INSTITUTE FOR SUSTAINED PERFORMANCE, ENERGY, AND RESILIENCE

SUPER Application and Institute Engagement:

Introduction

Engagement with SciDAC-3 Science Application Partnership (SAP) projects is at the core of SUPER, providing motivation, research directions, example code and problems, and opportunities for verification of approaches and near-term impacts. During FY15, SUPER has funded collaborations with 14 application projects.

SUPER contributes to the SAPs by promoting best practices, by providing technology and architecture expertise, and by collaborating directly on performance engineering tasks. Some example activities and quantifiable impacts are described here.

Engagement with the SAPs has also led to direct collaborations with each of the other three SciDAC Institutes on crosscutting issues and technologies.

CAM / MPI_Alltoallv

Patrick Worley
Oak Ridge National Lab

Background

❖ Applying Computationally Efficient Schemes for BioGeochemical Cycles (ACES4BGC) BER SciDAC Partnership (PI: F. Hoffman). Also relevant to Multiscale Methods for Accurate, Efficient, and Scale-Aware Models of the Earth System (PI: W. Collins)

❖ The Community Atmosphere Model (CAM), used in both the Community Earth System Model (CESM) and the Accelerated Climate Model for Energy (ACME), has an effective load balancing scheme, remapping work as needed between computation of the dynamics and of the physical parameterizations.

❖ The effectiveness of load balancing depends on the relative cost of associated MPI communication. Algorithm typically uses MPI_Alltoallv but a tunable point-to-point (P2P) implementation is also supported.

❖ April 2015 experiments on Titan, the Cray XK7 at the Oak Ridge Leadership Computing Facility, demonstrated unexpectedly high MPI_Alltoallv cost in the load balancing algorithm, resulting in a 1.4X slowdown for a high resolution simulation as compared to not using load balancing.

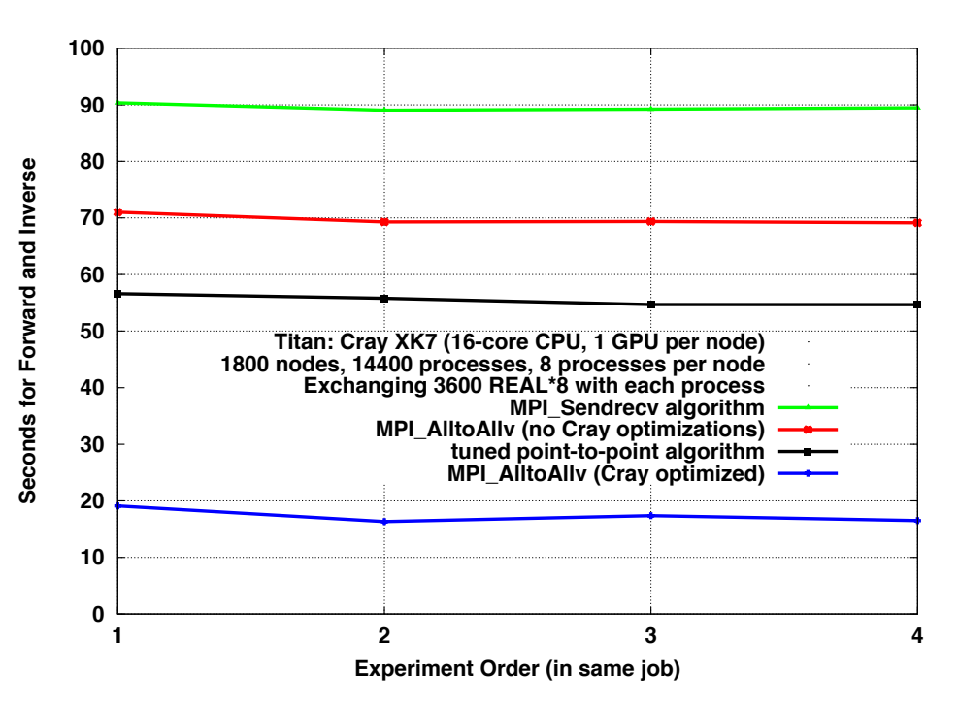
Approach and Results

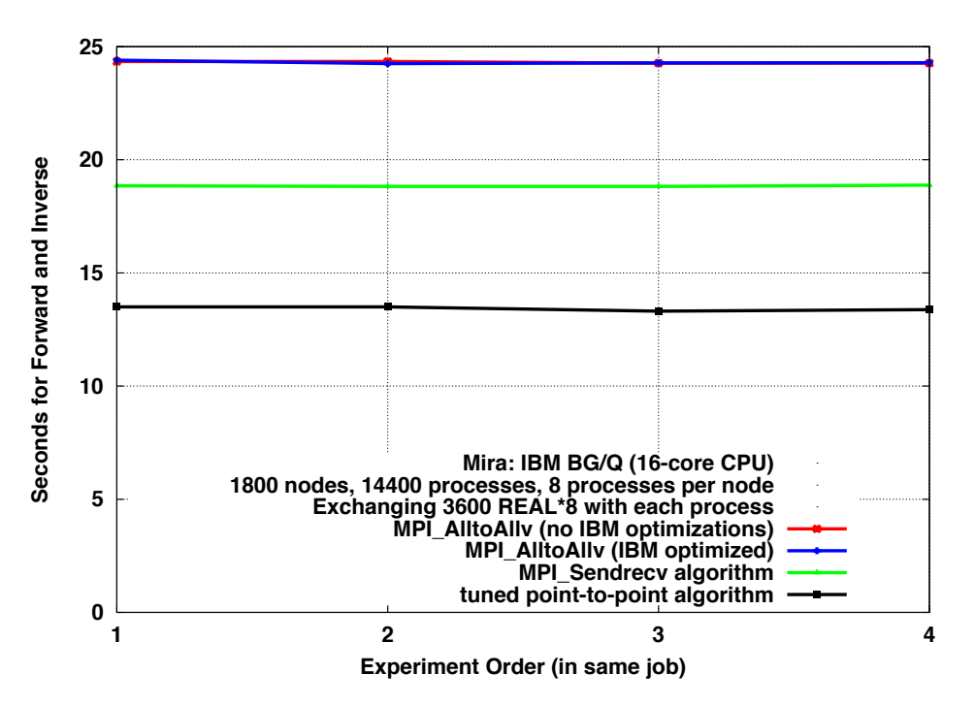
❖ Repeat CAM load balancing experiment using P2P implementation.

– Performance is much improved on Titan. Load balancing now improves performance.

❖ Evaluate MPI_Alltoallv and P2P implementation in standalone kernel using a uniform distribution, measuring forward and inverse together

– 14400 processes, 8 processes per node, vector length 51,840,000, individual message size 3600 doubles

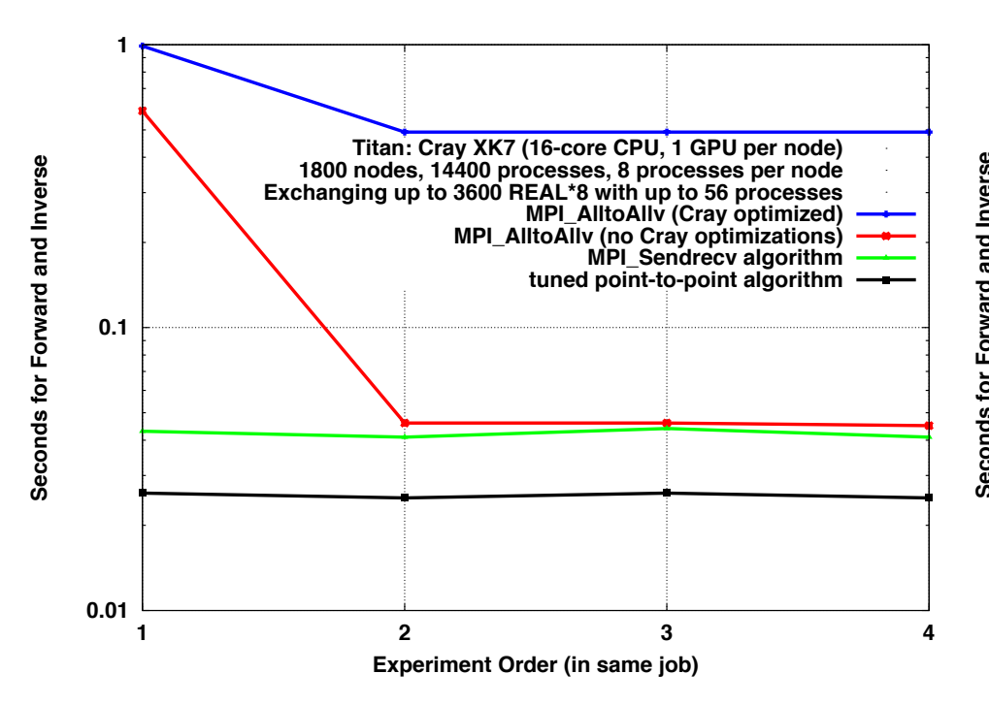


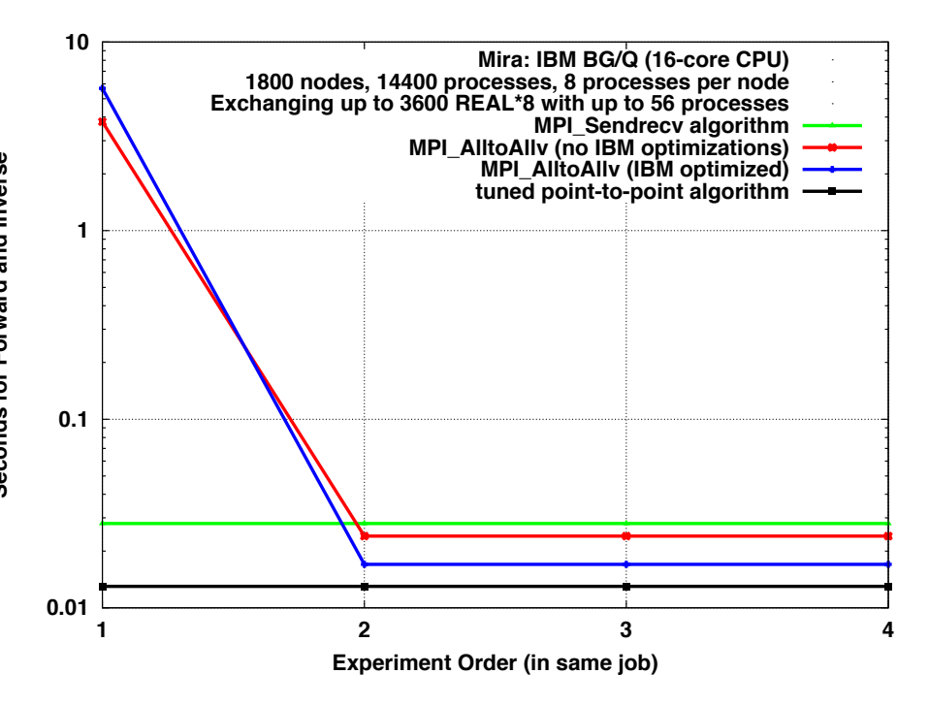


– Cray optimized MPI_Alltoallv performs best on Titan, but P2P is faster on Mira. Does not explain CAM load balancing performance issues.

❖ Evaluate MPI_Alltoallv and P2P implementation in standalone kernel using message pattern extracted from CAM load balancing example.

– 14400 processes, 8 processes per node, vector lengths 33,000 to 63,000, individual messages 850 to 3600 doubles, per process send (receives): 12 to 56





– On Titan, a P2P algorithm is fastest, and the vendor optimizations hurt performance on one platform. On Mira, P2P is slightly faster, but vendor optimized MPI_Alltoallv is competitive.

Conclusion

❖ Message pattern can matter, and optimizations for standard usage may hurt in special cases.

