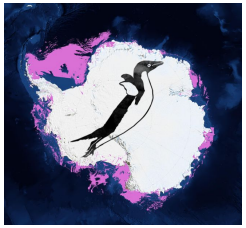


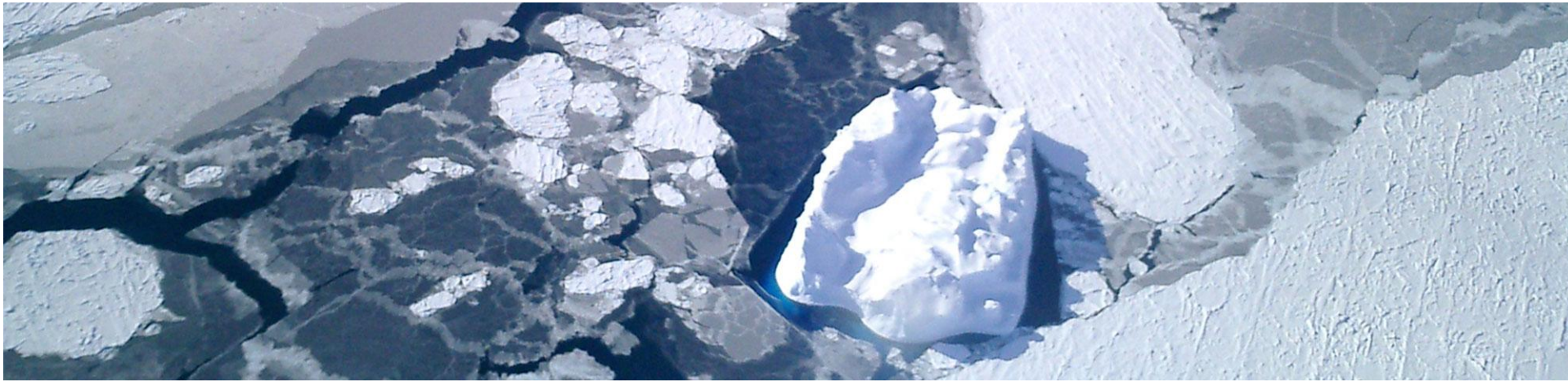
XSEDE
Extreme Science and Engineering Discovery Environment

NSF

Team OBLIMAP

XSEDE16 Polar Compute Hackathon

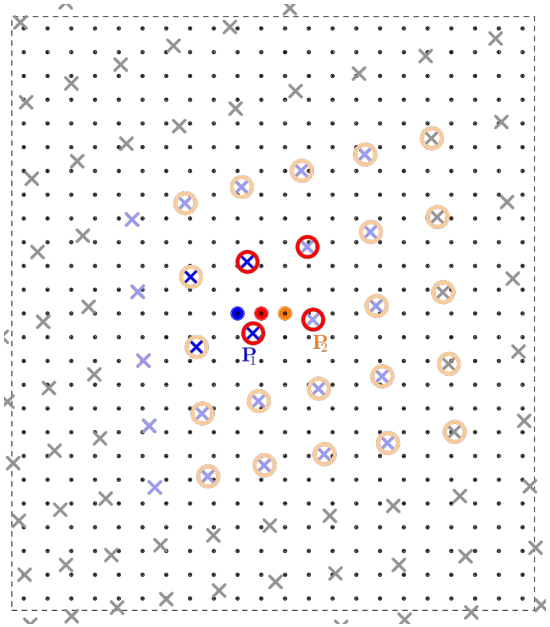
Thomas Reerink, Erin Hodgess, Cyrus Proctor

