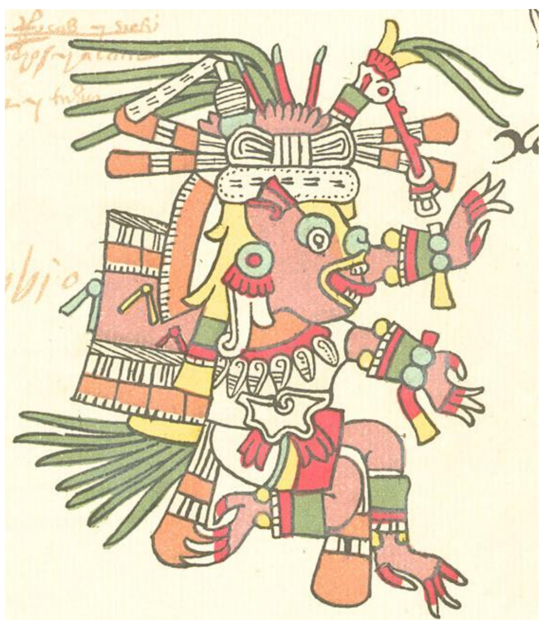
Activities and Impacts

U.S. DEPARTMENT OF ENERGY | Office of Science

Xolotl/Paraspace Performance Comparison

Phillip C. Roth
Oak Ridge National Lab

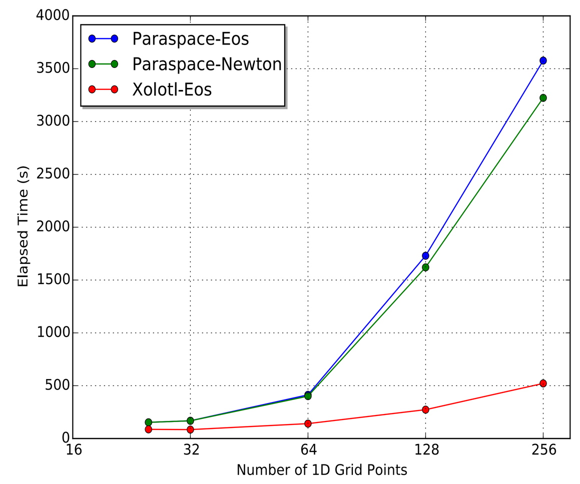


- Xolotl plasma surface interactions simulator
  - New code being developed as part of Plasma Surface Interactions FES SciDAC Partnership (PI: Brian Wirth)
  - Continuum model for solving cluster dynamics advection-diffusion-reaction equation with incident flux
  - Support for 1D, 2D, and 3D problems
  - 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 4×10^{25} He/m^2/s for 1×10^{-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

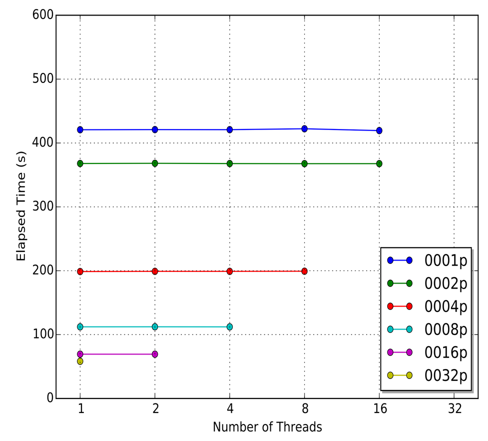
Time required to simulate full He retention in W100 problem on one Eos compute node

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

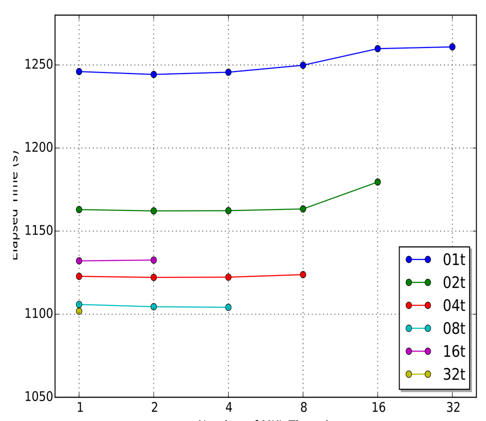


Time required to simulate full He retention in W100 problem on one Eos compute node with varying numbers of grid points