LibTensor / CTF

Khaled Ibrahim, Samuel Williams
Lawrence Berkeley National Lab

Evgeny Epifanovsky, Anna Krylov
University of Southern California

Background

❖ Electronic Excitations in Molecular and Nanoscale Materials BES SciDAC Partnership (PI: Martin Head-Gordon)

❖ In the past, the LibTensor tensor contraction library (USC) used in this project was restricted to running on large SMPs with spinning disks (**bad match for DOE supercomputers**)

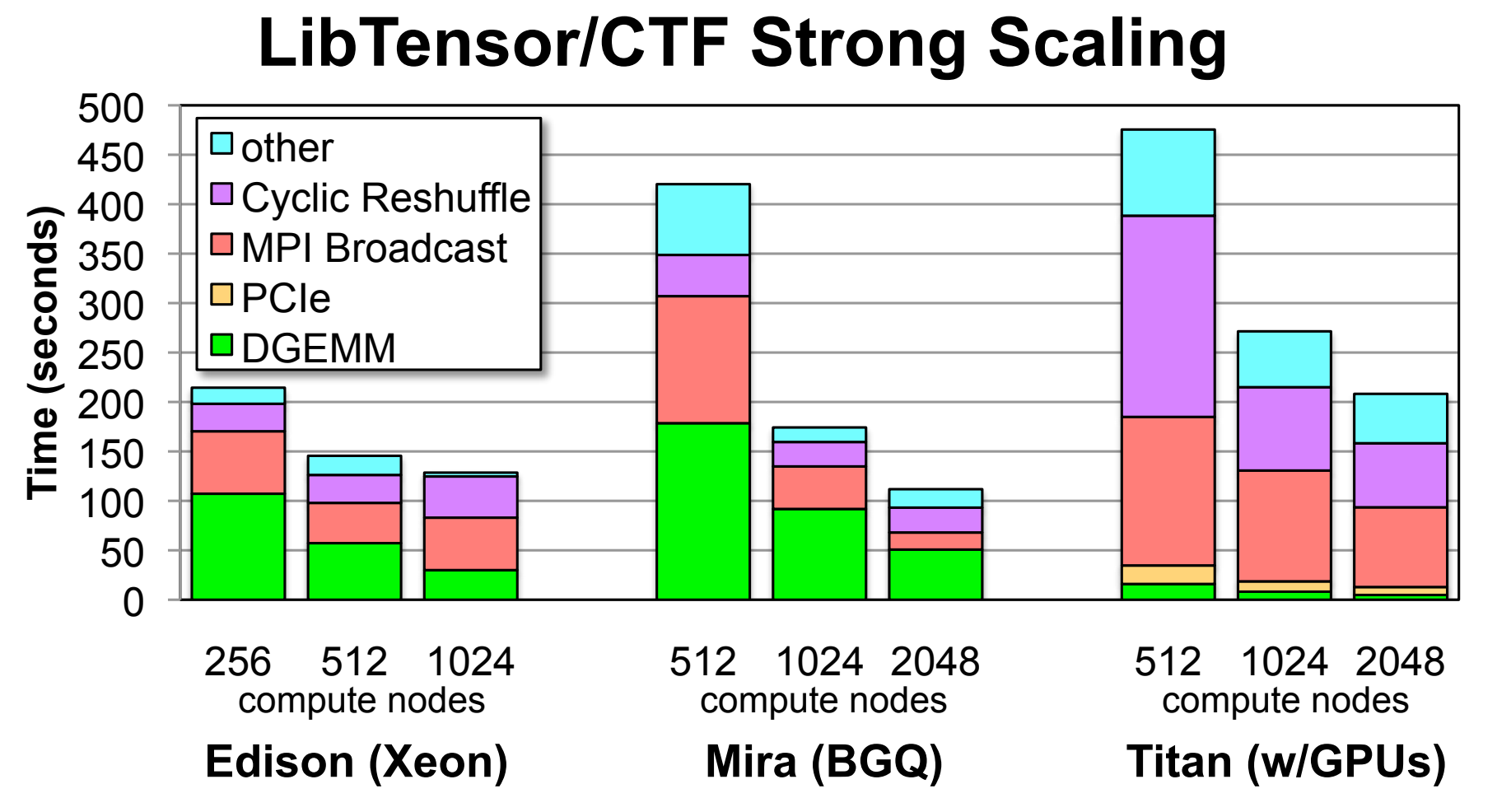
❖ USC/LBL decided to leverage the **Cyclops Tensor Framework (CTF)** as a backend for LibTensor…

- library for scalable tensor contractions on **distributed-memory supercomputers**
- created at UC Berkeley by Solomonik et al. in 2013
- uses a cyclic distribution of tensor elements that allow for a regular parallel distribution (**MPI All-to-All stresses the network**)
- distributed SUMMA-based matrix-matrix multiplication (**MPI row broadcasts** plus calls **to highly-tuned vendor BLAS routines**)

❖ LBL/SUPER researchers evaluated performance and scalability…

- Edison (Cray XC30 at NERSC) up to 16,384 cores
- Mira (IBM Blue Gene/Q at Argonne) up to 32,768 threads
- Titan (**GPU-accelerated** Cray XK7 at Oak Ridge) up to 2048 GPUs
- **strong-scaled** methylated uracil water dimer test problem with 302 basis functions and C_s symmetry
- Modified to exploit tuned BLAS routines (including CULA for GPUs)
- Tuned MPI vs. OpenMP (or CUDA) to balance MPI vs. compute

LibTensor/CTF Strong Scaling



Observations

❖ **GPU-acceleration reduced DGEMM to 3% of the run time.**

❖ Unfortunately, Titan uses an older 3D torus network on which **SUMMA's MPI broadcasts did not scale well**.

❖ Although Edison's newer Aries network saw better performance, salability was also impaired.

❖ Although local DGEMM was slowest on Mira, **Mira's network ensured it provided the best overall performance and scalability**.

❖ **Libtensor/CTF on 2K nodes of Mira is more than 150x faster than the original Libtensor/SMP** backend running on a large, multi-socket, big-memory SMP at NERSC.

❖ Future work will examine customized MPI collectives to improve performance and scalability on Edison and Titan.

BIGSTICK

Hongzhang Shan, Samuel Williams
Lawrence Berkeley National Lab

Kenneth McElvain
UC Berkeley

Calvin W. Johnson
San Diego State University

Background

❖ CalLat NP SciDAC Partnership (PI: Wick Haxton)

❖ Configuration-interaction (CI) method is a popular technique for solving quantum many-body systems

- often cast as large eigenpair problem using iterative methods like Lanczos
- matrix dimensions can easily exceed 1 billion, **nonzeros can exceed 10 trillion**
- stored matrix representations can require in **excess of 100TB of memory** (1000s of compute nodes are needed to solve these problems).

❖ BIGSTICK is a scalable, memory-efficient CI code

- a series of tables are used to compute nonzeros on the fly
- **reduces memory requirements down to less than 0.5TB**
- **As such, BIGSTICK can run very large problems on small machines**

❖ Unfortunately, raw performance can suffer…

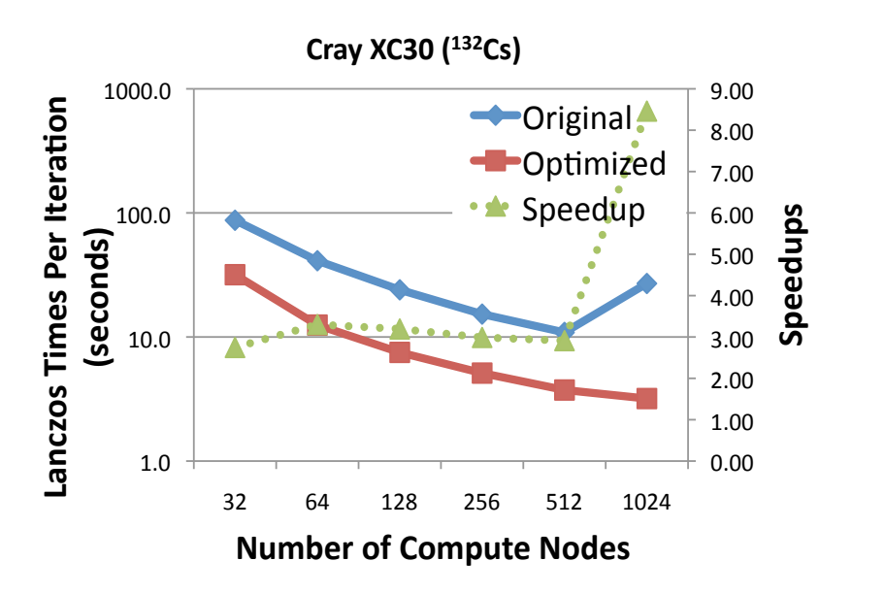
- performance per nonzero is now highly variable (table lookups)
- complex data and computational decomposition can lead to substantial load imbalance and performance degradation.

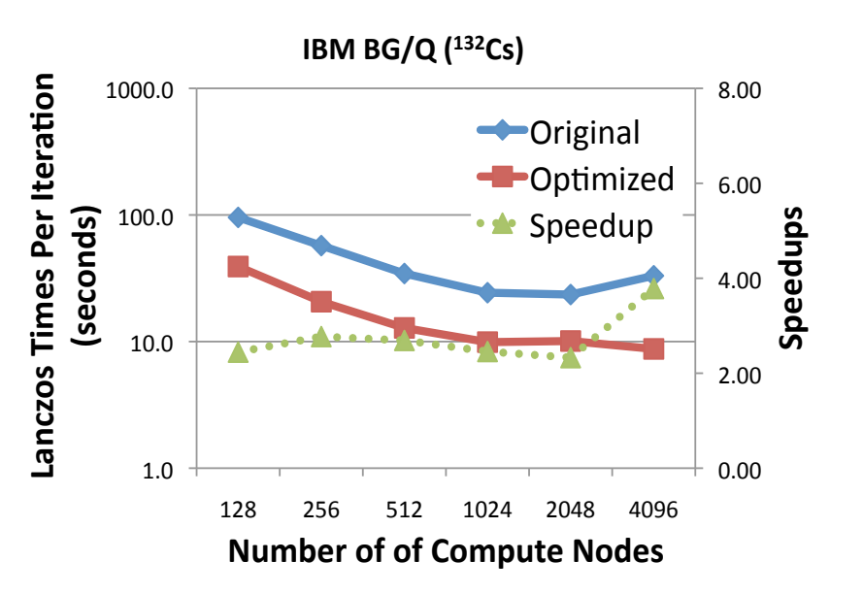
❖ LBL, UCB, and SDSU collaborated to evaluate and optimize BIGSTICK on two large DOE supercomputers…

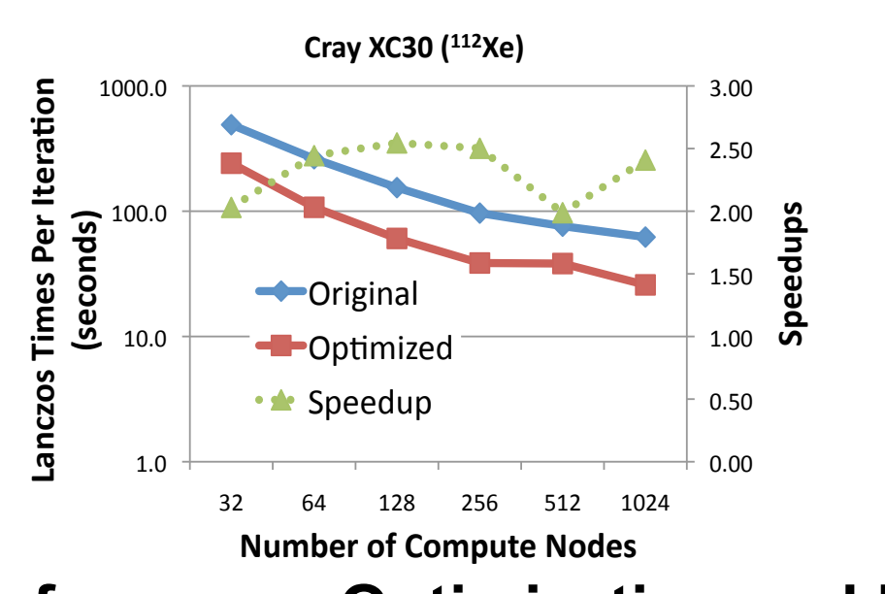
- Cray XC30 at NERSC (Edison)
- IBM Blue Gene/Q at Argonne (Mira)

