Performance focused developments within MPAS-Ocean

Doug Jacobsen[1], Todd Ringler[1], Mark Petersen[1], Leonid Oliker[2], Sam Williams[2],
Abhinav Sarje[2], Kevin Huck[3], Pat Worley[4]

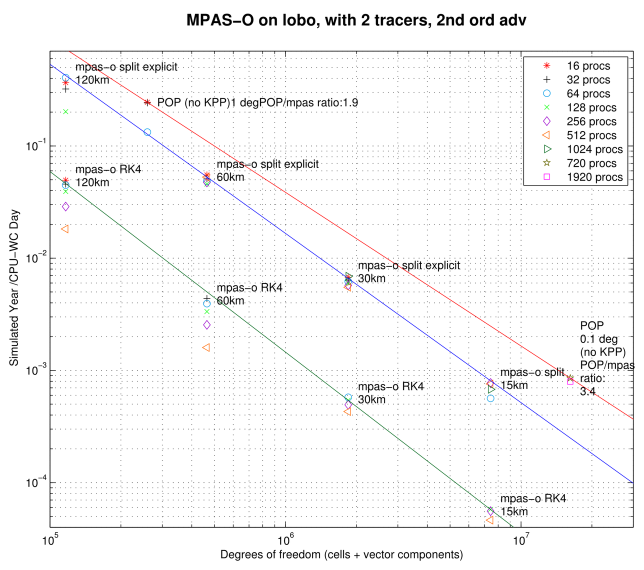
[1] - Los Alamos National Laboratory
[2] - Lawrence Berkeley National Laboratory
[3] - University of Oregon
[4] - Oak Ridge National Laboratory

Abstract

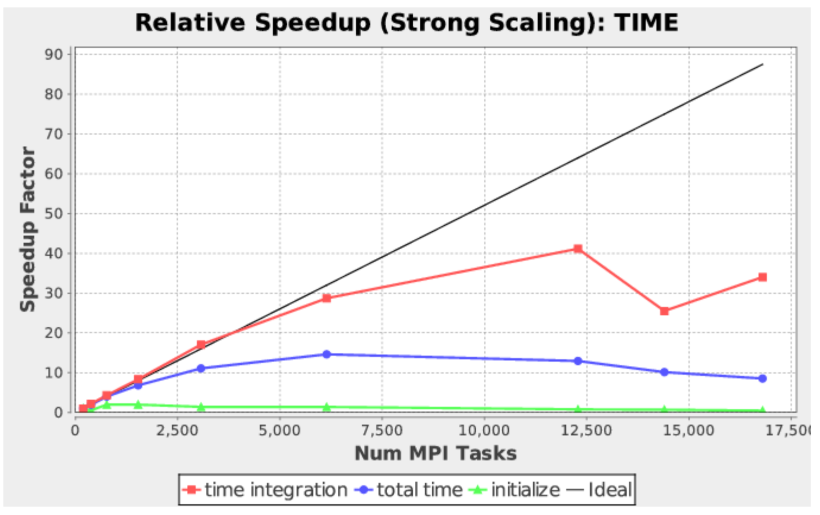
The Model for Prediction Across Scales is a modeling framework for the rapid prototyping and development of dynamical cores intended for climate simulations. It is built upon an unstructured meshing framework that comes with its own unique performance challenges. Over the past few years, the development team for MPAS-Ocean (the ocean dynamical core within the MPAS framework) has been interacting closely with the SciDAC SUPER Institute to improve the performance of the model. Throughout the collaboration SUPER has provided assistance for general profiling, targeting areas of the model for improvement, and exploring alternative methods of decomposing the horizontal mesh. Planned future performance related advancements to MPAS include aggregated communications, custom non-blocking communications, multiple levels of parallelization (including threading), and advanced decompositions.

Effect of unstructured horizontal mesh

Traditionally ocean models have been implemented utilizing structured memory layouts, this provides uniform memory access patterns for the compiler. However, this limits the physical layout of horizontal mesh elements. MPAS utilizes a fully unstructured horizontal mesh, which allows more flexibility in the description of the horizontal mesh locations, but also has an impact on performance due to indirect addressing.



Performance of MPAS-Ocean (v1.0) compared to POP. Relative to degrees of freedom in horizontal mesh.



Strong scaling of MPAS-Ocean (v1.0) on Hopper (NERSC).