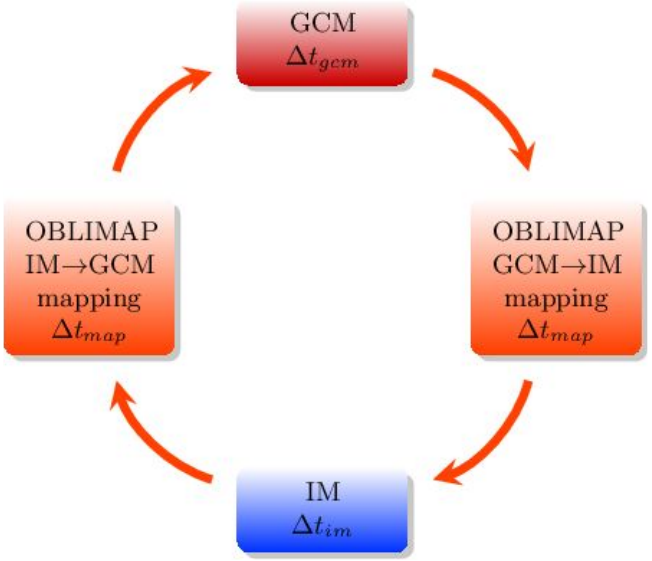


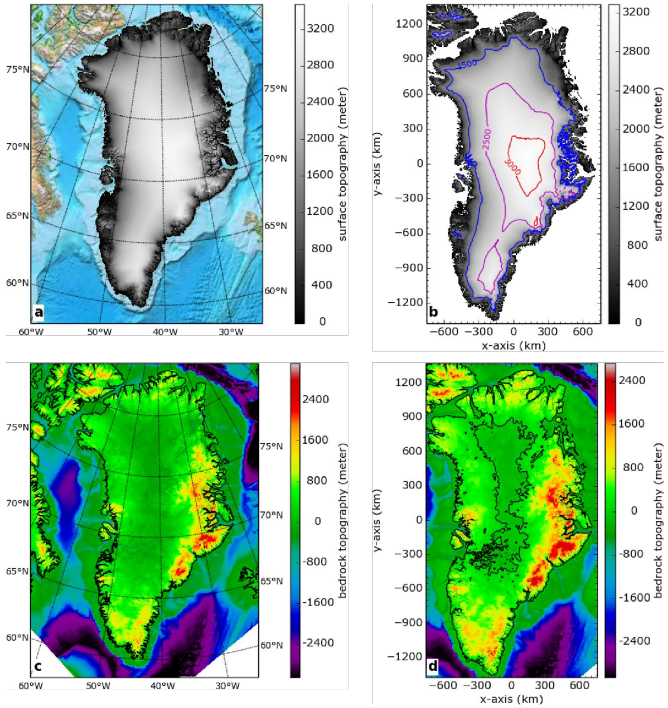
July 17 - 19, 2016

http://polar-computing.org

OBLIMAP: Overview & Objectives





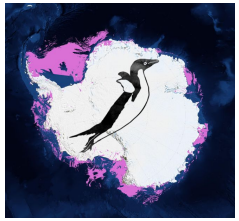


Issue

OBLIMAP's scanning phase to interpolate between GCM and IM grids takes the most computational time

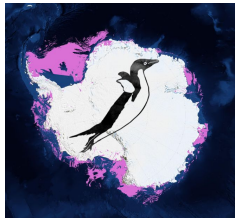
Goal

Parallelize aspects of the scanning phase to speed up interpolation time



OBLIMAP Computational Challenge

- Explore the scanning phase of OBLIMAP and determine a parallelization strategy
- Identify portions of OBLIMAP -- a serial code -- and refactor to become MPI-aware
- Edit Makefiles to run on Stampede and Comet XSEDE resources
- Load balance the work that each process is responsible for
- Modify file input/output to remedy parallel contention
- Communicate important variables/data structures between MPI processes
- Verify results and benchmark via strong scaling parallel efficiency
- Keep track of all the changes in an unfamiliar version control system