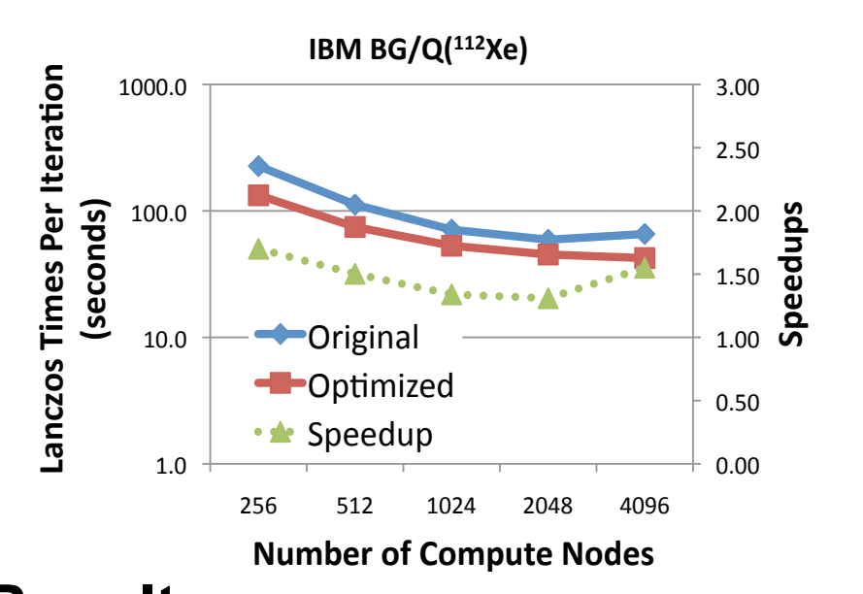
❖ We use two challenging test problems…

- ^{132}Cs using a ^{100}Sn frozen core with 5 valence protons and 27 valence neutrons
- ^{132}Xe using a ^{100}Sn frozen core with 4 valence protons and 8 valence neutrons (^{132}Xe is larger (easier to load balance), but sparser (harder to get efficiency))









Performance Optimization and Results

❖ Initially, performance was impaired by matvec load imbalance and a lack of scalability in Lanczos reorthogonalization

❖ Applied a series of on-node and inter-node optimizations…

1. Using empirical performance observations to load balance OpBundles (nonzeros)
2. Tuning MPI+OpenMP for each machine/scale
3. Fusing collectives… MPI_Reduce+MPI_scatter \ custom P2P MPI_reduce_scatter
4. Using empirically-tuned reduced concurrency for reorthogonalization

❖ **Overall, we improved scalability and performance by 1.3x - 8x.**

❖ Future work will continue load balancing optimizations as well as move to NERSC's forthcoming Cori supercomputer

XGC1

Eduardo D'Azevedo, Sarat Sreepathi, Patrick Worley
Oak Ridge National Lab

Choong-Seock Chang, Robert Hager, Seung-Hoe Ku, Jianying Lang
Princeton Plasma Physics Lab

Eisung Yoon
Rensselaer Polytechnic Institute

Background

❖ Center for Edge Physics Simulation (EPSi) FES SciDAC Partnership (PI: C-S Chang)

❖ Continued performance optimization and engineering for XGC1 and XGCa plasma microturbulence particle-in-cell simulation codes.

❖ Initial implementation of new nonlinear Fokker-Planck collision solver was too costly to be used in production. Diagnosis indicated opportunities to improve performance by decreasing serial complexity, improving OpenMP threading, and improving load imbalance.

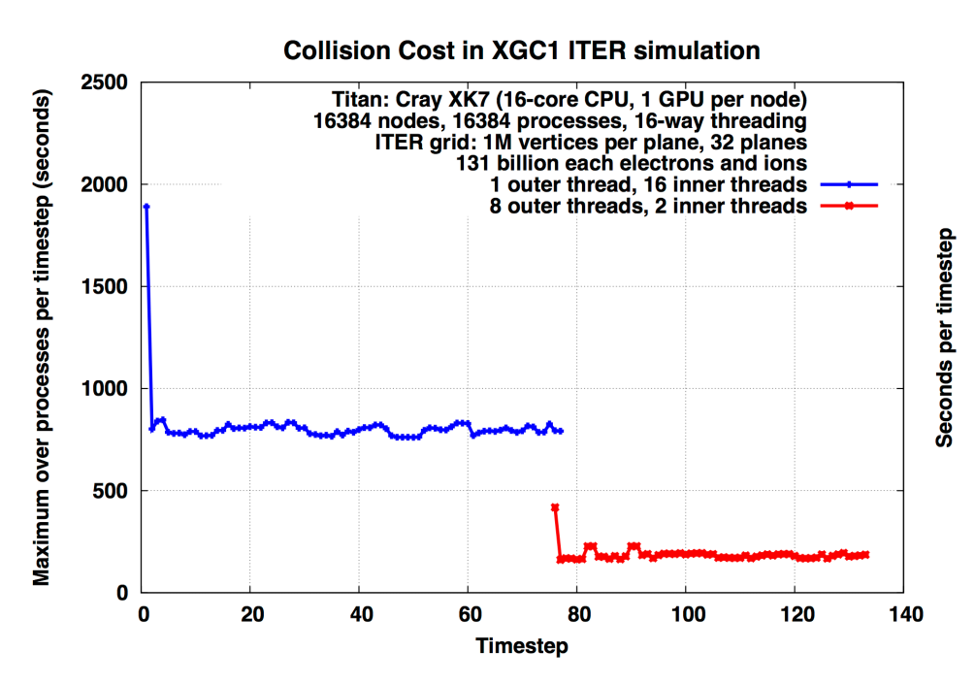
❖ At scale, MPI communication overhead is non-negligible and also sensitive to network topology of allocated nodes.

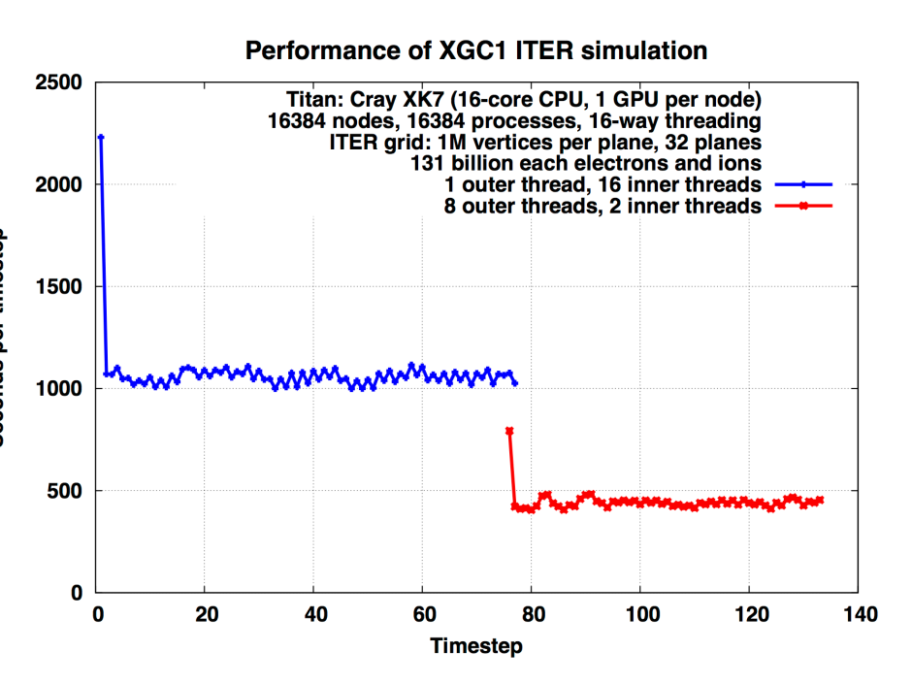
Performance Optimization of Collision Operator

❖ All three performance optimization thrusts were investigated simultaneously.

❖ E. S. Yoon developed a series of increasingly more computationally efficient implementations, decreasing the cost of the collision operator in a realistic run by a factor of 3X.

❖ Outside of the collision operator, XGC1/XGCa makes effective use of OpenMP parallelism. In the collision operator there is insufficient work in a single grid cell to fully exploit OpenMP threading. By moving to threadsafe linear solvers, we were able to also introduce threading at a higher level, to the loop over cells. This is implemented with a nested OpenMP formulation, with the number of threads at each level a runtime performance tuning option.



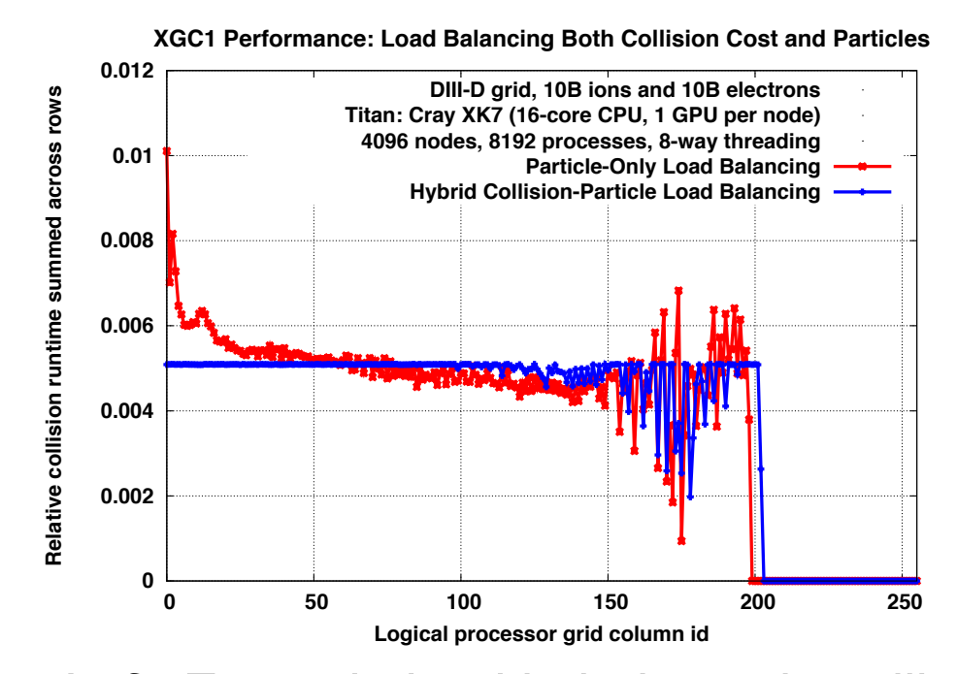


Performance impact of nested OpenMP. Change from no outer threading to 8 outer and 2 inner threads in collision operator in June 2015 large Titan runs demonstrates 5.5X per process improvement in collision operator, reducing model runtime by over 2X. (Both E. S. Yoon's optimizations and load balancing enabled in these results.)