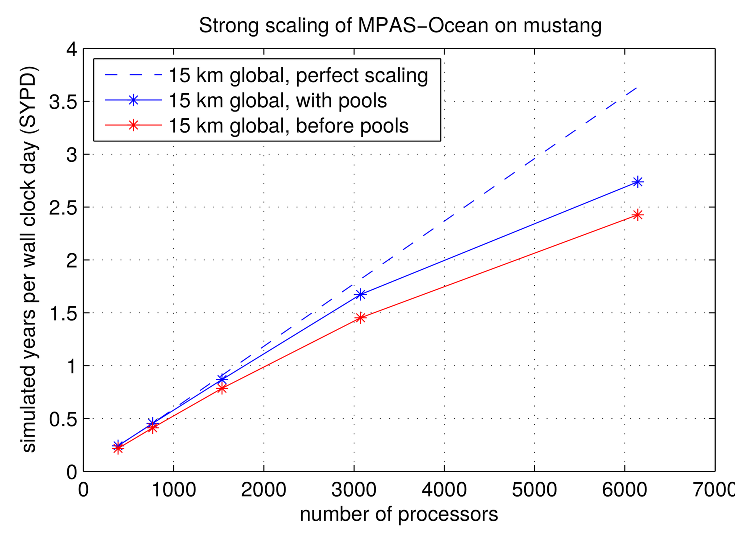
Data Structure Rewrite

The data structures within MPAS previously used several levels of indirection via pointers. i.e.

```
domain % block % state % time_levs(1) % state % normalVelocity % array(:,:)
```

Some compilers are not capable of vectorizing the lowest level array in this type of instruction. In order to improve vectorization of MPAS-Ocean, the data structures were rewritten utilizing hash tables. In addition to performance based reasons, this modification made the code more flexible. The array access from above would now look like:

```
real (kind=RKIND), pointer, dimension(:,:) :: normalVelocity
call mpas_pool_get_array(statePool, 'normalVelocity' ,normalVelocity, 1)
```



Scaling of MPAS-Ocean before and after data structure rewrite ("pools") on Mustang. A 10-15% performance improvement is shown

Collaborations with SUPER

For the past two years, MPAS-Ocean developers have been closely collaborating with the SciDAC institute "SUPER" in order to target specific performance optimizations within MPAS-Ocean. SUPER has also helped MPAS-Ocean developers track performance of the model via usage of profiling tools such as TAU[1] (U of Oregon).

A list of performance explorations to date have been:

- Profiling of MPAS-Ocean via TAU[1]
- Implementations of OpenMP in MPAS-Ocean, and profiling
- Hierarchical decompositions
- Element Reordering