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



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



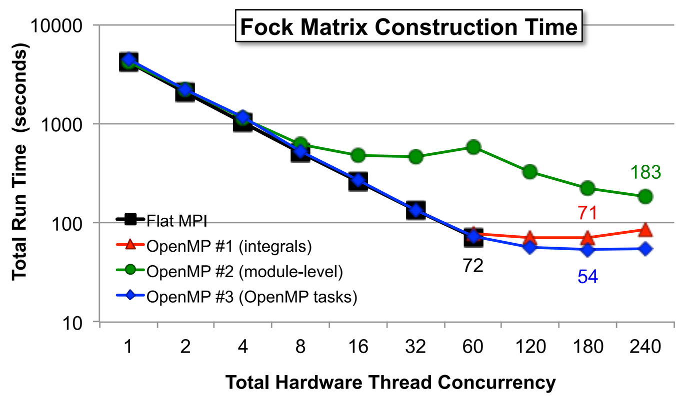
- Xolotl outperformed Paraspace on full problem, scaled well when increasing problem size, but…
  - …Paraspace computes more time steps at higher accuracy (hypothesized)
  - Currently testing relaxed Paraspace solver tolerances that reduce number of time steps but may also reduce accuracy
- Working on 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

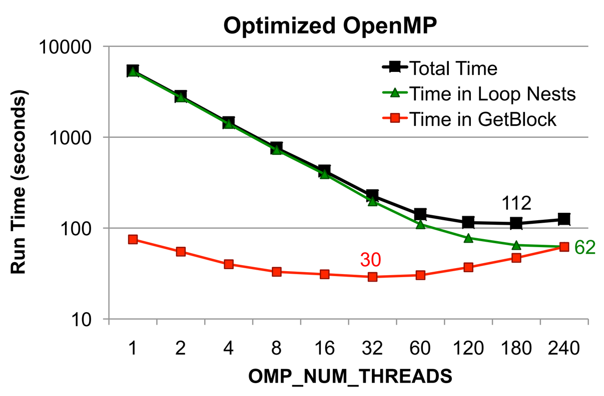
Hongzhang Shan, Bert de Jong, Lenny Oliker,, Samuel Williams
Lawrence Berkeley National Lab

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

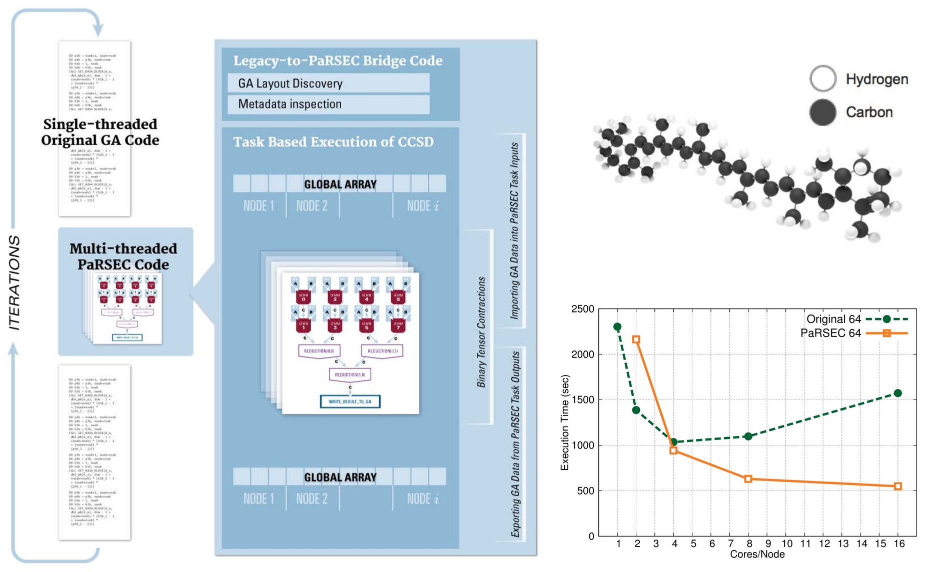
- Flat MPI constrained to 60 MPI processes

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.