❖ Collision operator exhibits significant load imbalance relative to the grid decomposition that load balances the particle distribution. Moreover, both particle and collision operator load imbalances evolve with the simulation. A new load balancing strategy was developed to address the combined particle and collision operator load imbalances:

❖ balance collision cost subject to constraint on particle load imbalance

❖ optimize XGC1 performance by varying constraint periodically, converging to the optimum if distributions are static and adapting to the changing distributions otherwise



Timestep cost (secs)	Step 1: Particle-only load balance	Step 2: Particle +Collision
1 outer thread	2230	1072
8 outer threads	793	422

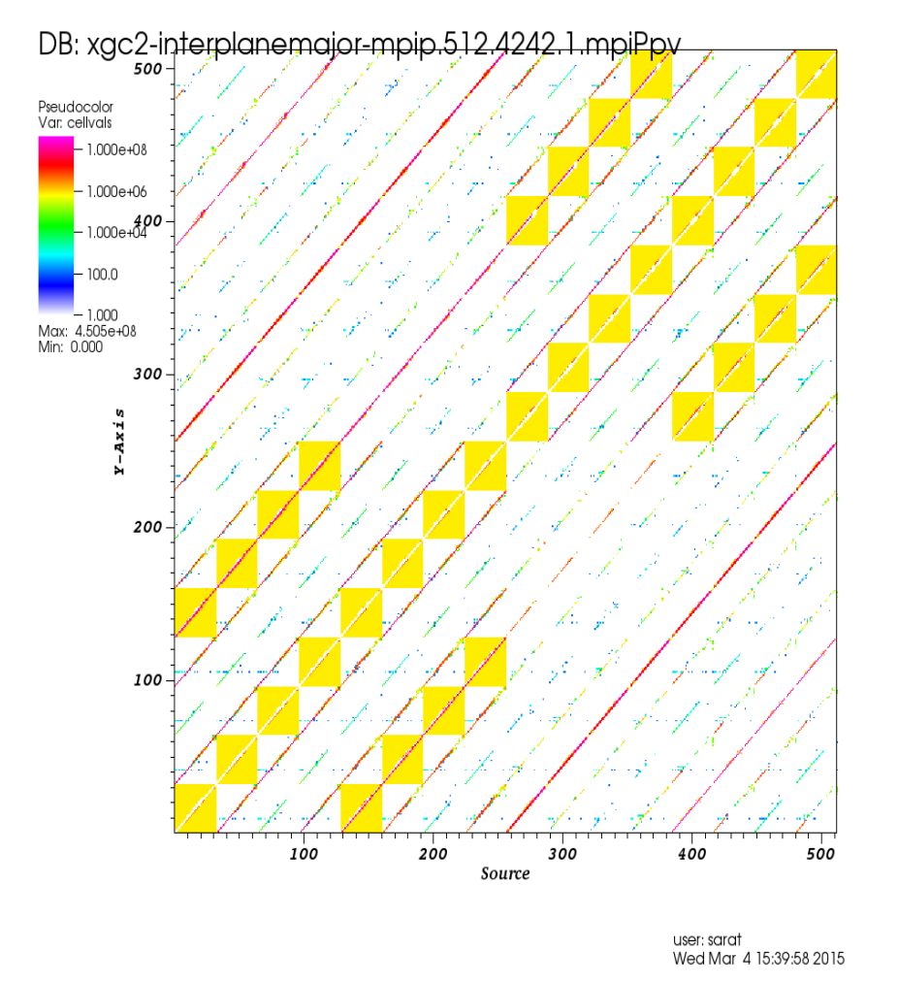
Left: Example load imbalance in collision operator cost, comparing load balancing only particle distribution with also load balancing collision cost. Cost is summed over rows of virtual 2D processor grid. Full model performance improvement was 30% for this example. Right: Performance improvement in June 2015 Titan ITER experiments between first step using only particle load balancing and second step that also balances collision cost.

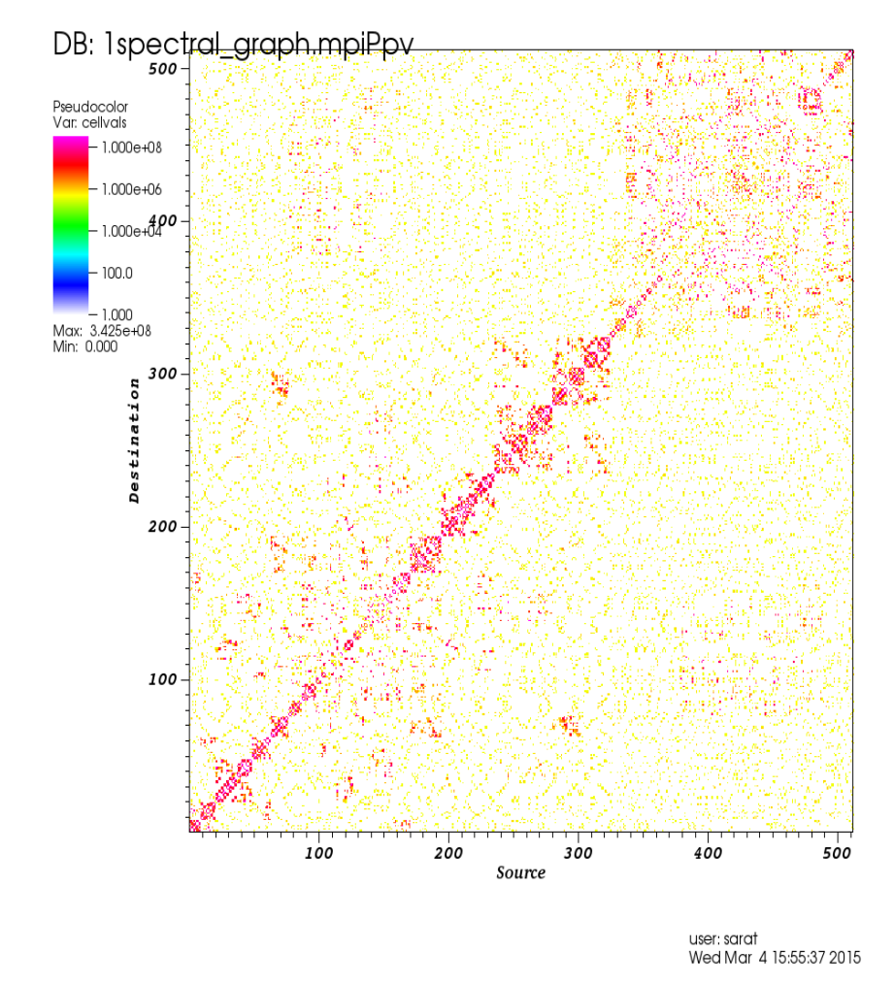
Performance Optimization of MPI communication

❖ Goal is to optimize task placement of XGC1 to account for application communication characteristics and physical node layout of allocated jobs on DOE Leadership Computing Systems.

❖ Job scheduling system on Titan (and many other systems) allocates non-contiguous nodes to jobs. Default MPI ordering does not account for actual physical node layout available to application, resulting in degraded performance.

❖ Initial approach has been a two phase optimization strategy: (a) collect communication profile of XGC1 using mpiP, (b) optimize MPI task mapping by coupling application communication graph with locality information of allocated nodes. As part of this, have developed and experimented with various reordering algorithms, including spectral bisection, neighbor join tree etc.





Communication volumes of XGC1 using interplane-major mapping and poloidal decomposition within the plane in two scenarios (512 ranks) (left) default MPI ordering and (right) reordered MPI tasks using spectral bisection with weighted Laplacian technique. The axes represent source and destination MPI ranks.

❖ Preliminary results: 12-15% improvement in communication performance.

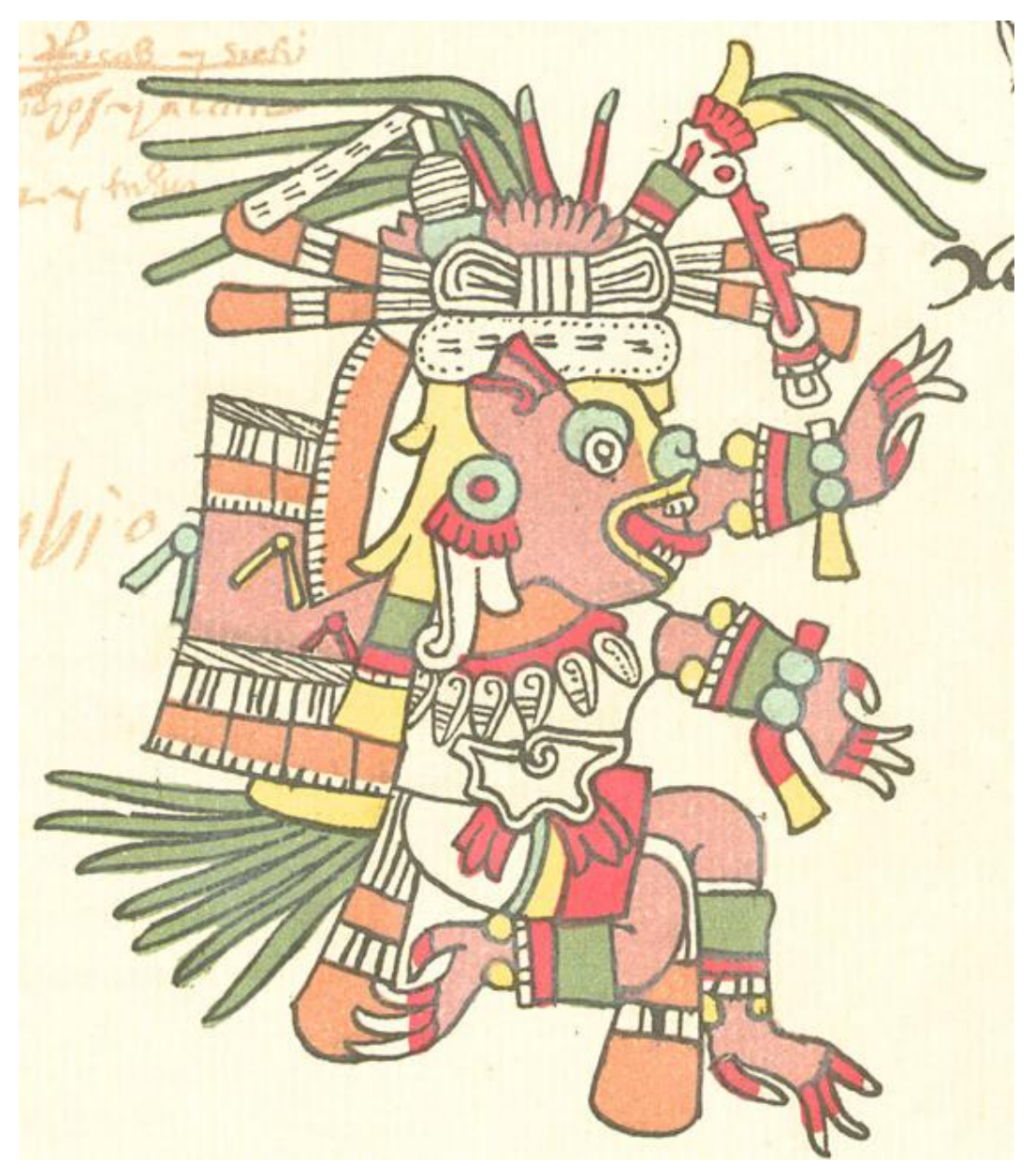
Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research

