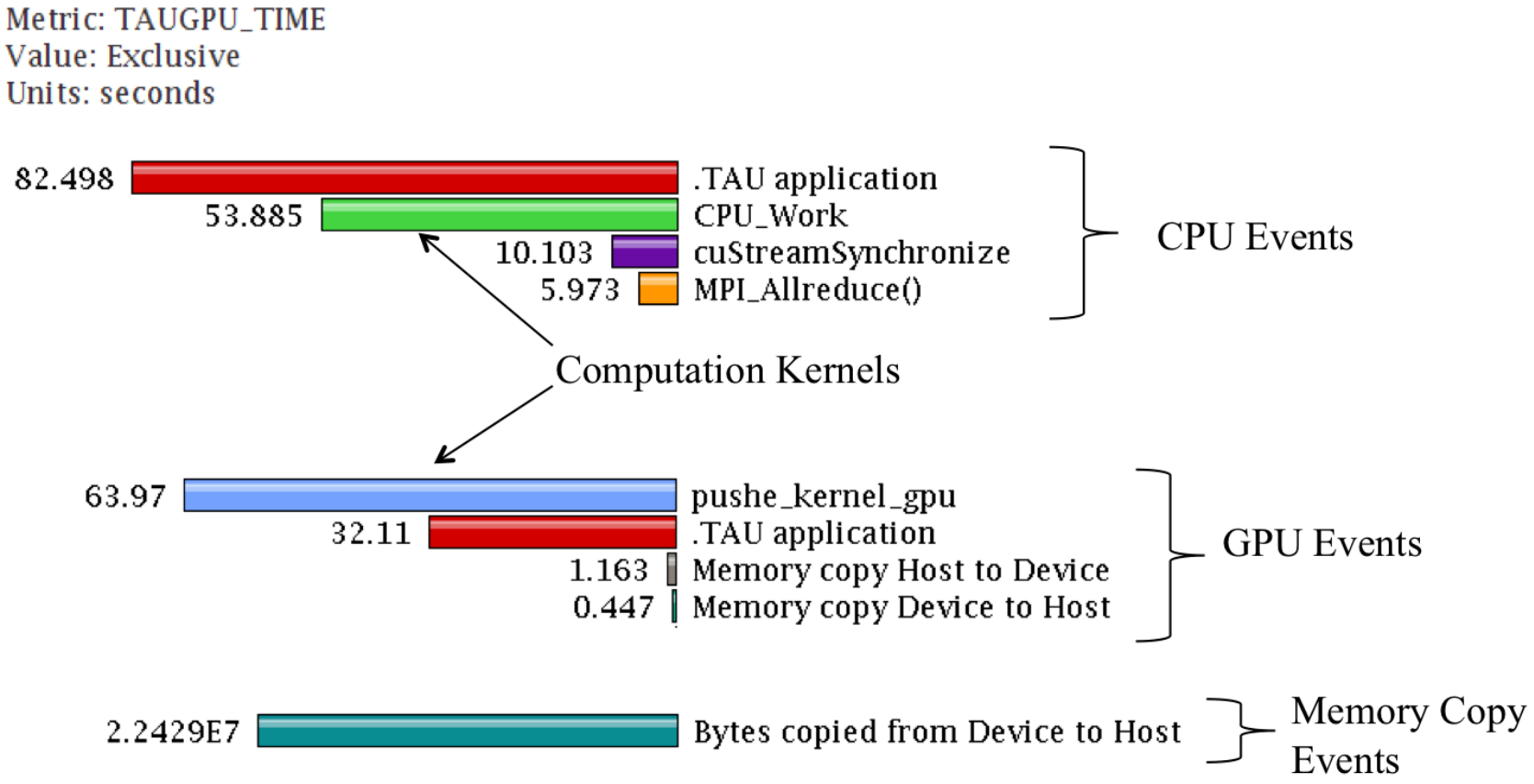
XGC MPI Imbalance Due to PCI Bus Sharing

- 16 ACISS nodes
  - 3 GPUs per node (6 seconds)
- Slow memory copies …
- … cause faster MPI ranks to stall at MPI_Reduce



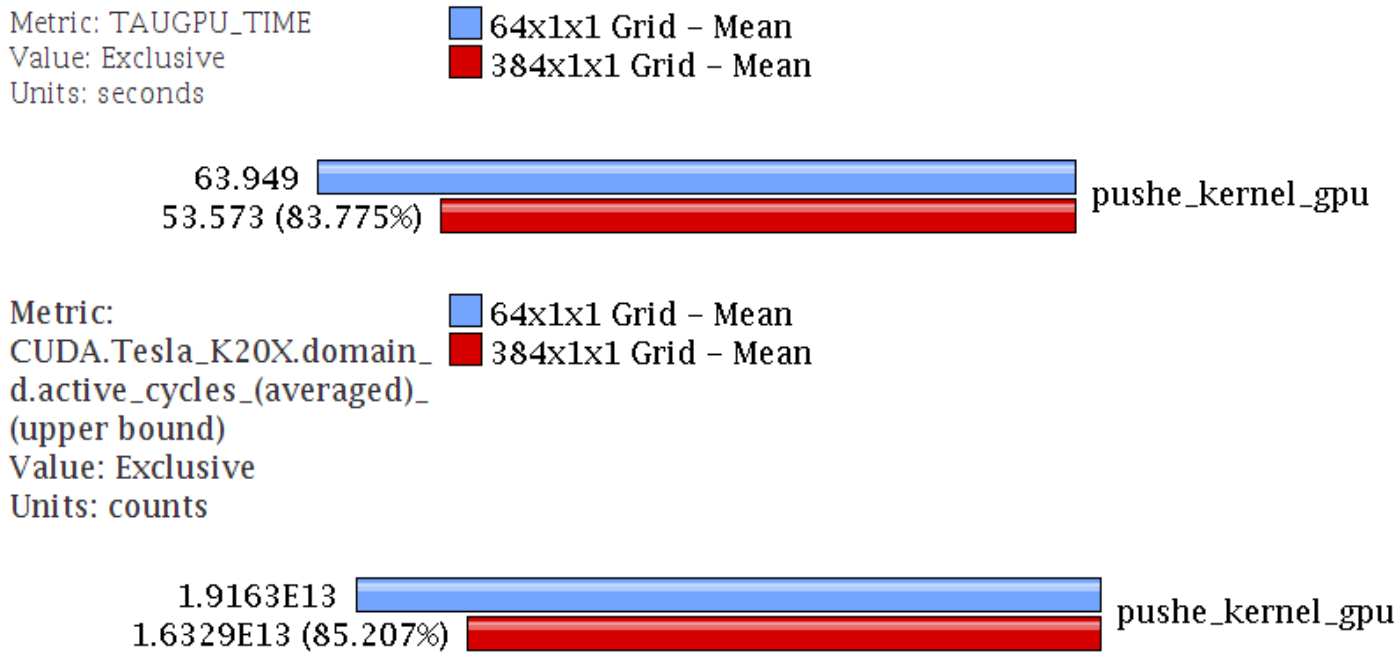
Heterogeneous Performance Measurement (Titan)