*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.*

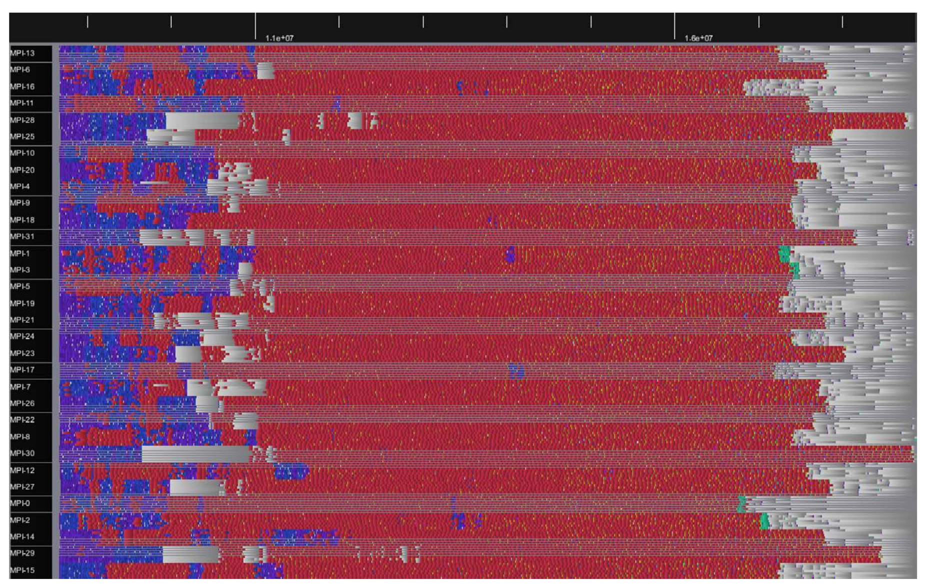
Accomplishments:

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

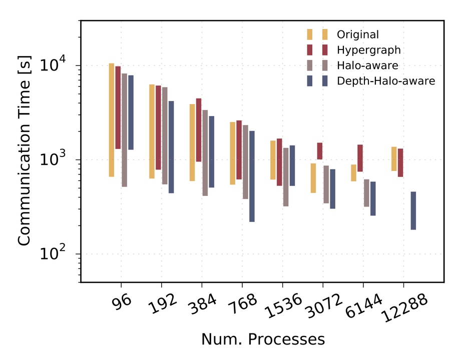
Abhinav Sarje, Samuel Williams, Leonid Oliker
Lawrence Berkeley National Lab
Douglas Jacobsen
Los Alamos National Lab
Kevin Huck, Allen Malony
University of Oregon
Sukhyun Song, Jeffrey Hollingsworth
University of Maryland

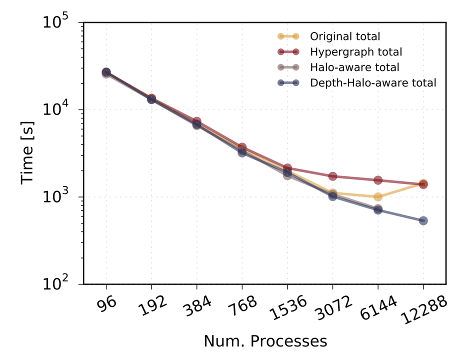
Background

- 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.