Activities and Impacts

U.S. DEPARTMENT OF ENERGY | Office of Science

Xolotl/Paraspace Performance Comparison

Phillip C. Roth
Oak Ridge National Lab

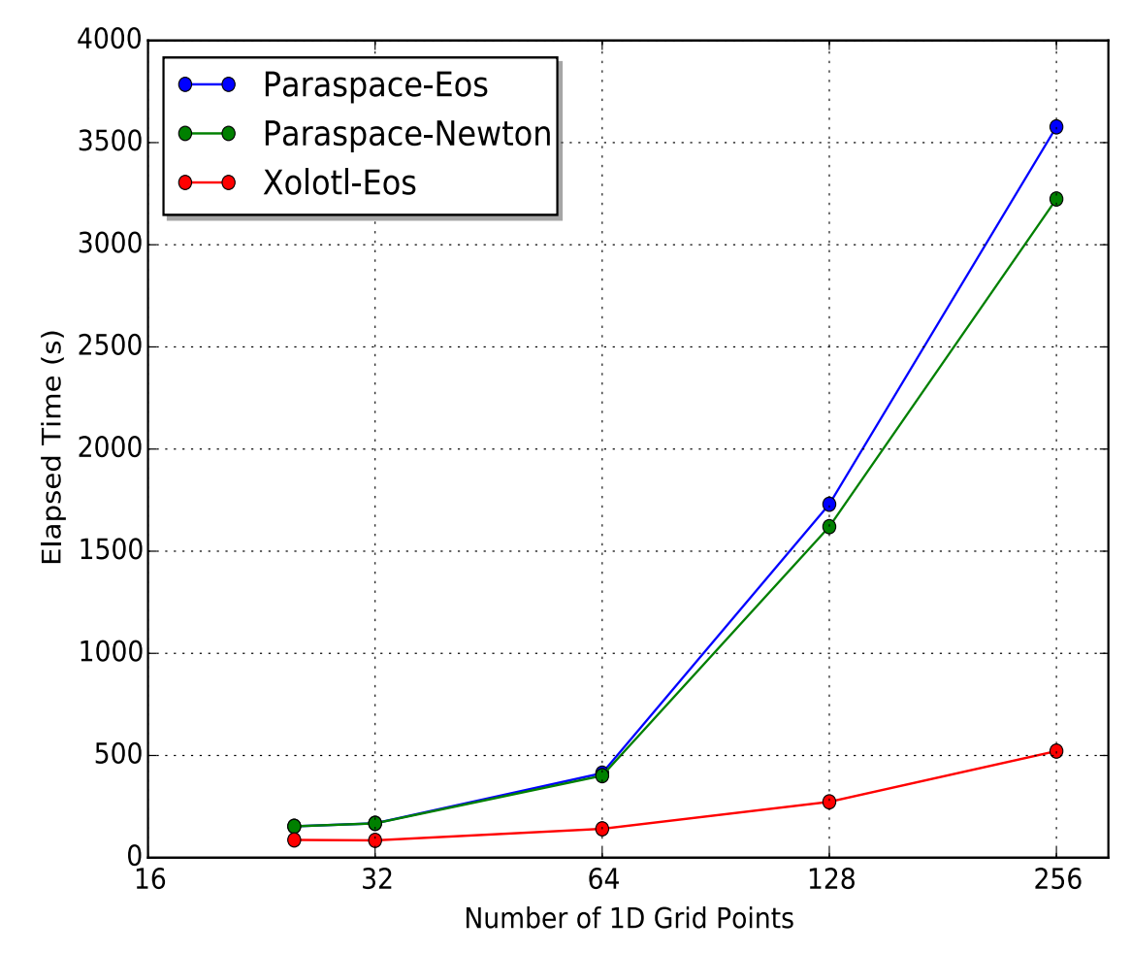


- Xolotl plasma surface interactions simulator
  - New code being developed as part of Plasma Surface Interactions FES SciDAC Partnership (PI: Brian Wirth)
  - Continuum model for solving cluster dynamics advection-diffusion-reaction equation with incident flux
  - Support for 1D, 2D, and 3D problems
  - Uses PETSc solver
  - MPI only
  - Open source, publicly available via SourceForge

- Recently compared Xolotl performance and scalability against that of Paraspace
  - Paraspace: PARAllel SPAtially-dependent Cluster Evolution
    - Implements parallel cluster dynamics with spatial dependence by solving reaction diffusion equation with incident flux
    - Mature code, but limited to 1D and OpenMP only
  - Used both programs to simulate same simulation problem
    - Helium retention in Tungsten diverter wall with incident He flux of $4x10^{25}$ He/m^2/s for $1x10^{-6}$ seconds
    - Used several 1D data discretizations (mainly nx=256, dx=0.25)
    - Ran on Eos, a Cray XC30 within the Oak Ridge Leadership Computing Facility
      - Two eight-core Intel E6-2670 processors at 2.6GHz per node, Hyper-Threading supported
      - 64 GB SDRAM per node
      - Aries interconnection network
- Determined best run-time configuration
  - Xolotl: 32 single-threaded processes per node (Hyper-Threading enabled), process affinity to NUMA node, and 16 processes per NUMA node
  - Paraspace: 32 threads (Hyper-Threading enabled), affinity to NUMA node

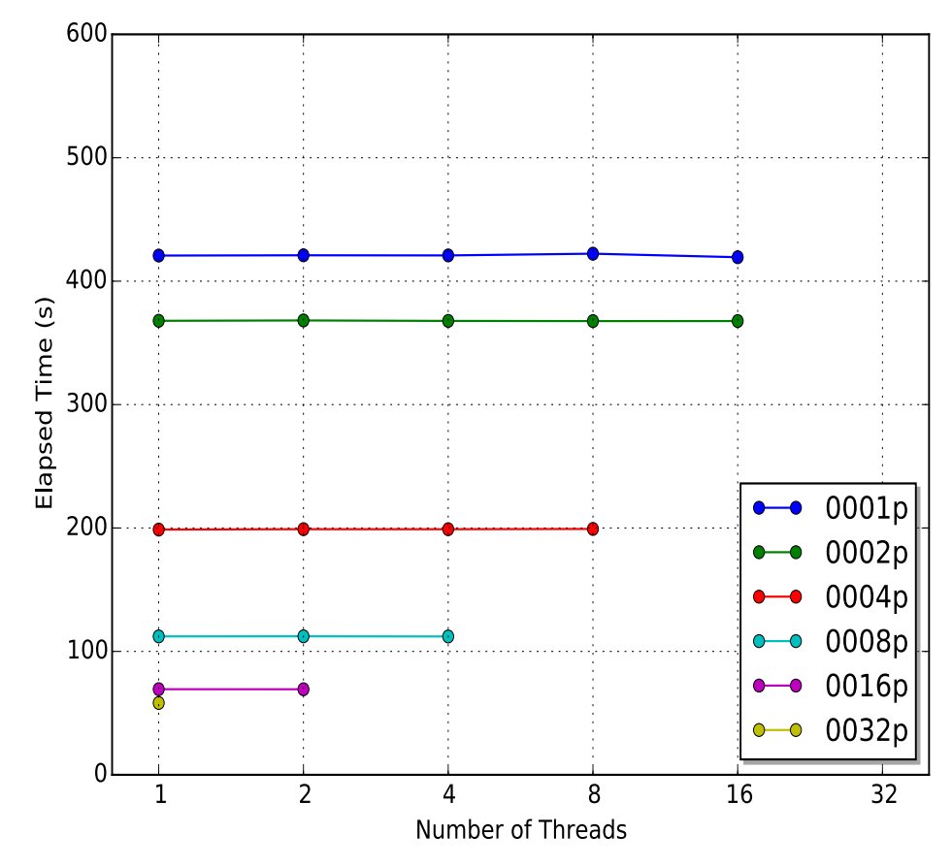
Time required to simulate full He retention in W100 problem on one Eos compute node

Metric	Xolotl	PARASPACE
Elapsed time (s)	520.32	3548.54
Number of time steps	104	227
Throughput (time steps/s)	0.200	0.0640

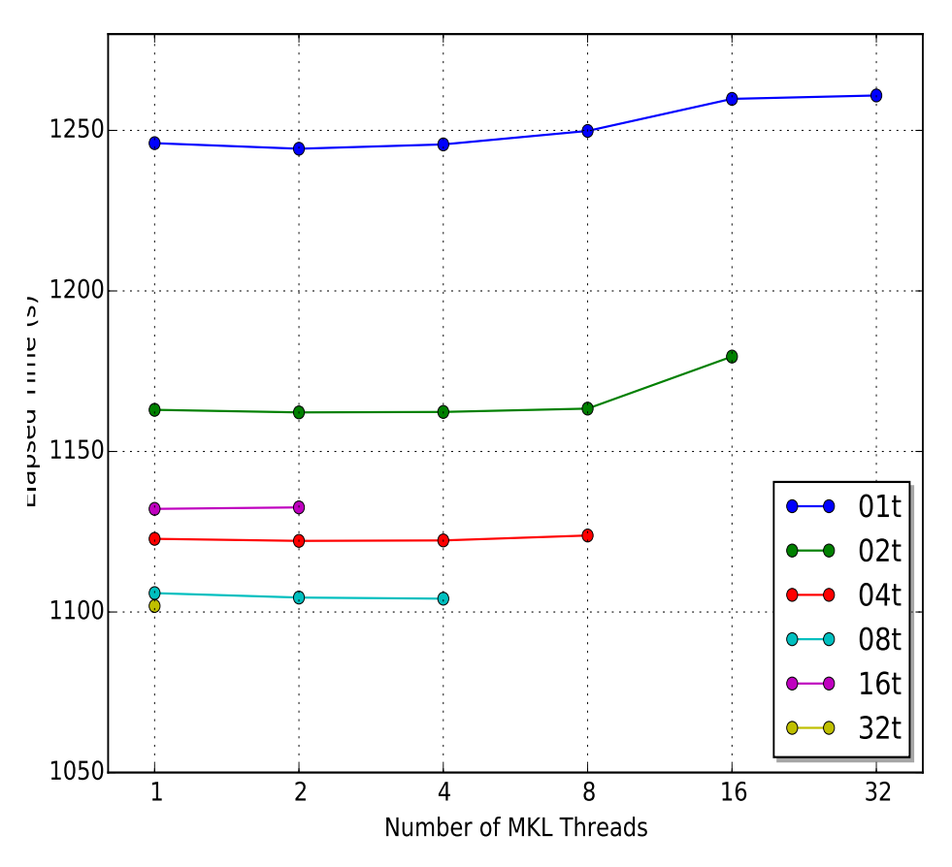


Time required to simulate full He retention in W100 problem on one Eos compute node with varying numbers of grid points