- CPU+GPU worksharing execution scenario



XGC Increasing Grid Size (Titan)

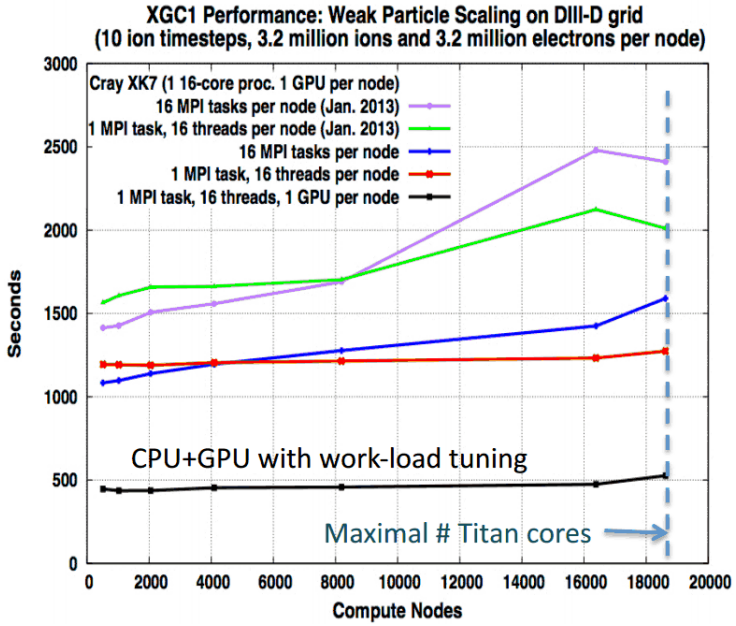
- Allocating a larger grid to improve GPU efficiency

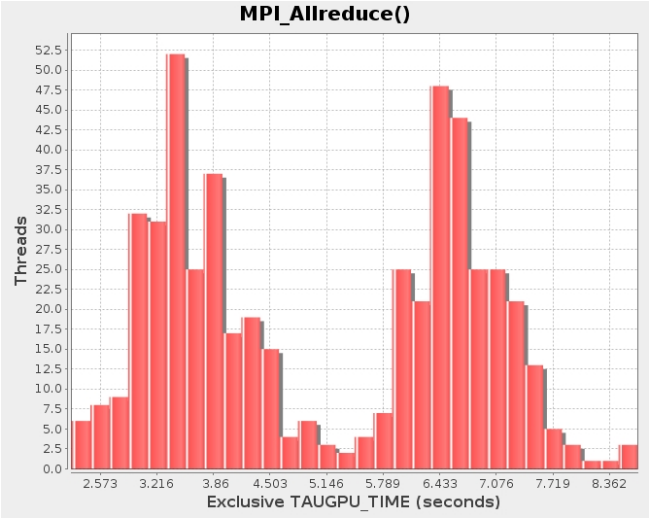


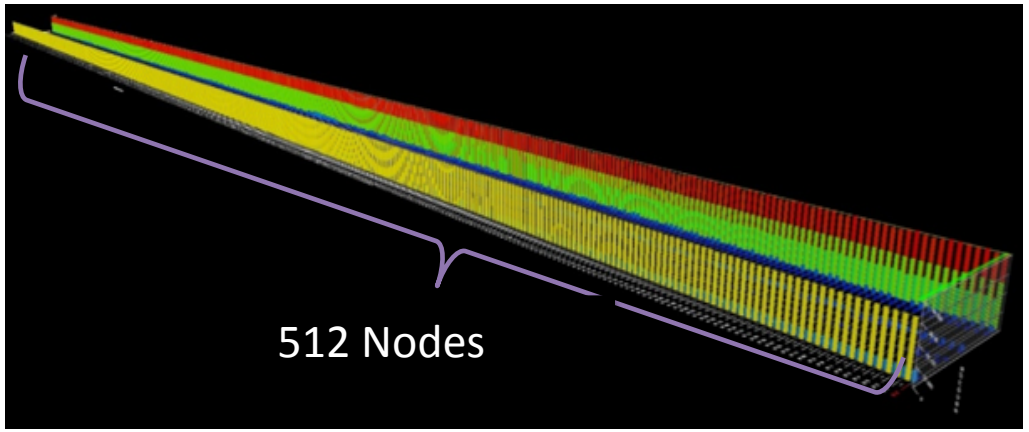
- Larger grid size increases the number of active cycles by 15% and gives a 5% increase in total SM efficiency

XGC1 Performance Scaling (Titan)

- EPSi-SUPER collaboration
- Weak scaling on DIII-D grid
  - 10 ion timesteps
  - 3.2 million ions
  - 3.2 million electrons per node
- Ran 512 node experiment

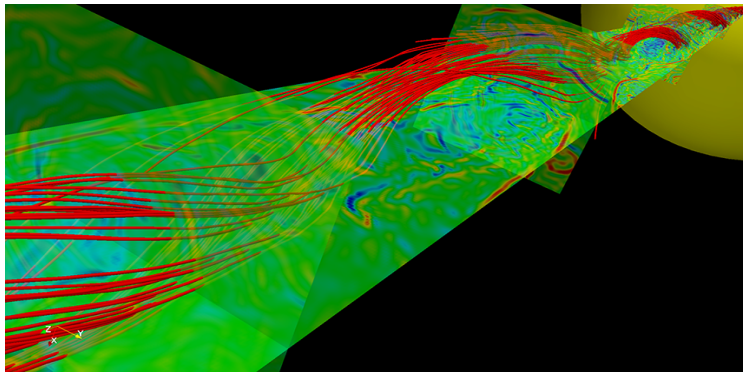




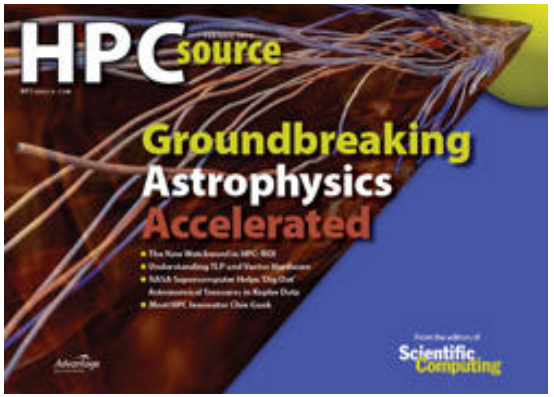


IRMHD Performance (Intrepid and Mira)