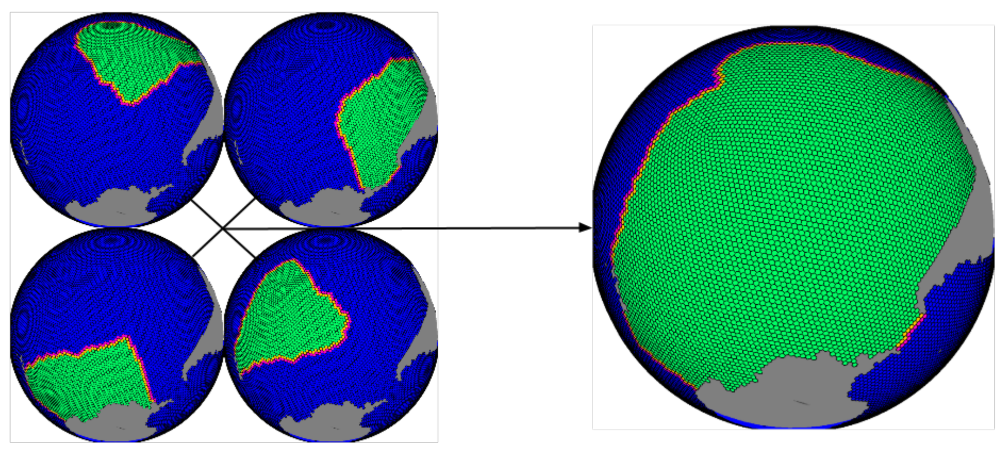
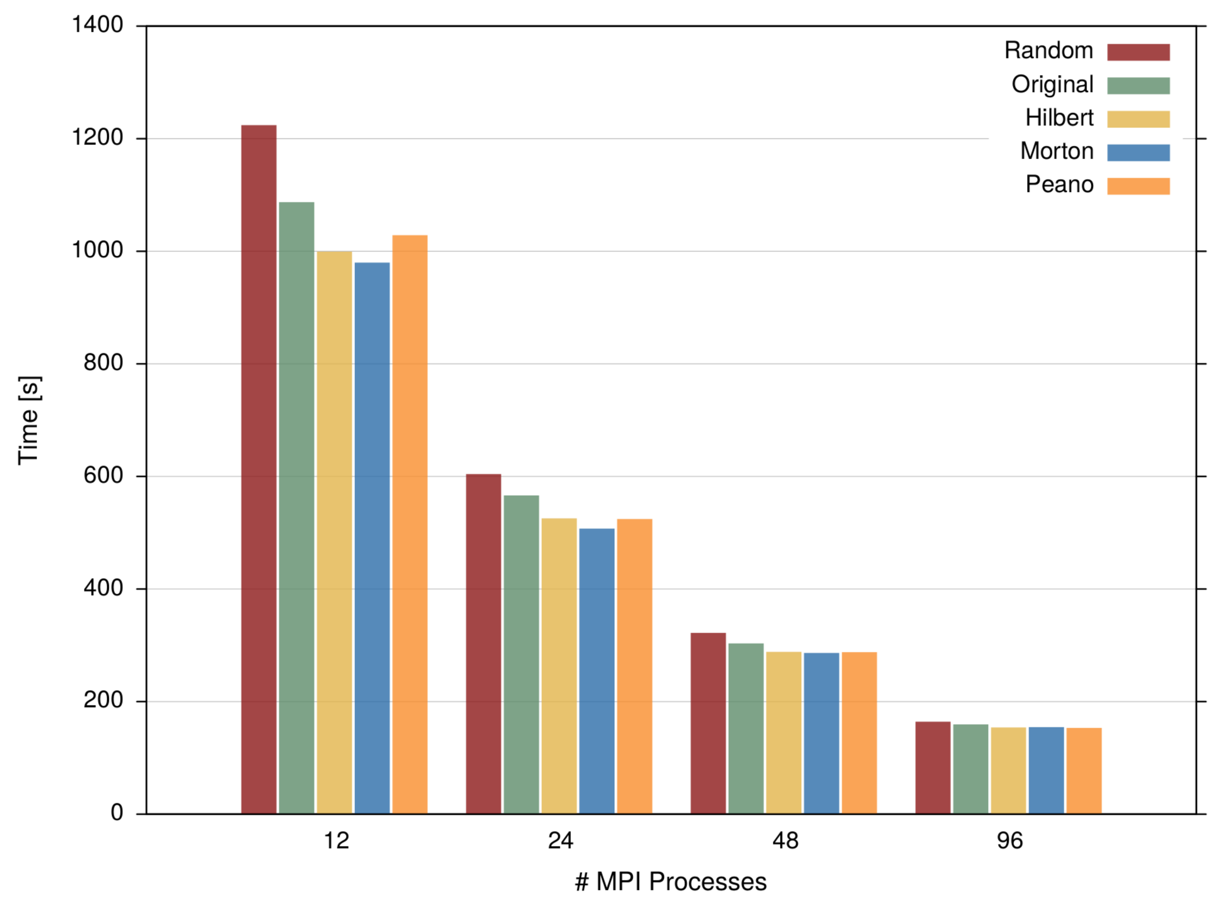


Image showing a hierarchical decomposition of a MPAS-Ocean horizontal mesh. Four images on the left represent intra-processor decomposition while image on the right shows a single processor's computational domain.



Comparison of different space filling curve implementations to reorder MPAS-Ocean horizontal cells. Simulations performed on Edison.

Future Performance Tasks

Future planned efforts with the SUPER instituent include:

- Final implementation of OpenMP
- Continued profiling of MPAS-Ocean
- Profiling of hierarchical decompositions
- Final implementation of element reordering schemes
- Hypergraph partitioning
- Aggregated communications

References

[1] "TAU: The TAU Parallel Performance System", S. Shende and A. D. Malony. International Journal of High Performance Computing Applications, Volume 20 Number 2 Summer 2006. Pages 287-311.