Time required to run 10 Xolotl time steps, Hyper-threading enabled



Time required to run 5 Paraspace output time steps, Hyper-threading enabled



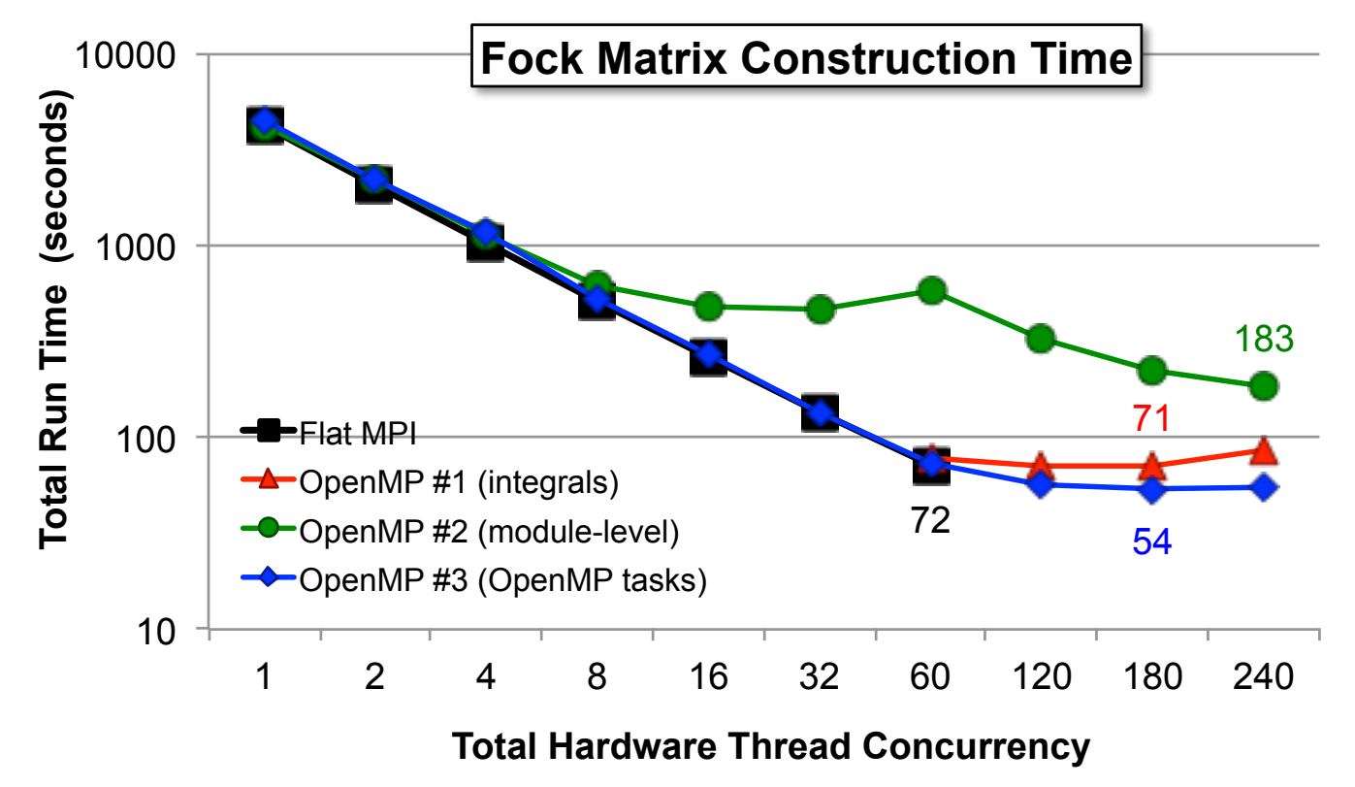
- Xolotl outperformed Paraspace on full problem, scaled well when increasing problem size, but…
  - …Paraspace computes more time steps at higher accuracy (hypothesized)
  - Currently testing relaxed Paraspace solver tolerances that reduce number of time steps but may also reduce accuracy
- Working on threading/GPU optimizations for Xolotl
  - Targeting 2D and 3D problems
  - Little improvement expected for 1D problems including the one used in this performance comparison

NWChem OpenMP Threading

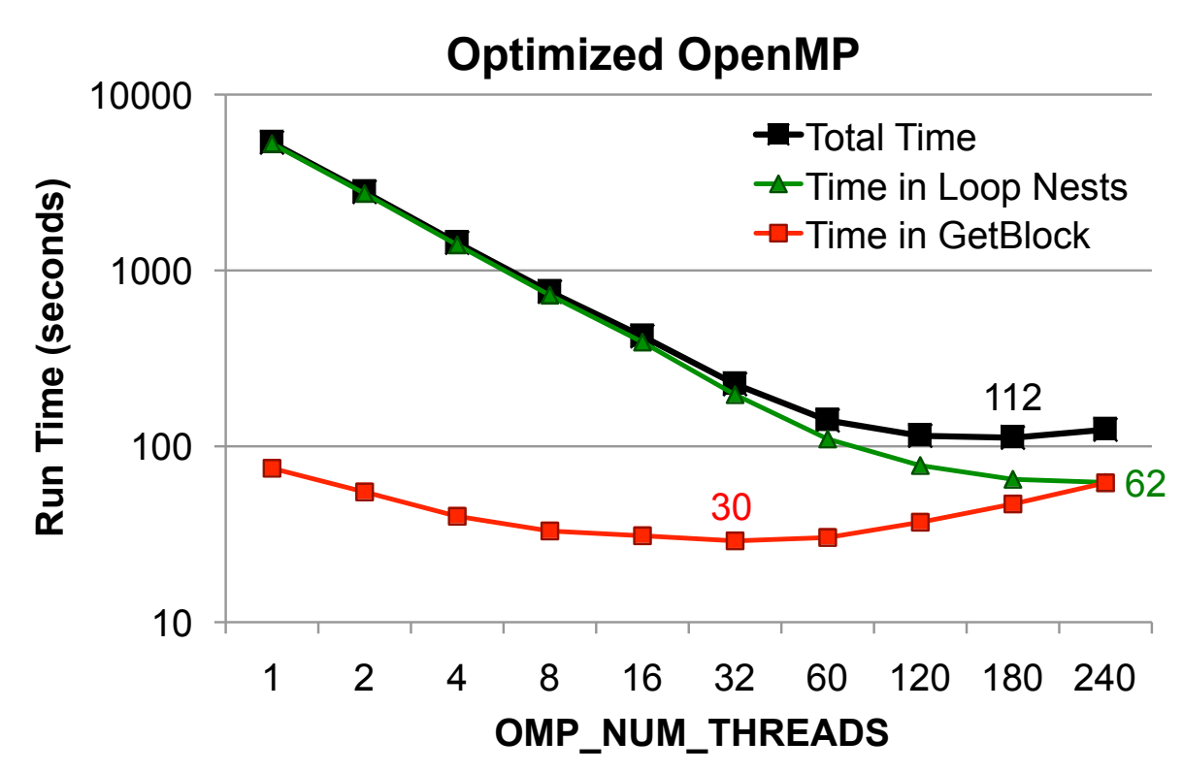
Hongzhang Shan, Bert de Jong, Lenny Oliker., Samuel Williams
Lawrence Berkeley National Lab

Background

- ❖ Developing Electron-Correlated Methods for Excited State Structure and Dynamics in the NWChem Software Suite BES SciDAC Partnership (PI: Christopher J. Cramer)
- ❖ Goal: Accelerate NWChem performance by implementing thread-level parallelism on the Intel Phi many-core architecture.
- ❖ Examined two important NWChem modules: Coupled Cluster Triples Algorithm CCSD(T) & Fock Matrix Constructions of TEXAS integral.
- ❖ Optimization insights shared with community via NERSC training



Performance of Fock Matrix Construction using our three approaches. The flat MPI implementation is limited to 60 processes, while the threaded version can use all 240 hardware thread contexts and results in a 1.64x speedup



Optimized OpenMP CCSD(T) run time, showing an overall speedup of 2.5x compared with the original threaded implementation, and a 65x speedup over the flat MPI version which is limited to a single process due to memory constraints

PROGRESS:

SUPER Institute collaboration to integrate OpenMP parallelism
- Native mode optimization to prepare for next-generation NERSC8 Cori
- Threading is essential to exploit full capability of MIC architecture

Performance of triples part of CCSD(T) improved 65x over original flat MPI implementation
- Flat MPI constrained to single process because of memory limitation

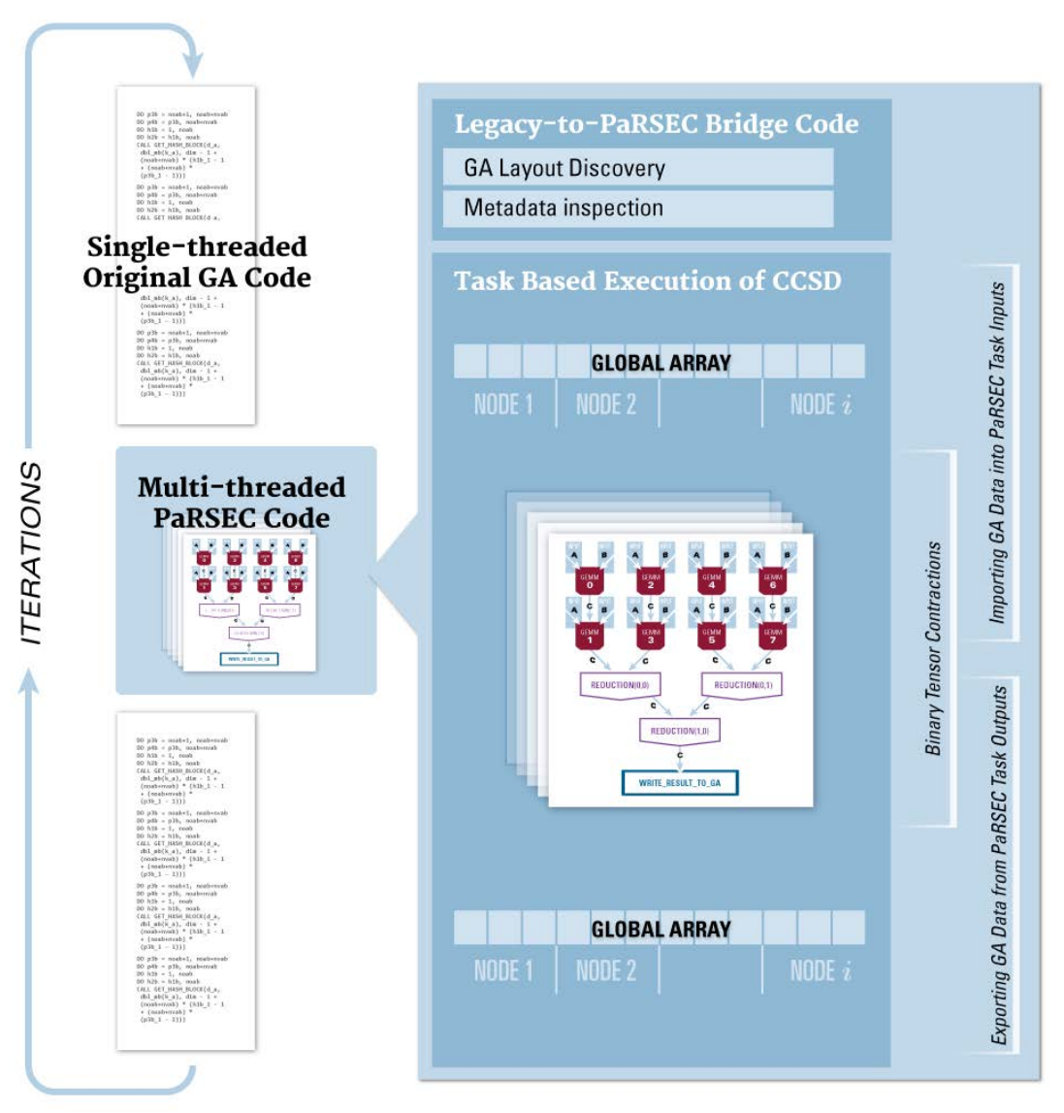
Performance of Fock matrix construction improved 1.64x over original flat MPI
- Flat MPI constrained to 60 MPI processes

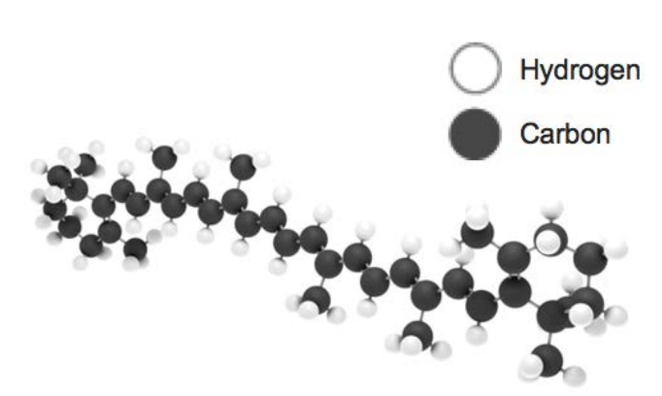
NWChem CCSD Data-flow Implementation

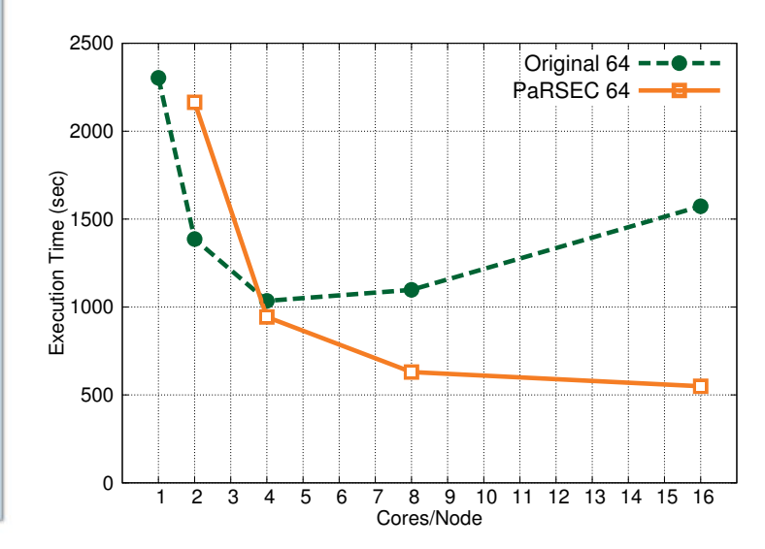
Anthony Danalis, Heike McCraw, George Bosilca
University of Tennessee

Objective

- ❖ Increase scalability and performance by porting CCSD of NWChem to a data-flow representation.