- INCITE magnetohydrodynamcis simulation to understand solar winds and coronal heating
  - First direct numerical simulations of Alfvén wave (AW) turbulence in extended solar atmosphere accounting for inhomogeneities



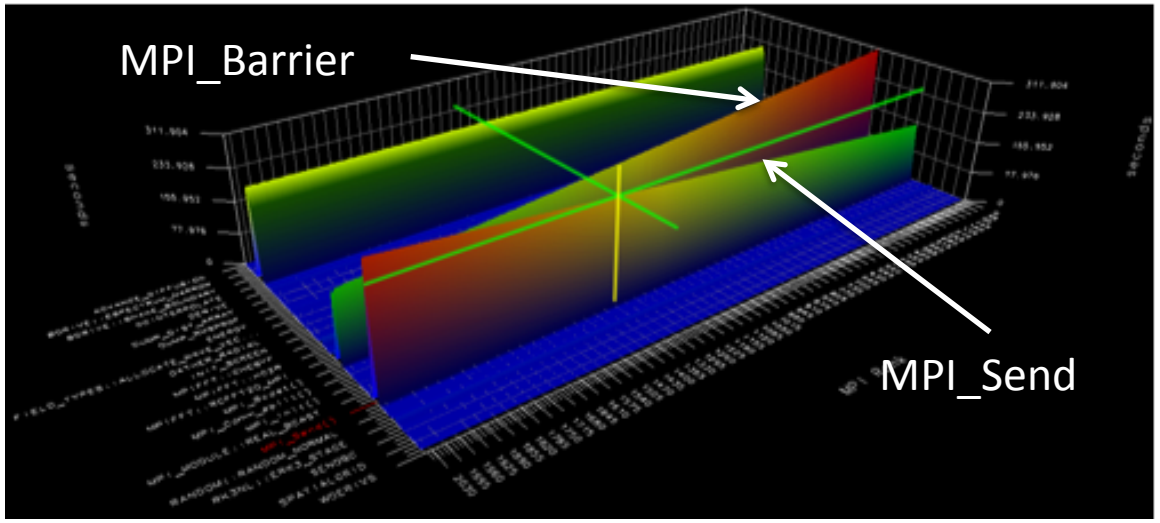
- IRMHD
  - Inhomogeneous Reduced Magnetohydrodynamics
  - Fortran 90 and MPI
  - Excellent scaling properties
  - Tested on Argonne BG/P (Intrepid)
  - Benchmarked on Argonne BG/Q (Mira)
- HPC Source article and ALCF news



https://www.alcf.anl.gov/articles/furthering-understanding-coronal-heating-and-solar-wind-origin

Communication Analysis (MPI_Send, MPI_Bcast)

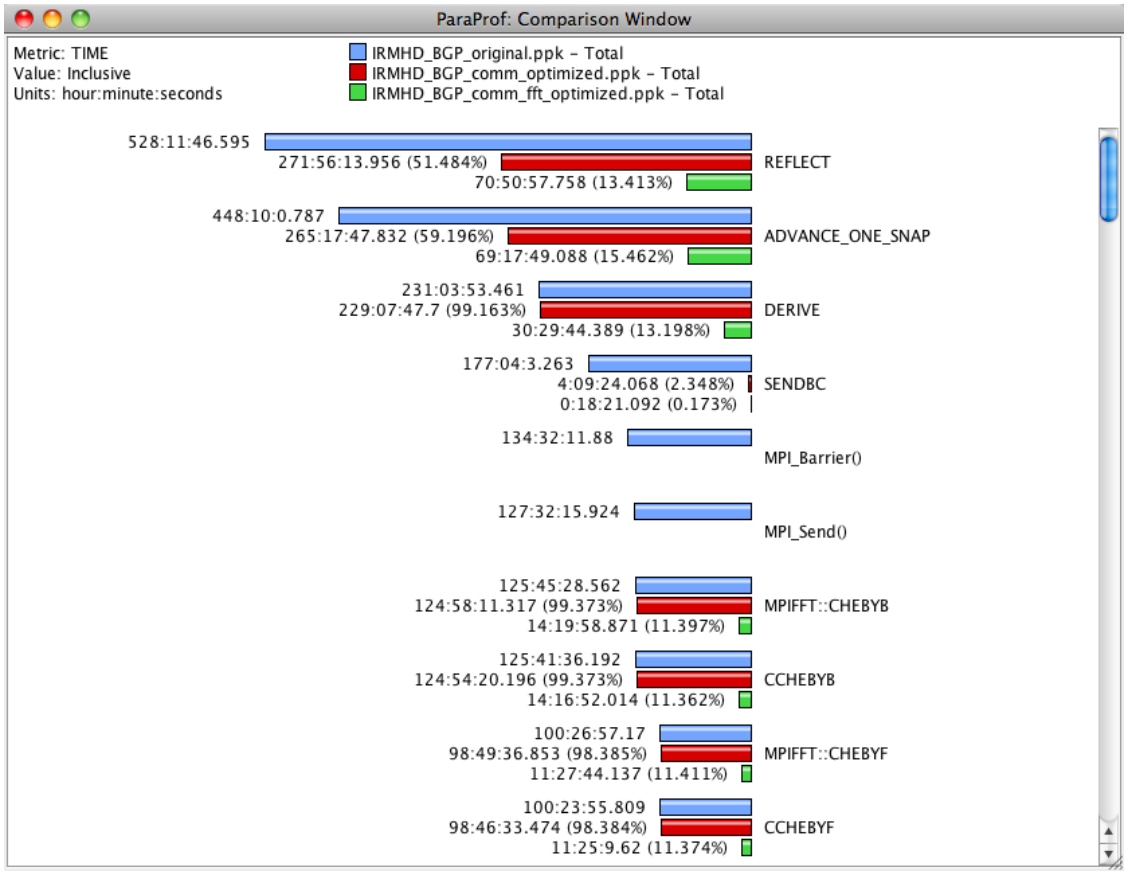
- IRMHD demonstrated performance consistent with common synchronous communication bottlenecks
  - Significant time spent in MPI routines
- Identify problems on 2,048-core execution on Intrepid
  - MPI_Send and MPI_Bcast took significant time
  - Suggest possible opportunities for overlapping computation and communication