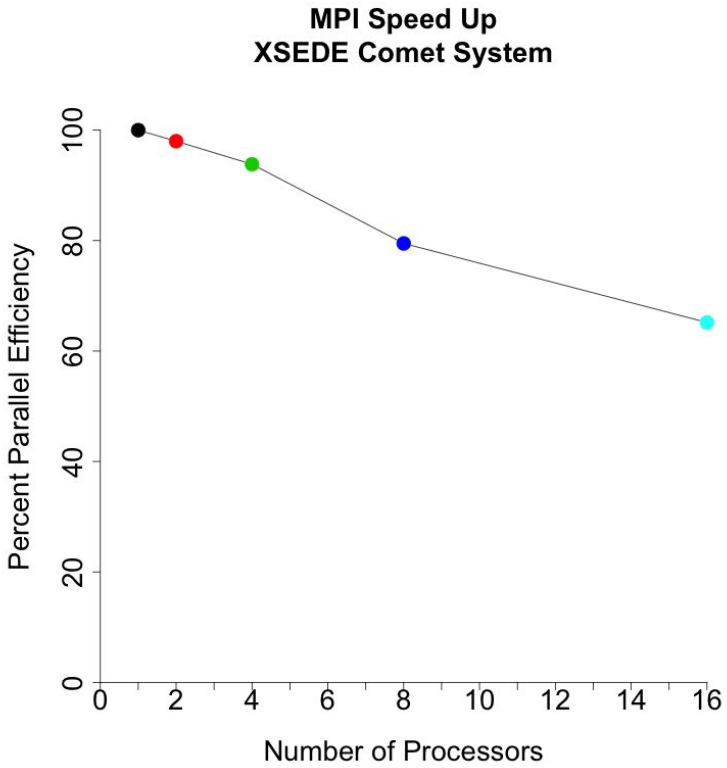


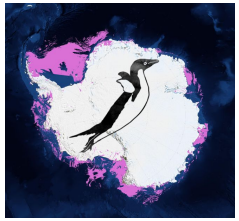
OBLIMAP Team Achievements

- The scanning phase is made of a quadruple-nested do loop structure
  - Outermost loop is amenable one-dimensional domain decomposition
- Team was able to insert necessary MPI calls, compile, build, and run an MPI-aware version locally as well as on Stampede and Comet systems
- Outer do loop was partitioned to provide equal amounts of work to each process
- Intermediate results were written to MPI rank-dependent filenames
- Parallel results were verified against prior serial work -- and are identical!
- Preliminary strong scaling parallel efficiencies were generated on Comet

Speedup Measures

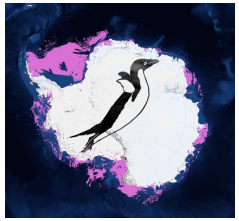
- We used Comet to compare the percent of parallel efficiency against the serial code for the GCM to IM scanning section
- The relationship is:
  - Y = (Serial/(Num of Proc*Parallel Time)) *100
- At N=2, we are at 97%, which is excellent.
- Even at 16 processors, we are still at 65 percent parallel efficiency; that is, the speedup is about 65 percent.
- This is currently using one node





OBLIMAP: Future Work

- Parallelize similar branches of the scanning phase (2 of 4 currently complete)
- Use the rank-dependent files in further portions of OBLIMAP
- Explore more sophisticated possibilities of parallel file I/O
- Look into further domain decomposition strategies within the scanning phase
- Scale up number of MPI processes as I/O and further parallelization is implemented

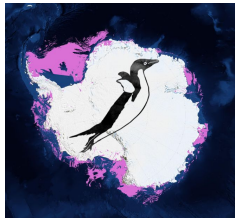


Any highlights? What worked? What didn't?

Erin: I enjoyed the speedup section. The results were very impressive, particularly in the initial stages. The problem data that we had was small. It will be most interesting to see how this plays out with a large data set. Something we did not include in the presentation is that with a small problem, using 2 nodes impacts the speedup negatively.

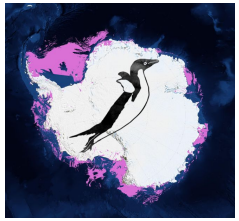
Thomas: The close collaboration, it accelerated my understanding of MPI significantly: Sitting at the same desk and working on a real world issue.

Cyrus: I'd say getting a chance to delve into a completely new domain of science that I've never had the privilege of working on. Also, system calls are tricky!



Biggest insight/lesson learnt?

- Even though the problem seemed extremely well-suited for a parallel MPI implementation initially, much more code modifications were required
- Implementing a simple MPI domain decomposition with the minimum MPI communication
- Writing I/O in the core of the parallel portion of the code complicates matters significantly



Conclusions

- Staring out: OBLIMAP had not previously used XSEDE resources and ran serially
- Afterwards: OBLIMAP has a basic framework for MPI-aware domain decomposition
- We were able to use Stampede and Comet for building, testing, and running
  - While in development we used our local laptops and 1 node interactive development sessions
- With more development, OBLIMAP will be capable of using many nodes at once