ITERATIONS
Single-threaded Original GA Code
Multi-threaded PaRSEC Code
Legacy-to-PaRSEC Bridge Code
GA Layout Discovery
Metadata inspection
Task Based Execution of CCSD
GLOBAL ARRAY
NODE 1
NODE 2
NODE i
Binary Tensor Contractions
Importing GA Data into PaRSEC Task Inputs
Exporting GA Data from PaRSEC Task Outputs





*Performance improvement of dataflow version (executing over PaRSEC) in comparison to original MPI code for entire CCSD. The modified code yields **2x higher performance** and keeps scaling until **all 16 cores of all 64 nodes have been utilized** in contrast with original code.*

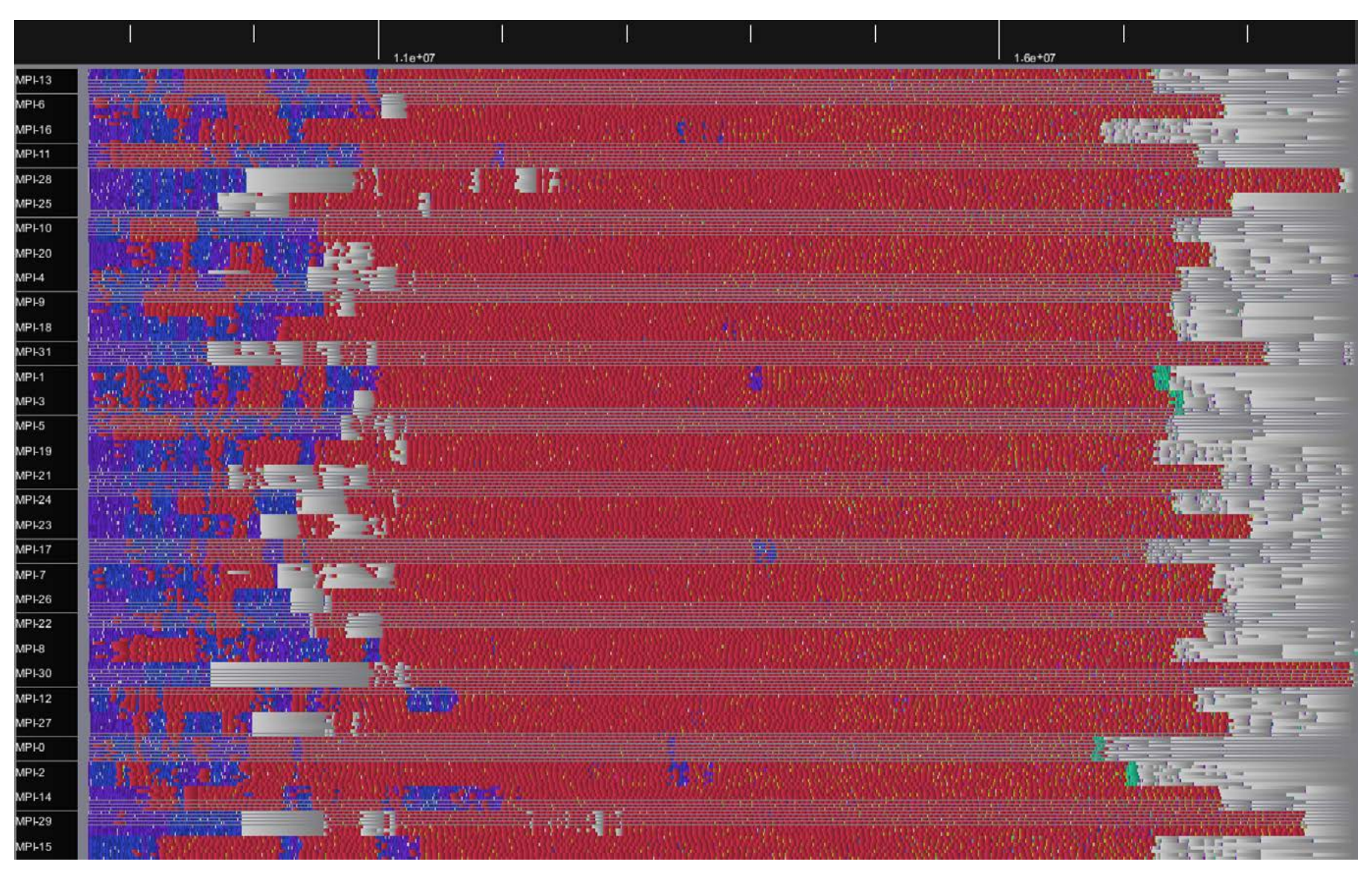
Accomplishments:

All time-consuming routines of NWChem's CCSD have been converted to a dataflow representation.

Modified version of CCSD was integrated into NWChem for seamless execution.

Integrated version of modified CCSD achieves more than 2x performance improvement.

Beyond performance gains, the dataflow version of NWChem can utilize PaRSEC's performance analysis tools that have *task* level granularity.



MPAS-Ocean

Abhinav Sarje, Samuel Williams, Leonid Oliker
Lawrence Berkeley National Lab

Douglas Jacobsen
Los Alamos National Lab

Kevin Huck, Allen Malony
University of Oregon

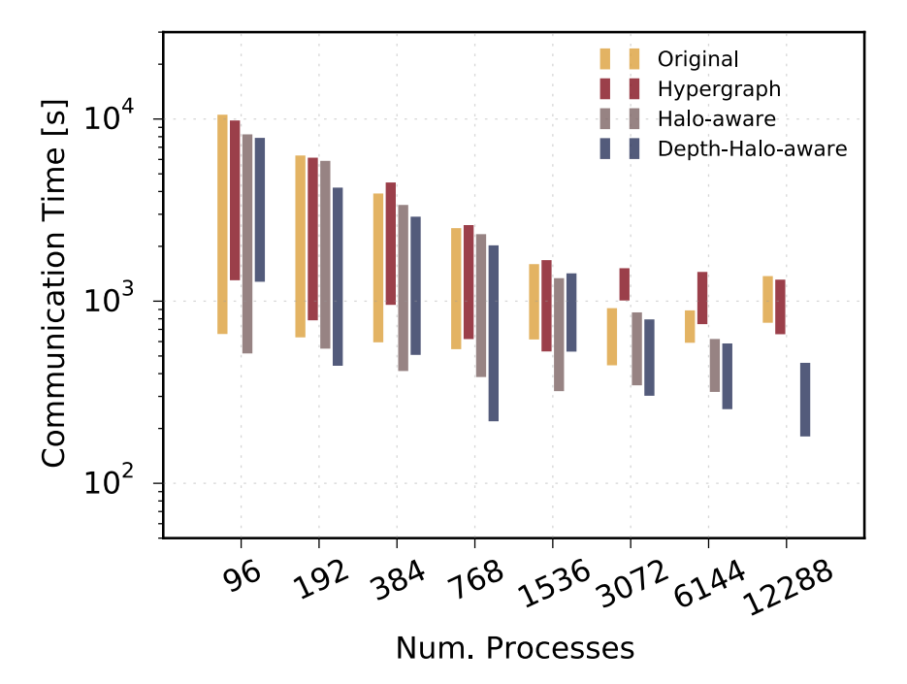
Sukhyun Song, Jeffrey Hollingsworth
University of Maryland

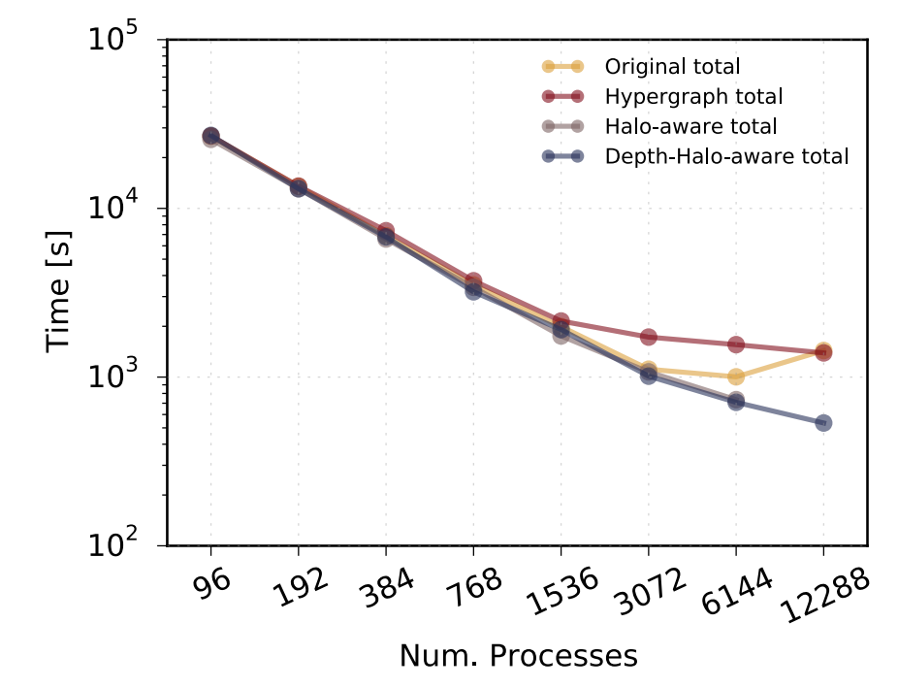
Background

- ❖ MPAS (Model for Prediction Across Scales) is a multi-scale climate modeling framework developed at LANL and NCAR.
- ❖ MPAS-Ocean core uses Voronoi tessellation based unstructured grids. It has the benefits of providing multi-resolution and quasi-uniform grid properties at the same time to facilitate better simulations.
- ❖ Unstructured grids have a negative impact on performance due to factors such as non-obvious domain decomposition, parallel load imbalance, unordered data and irregular memory access patterns.
- ❖ MPAS-Ocean utilizes deep halo regions on the grid. These magnify the load imbalance factor significantly.
- ❖ MPAS-Ocean performance was analyzed on DOE supercomputer, Edison, a Cray XC30 at NERSC.