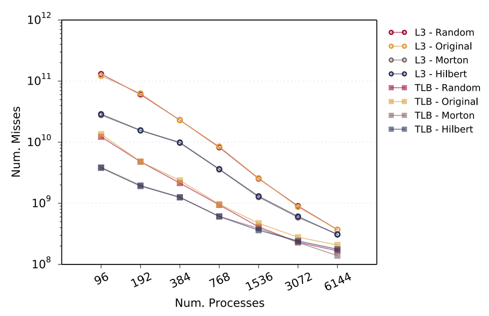
Performance Optimization and Results

- 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.

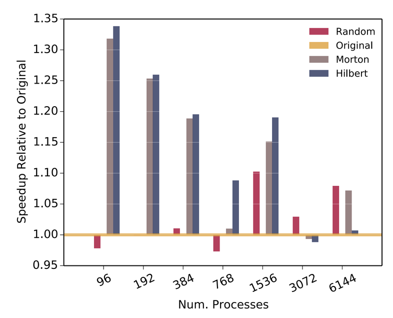




- Implemented Space Filling Curve based data reordering (Hilbert SFC, Morton ordering) for the grid elements to improve data locality.



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

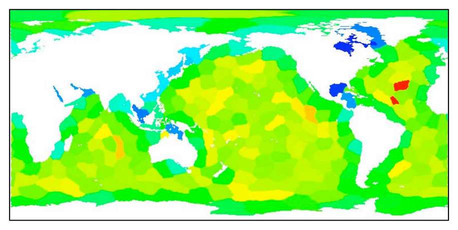
Background

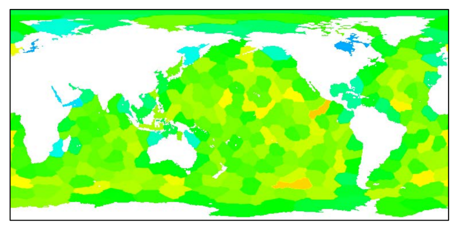
- SUPER and SDAV collaboration.
- Objective was to Map TAU performance measurements to the MPAS-Ocean spatial domain to assist in optimization of partition strategies

Progress and Accomplishments

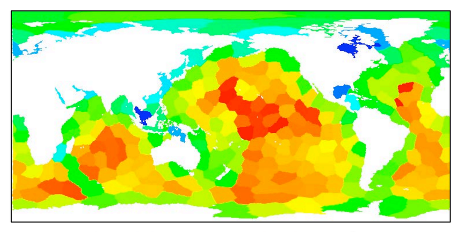
- Demonstrated that the load imbalance problem is correlated with variability among partition block size due to relatively large halo regions
- Visualizations also show that vertical depth, coastlines and number of neighbors likely affect computation, communication times

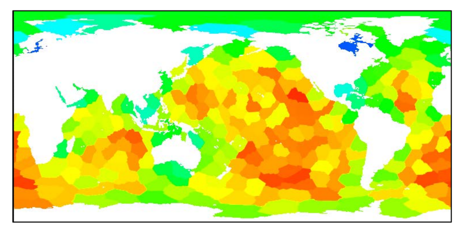
Original Partitions | **Refined Partitions**



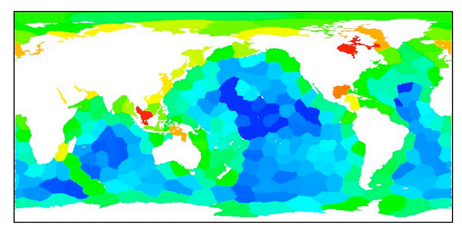


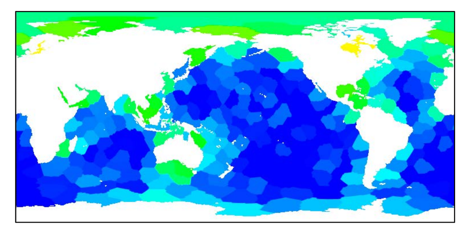
min: 473, max: 846 ***Total Cells per Block*** min: 535, max: 771





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





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

- *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.