  - Identified possible targets for computation improvements

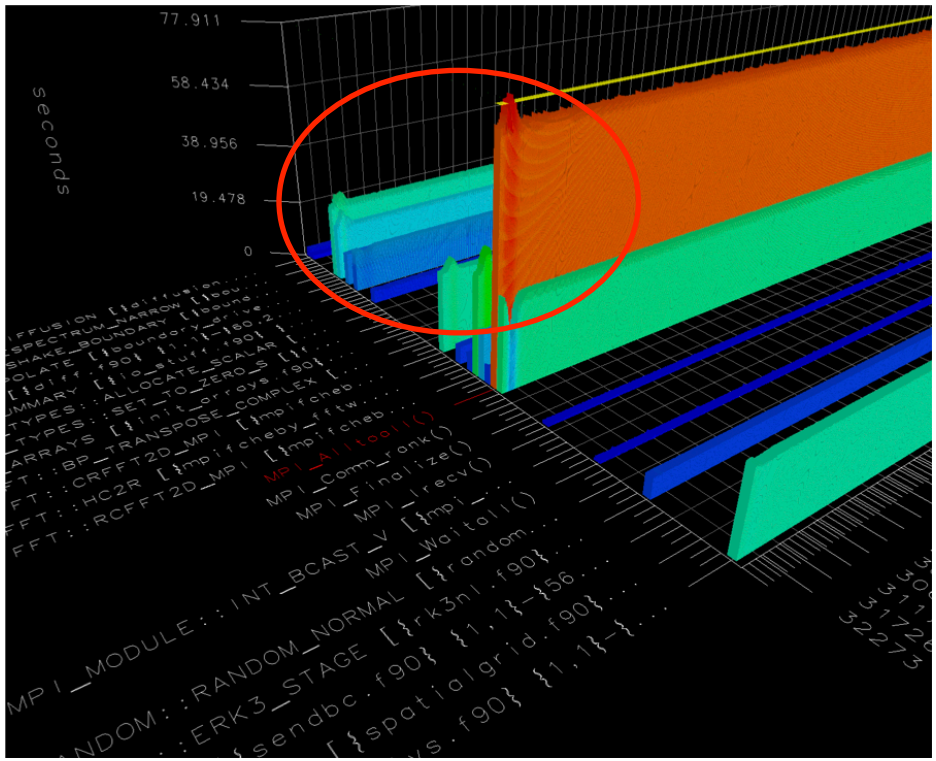
Effects of IRMHD Optimizations (Intrepid)

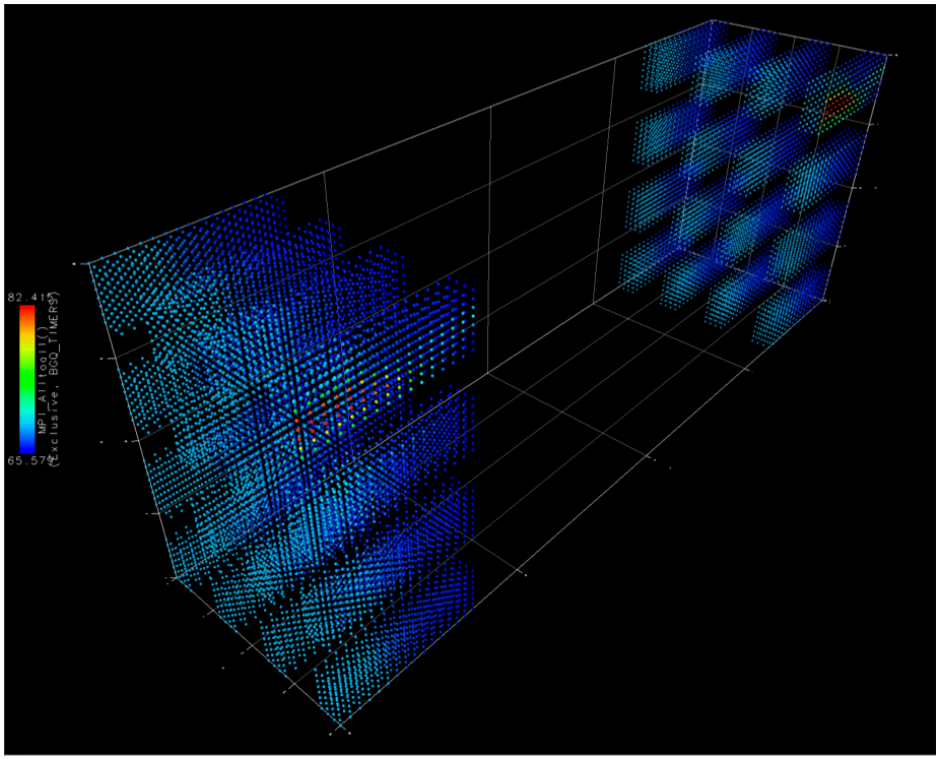
- Developed a non-blocking communication substrate
- Deployed a more efficient FFT implementation
- Overall execution time reduced from 528.18 to 79.85 core hours
  - 2,048 processors
  - >7x improvement



Performance Testing on Mira (BG/Q)