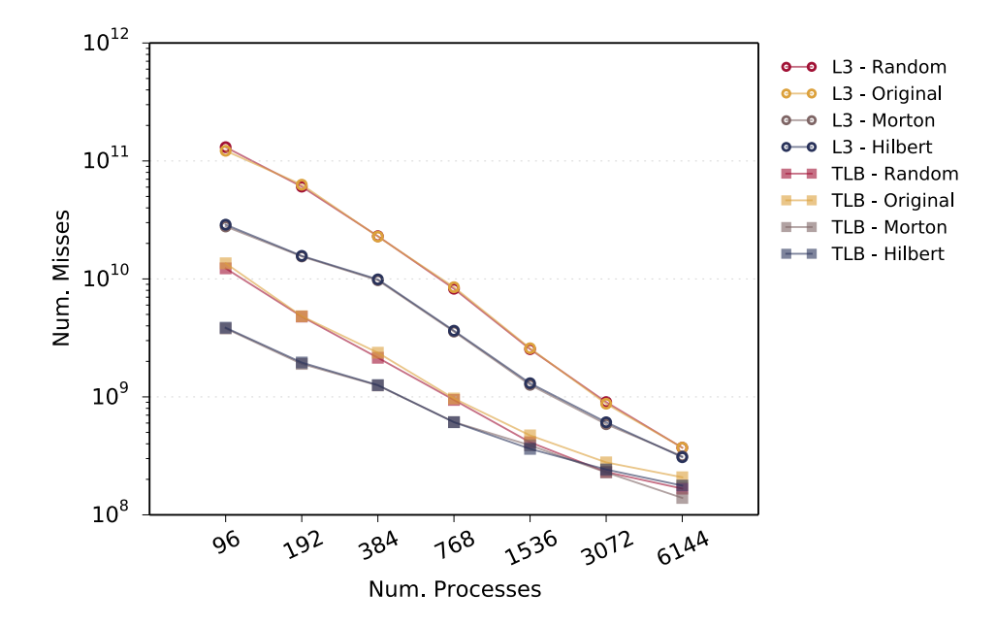
Performance Optimization and Results

- ❖ Developed a weighted halo-aware grid partitioning scheme based on iterative refinement of the partitions using halo information. Using hence generated grid partitioning resulted in improved scaling at high concurrencies.

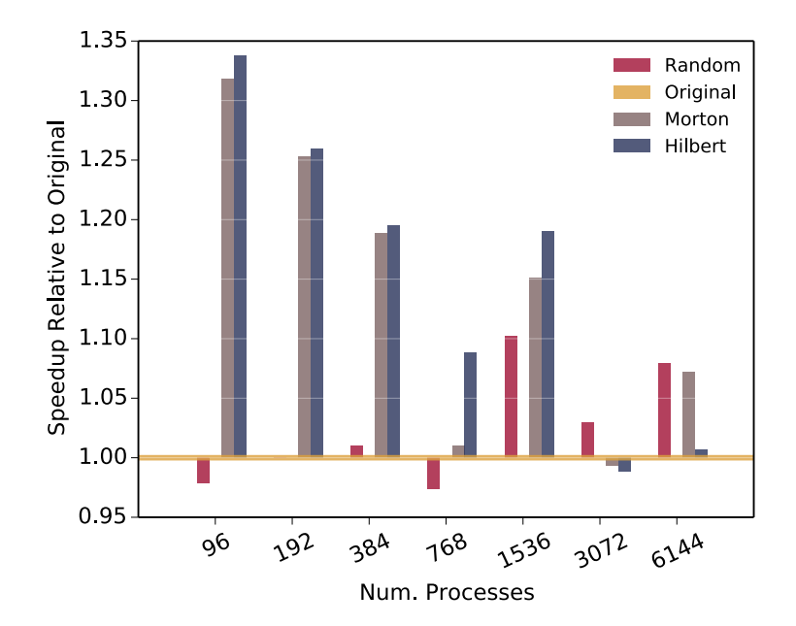




- ❖ Implemented Space Filling Curve based data reordering (Hilbert SFC, Morton ordering) for the grid elements to improve data locality.



Reduction in the number of cache misses using the SFC-based data reordering



Runtime speedup using SFC ordered data relative to the original data ordering

- ❖ Performance improvement due to the new partitioning scheme is more significant at high concurrencies due to better load balancing.
- ❖ Performance improvements due to data reordering is significant at low concurrencies, but the effect diminishes with increasing concurrency.
- ❖ With the new partitioning scheme and data reordering, we achieved overall performance speedup of up to 2.2x for the MPAS-Ocean core.
- ❖ Future work involves incorporation of multi-level OpenMP threading for improved parallelism and scaling. It also involves vectorization and porting the ocean core onto the many-core Intel Phi processors.

Analysis and Visualization of MPAS-Ocean Performance Data

Kevin Huck, (SUPER), Hank Childs (SDAV), Allen Malony (SUPER)
University of Oregon

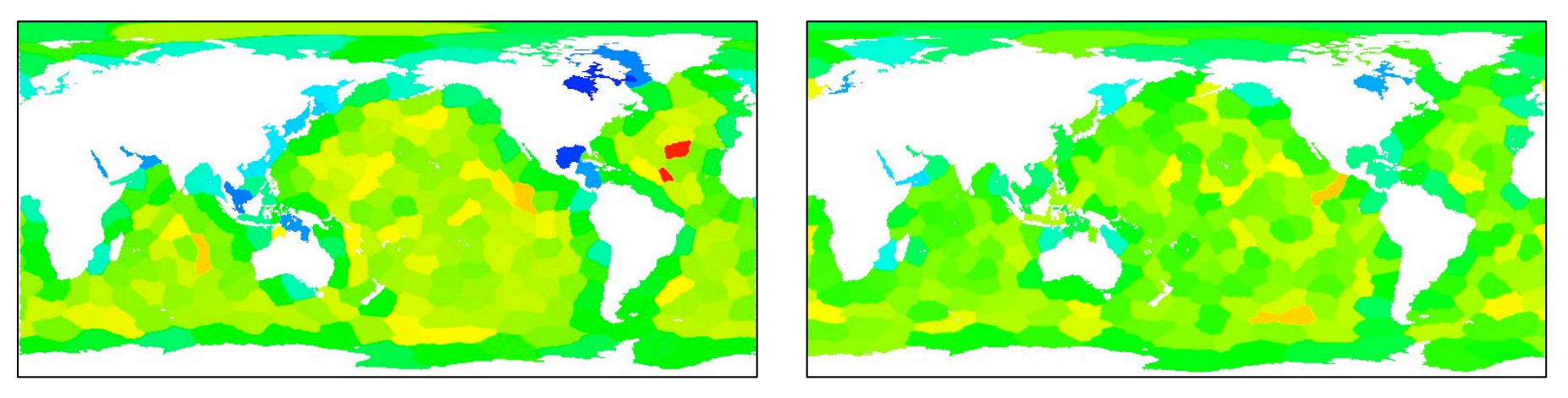
Background

- ❖ SUPER and SDAV collaboration.
- ❖ Objective was to Map TAU performance measurements to the MPAS-Ocean spatial domain to assist in optimization of partition strategies

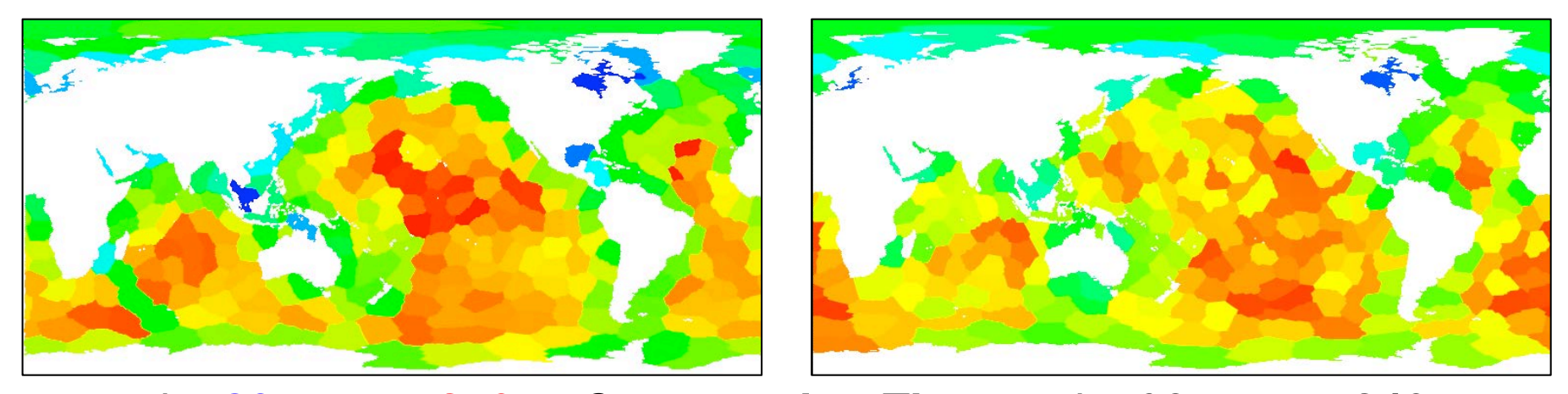
Progress and Accomplishments

- ❖ Demonstrated that the load imbalance problem is correlated with variability among partition block size due to relatively large halo regions
- ❖ Visualizations also show that vertical depth, coastlines and number of neighbors likely affect computation, communication times

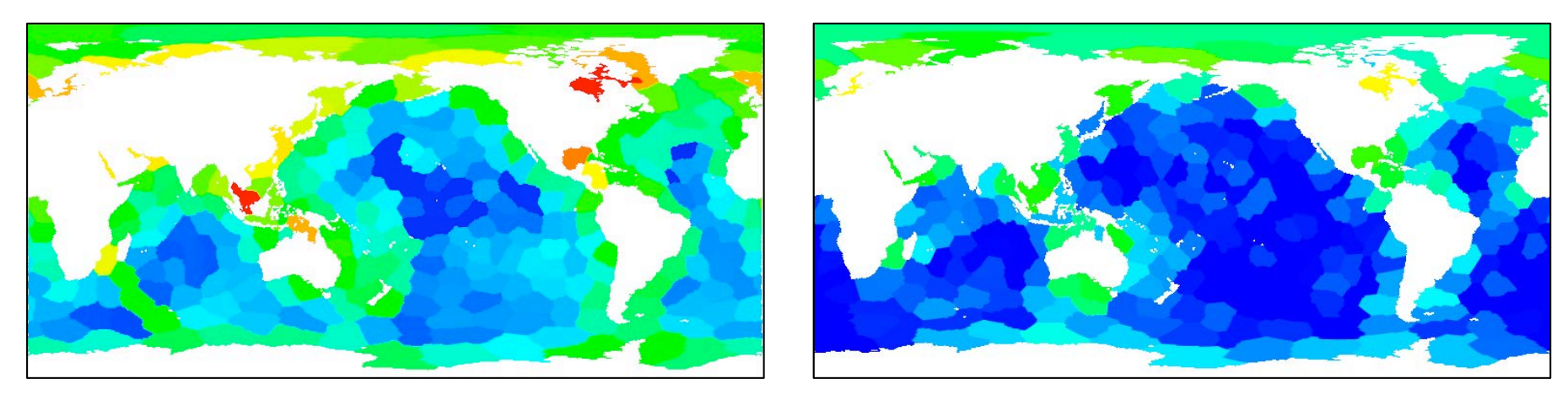
Original Partitions **Refined Partitions**



min: 473, max: 846 ***Total Cells per Block*** min: 535, max: **771**



min: 83s, max: 250s ***Computation Time*** min: 98s, max: **240**s



min: 27s, max: 190s ***MPI_Wait Time*** min: **9**s, max: **150**s

- ❖ *Hindsight* partition refinement using block+halo weights reduced mean MPI_Wait times by 40%, and overall execution time up to 15%
- ❖ Workshop publication: Huck et al., "Linking Performance Data into Scientific Visualization Tools", Visual Performance Analytics at SC'14

Impacts

- ❖ Successfully integrated TAU performance measurement data with application scientific data in VisIt
- ❖ Contributed directly to reduction in execution timer, for example up to 15% for 60km resolution case on 256 processors.

Awards of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under contract DE-AC02-06CH11357. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research also used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.