Use and Performance of a Parallel Ocean/Ice Circulation Model on Turing

Mike Dinniman
Center for Coastal Physical Oceanography
Old Dominion University

October 15, 2014
We use a structured grid ocean/ice circulation open source model (ROMS) in order to study processes such as ocean forced melting of the Antarctic Ice Sheet.

Simulation of the Amundsen Sea sector of Antarctica showing ocean temperature at 300 meters.

(courtesy Pierre St-Laurent)
Introduction

Because of the horizontal resolution needed to properly model certain circulation features, we often need a very large grid, e.g.

\[ 2120 \text{ pts (x-direction)} \times 2120 \text{ pts (y-direction)} \times 32 \text{ pts (z-direction)} \times 28 \text{ intermediate values needed} \times 8 \text{ bytes/variable} = 30 \text{ GB for just one state variable (and we have several)} \]

We also often want to model events that occur over decades, but for the model to be stable, we need very short (~ a few minutes) simulated time steps.

Not only do we need to run the model on several processors for speed, but it often requires more memory than available on a single node Distributed memory processing.
ROMS and MPI

The grid is split up (domain decomposition) horizontally

Each core works on its own section of the model grid, but information is needed from adjacent grids at the end of each time step.

Code is written in Fortran 90 and data is passed between cores using calls to the MPI (Message Passing Interface) library.

MPI is a standardized library (works w/ C, C++, F77, Fortran90, Fortran 2003, Fortran 2008, ?) with many implementations.

courtesy Kate Hedstrom
OpenMPI on Turing

Turing has several implementations of MPI, but we’ve been using OpenMPI (an open source implementation of the MPI standard) with the Open64 compiler (an open source compiler of C, C++ and Fortran on Linux machines)

Need to load the Open64 and OpenMPI modules to use on Turing

   module load open64/4.5.2.1
   module load openmpi/open64/64/1.6.4

Compile with “mpif90” (which is just a wrapper to the Open64 compiler built to link with the OpenMPI libraries)

Run on Turing on the openmpi_ib environment (-pe openmpi_ib in your submit script) using the mpiexec command
Scales well (linearly) up to at least 192 cores for this particular grid.

Still get performance increase up to ~350 cores (maybe more?)

Disk I/O matters at high number of cores (not using parallel I/O).

Scaling depends on grid size (not shown - larger problem seems to scale well up to at least 512 cores).
Summary

• It can be very complicated to re-write existing code to use distributed memory with MPI, but if the code is already setup to use MPI, it’s not hard to get it running on Turing.

• The OpenMPI implementation of the MPI standard on Turing allows us to run on many more cores than would be possible with a shared memory programing model.

• We are currently running problems that scale well up to several hundred cores, but scalability depends on problem size.