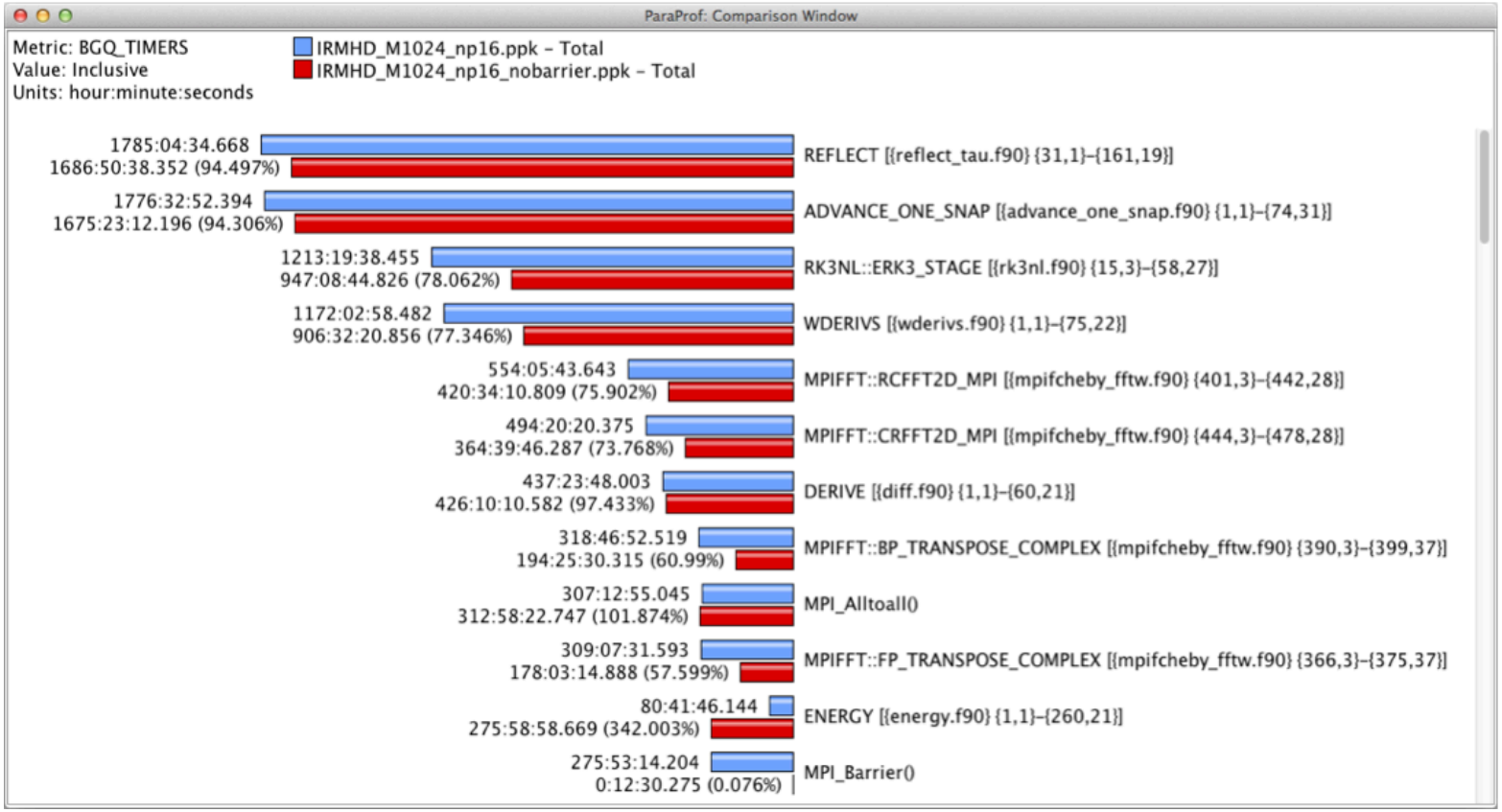
- Test with 32K MPI ranks
- Load imbalance apparent
  - See imbalance reflected in *MPI_Alltoall()* performance





Optimizations on Mira

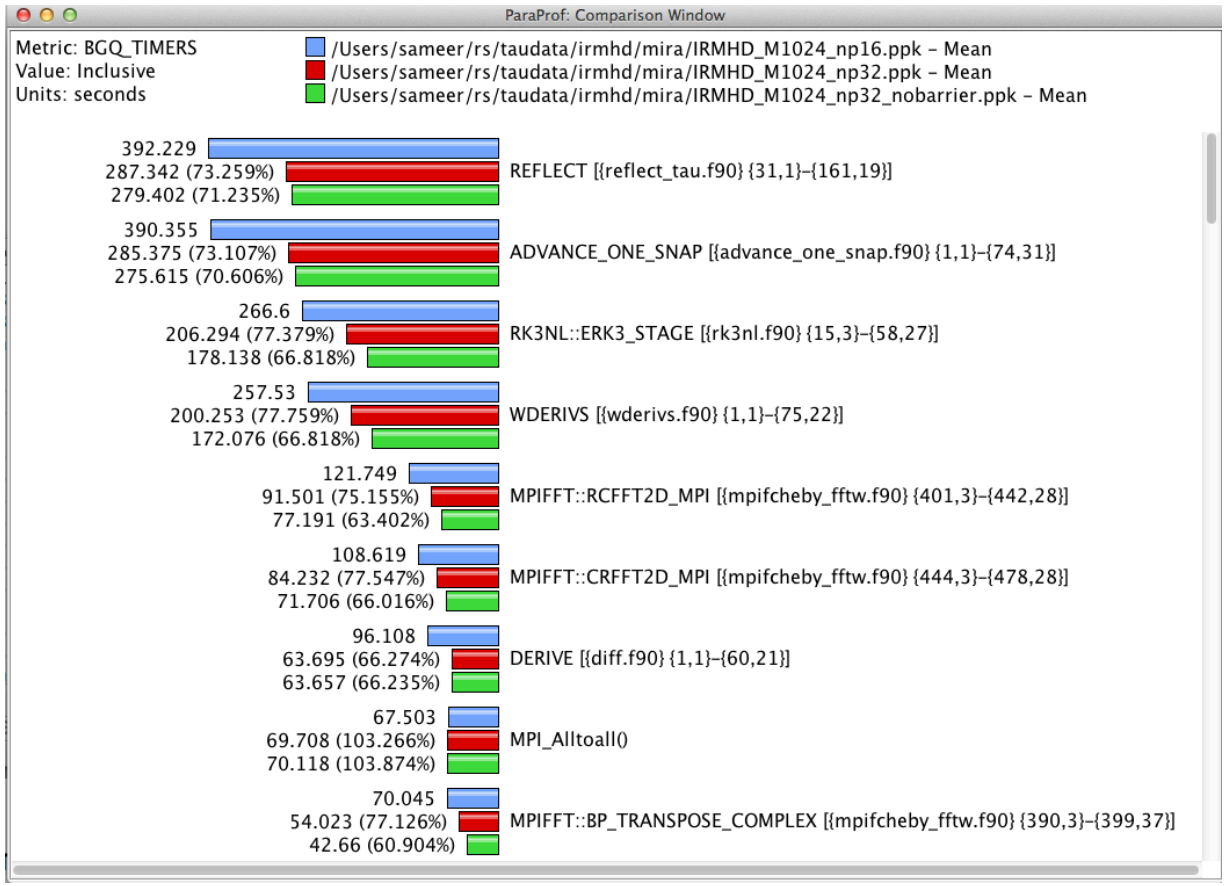
- Remove *MPI_Barrier* in regions where not needed



- Next, oversubscribe nodes …

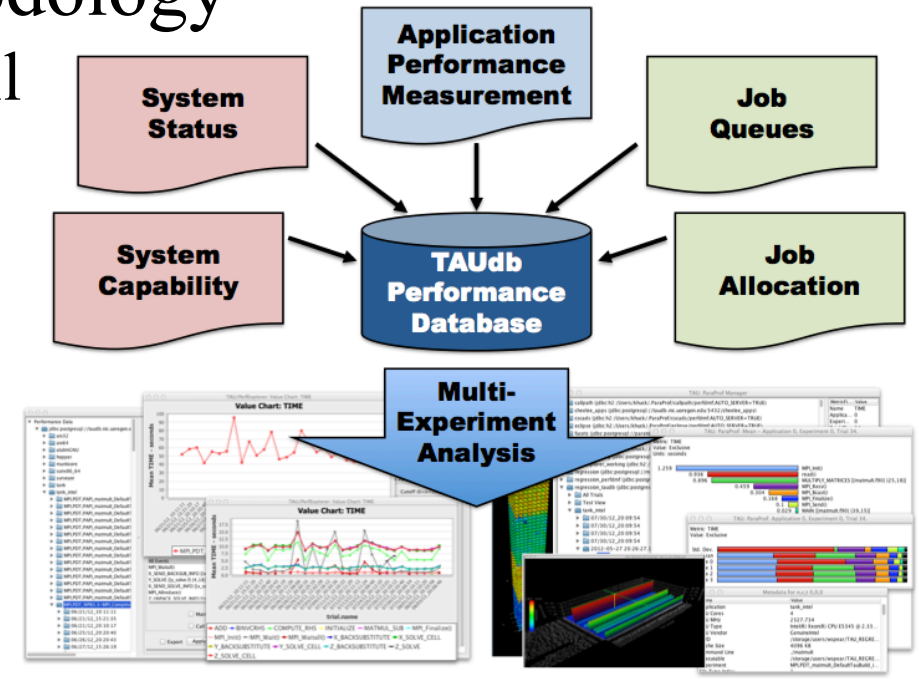
Oversubscribing Mira Nodes

- Vary #MPI ranks on nodes (1024 nodes)
  - 16 ranks per node (16K) versus 32 ranks per node (32K)
- Overall time improvement
  - 71.23% of original
- More efficient barriers within node leads to performance improvement



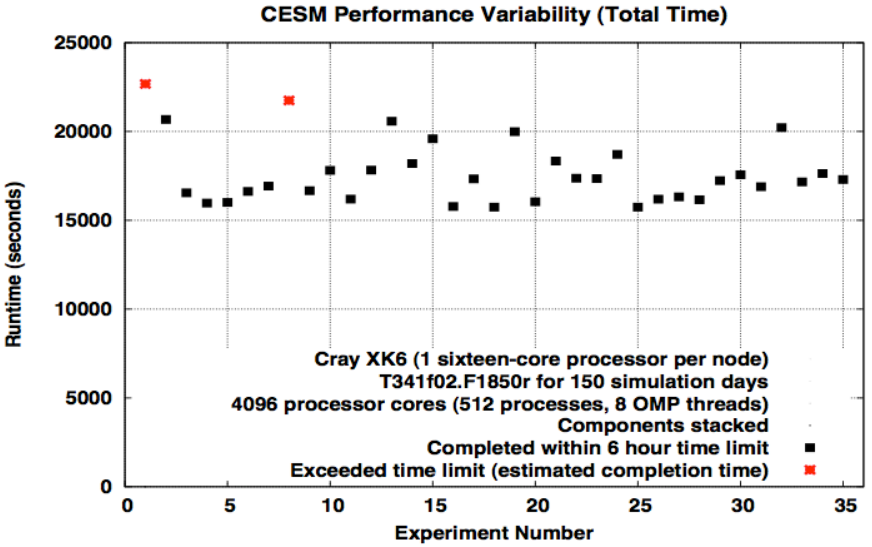
Performance Variability in CESM

- Community Earth System Model (CESM)
- Observed performance variability on ORNL Jaguar
  - Significant increase in execution time led to failed runs
- End-to-end analysis methodology
  - Collect information from all production jobs
    - problem/code provenance
    - system topology
    - system workload
    - process node/core mapping
    - job progress
    - time spent in queue
    - pre/post, total
  - Load in TAUdb, qualtify nature of variability and impact

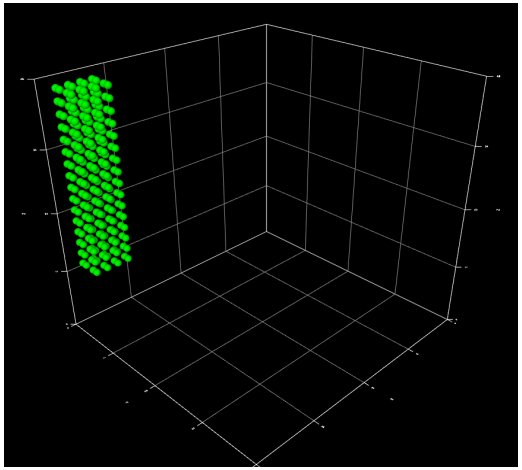


Example Experiment Case Analysis

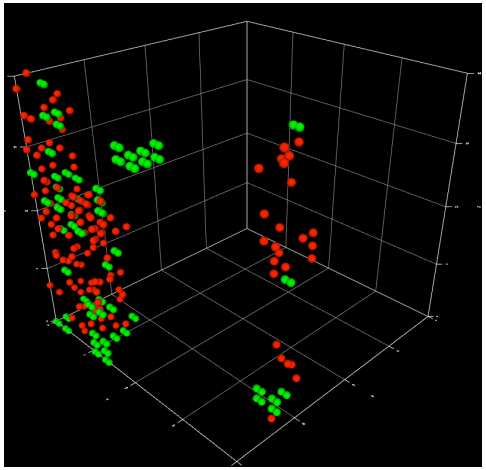
- 4096 processor cores
  - 6 hour request
  - 150 simulation days
- 35 jobs
  - May 15 – Jun 29, 2012
  - Two of which failed

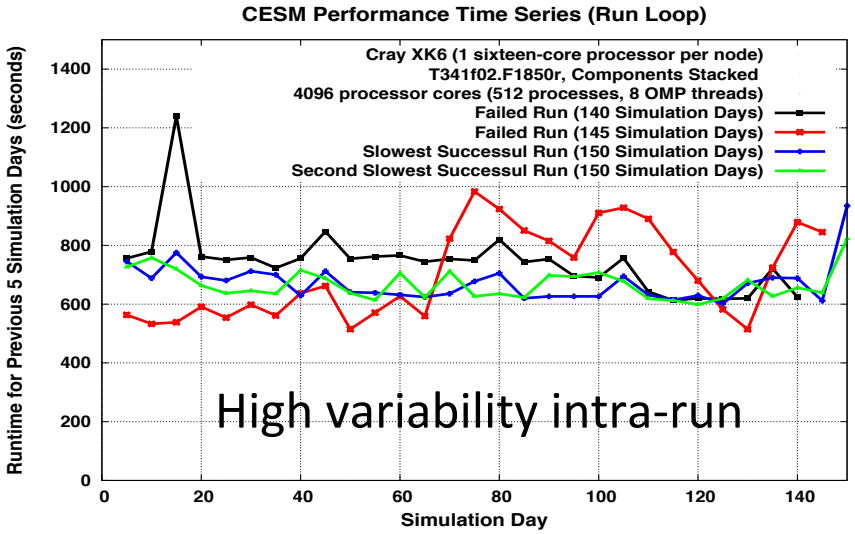


Minimum execution time

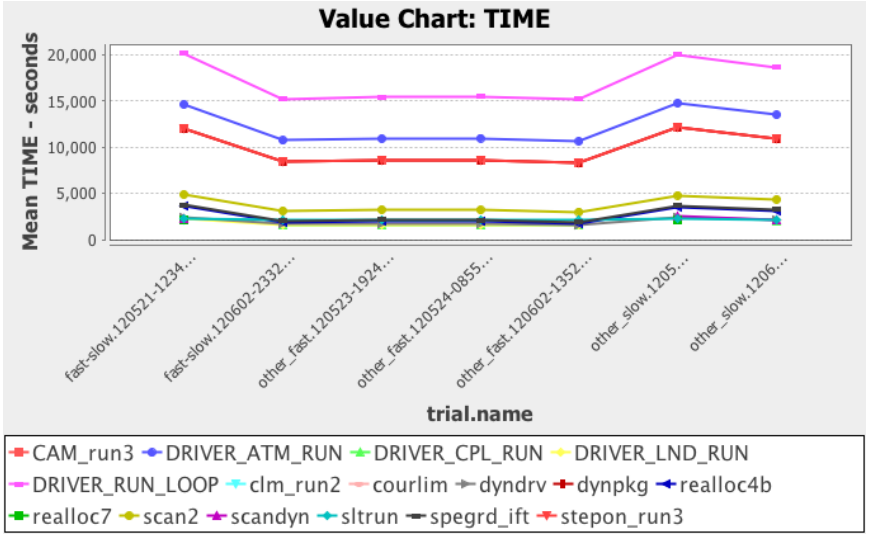


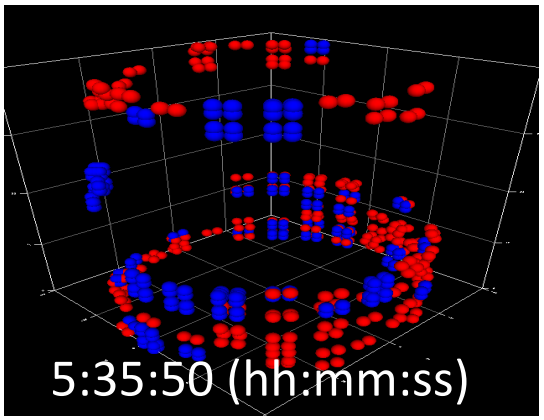
Maximum execution time



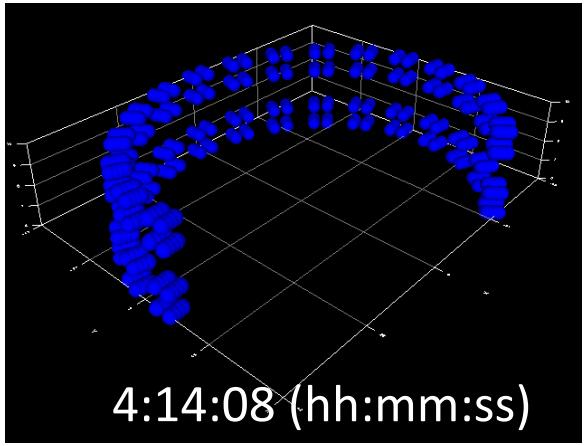


TAUdb and CCSM Data – Mismatched Nodes

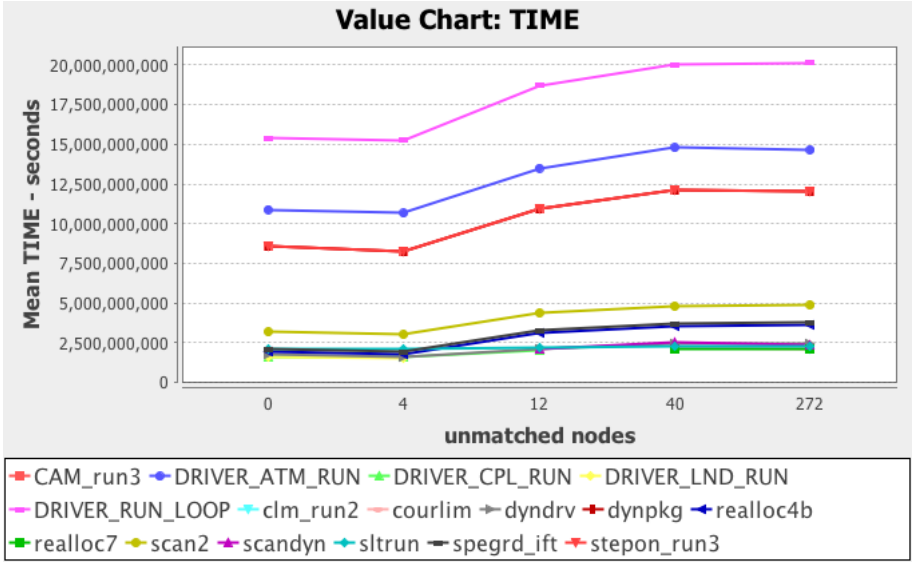




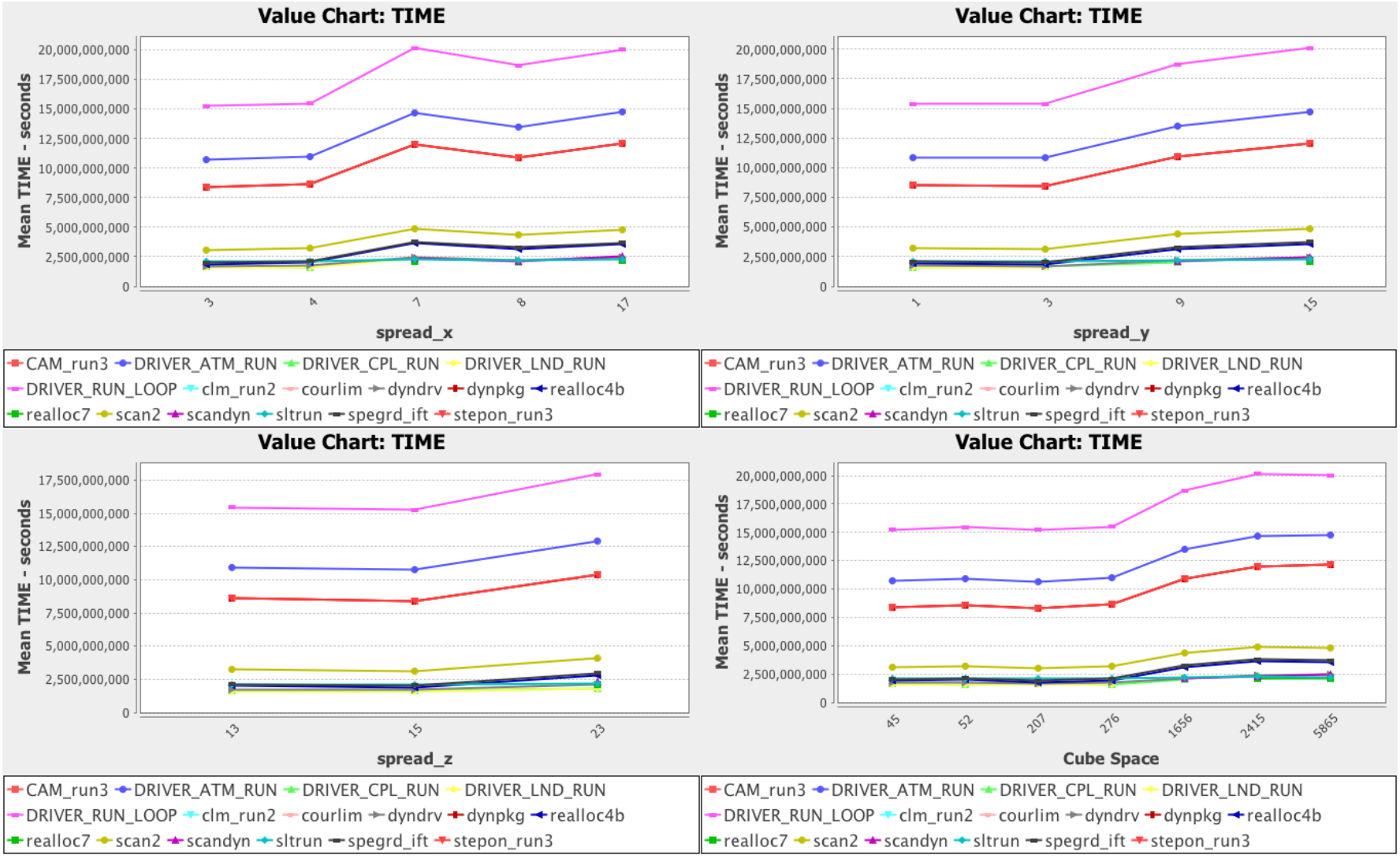
Slowest – 272 unmatched



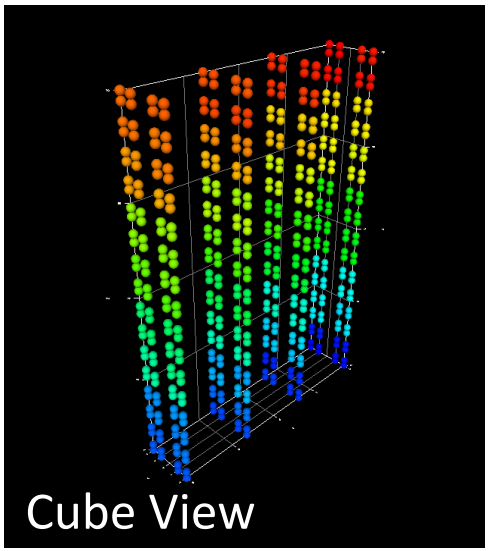
Fastest – 0 unmatched



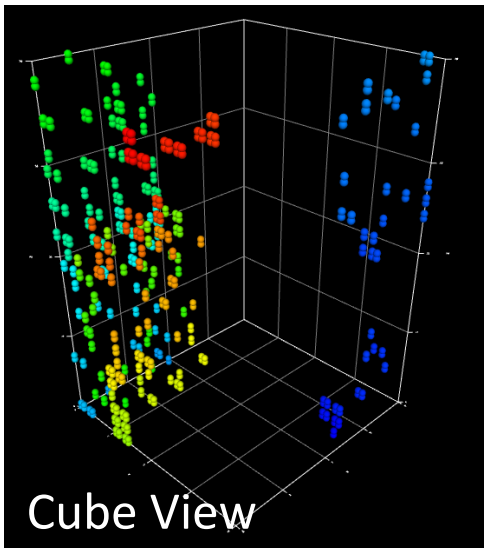
Extents of Gemini Network Matter



MPI Rank Placement Matters Too



Fastest case:
4:14:08
(hh:mm:ss)



Slowest case:
5:35:50
(hh:mm:ss)