# Programming Assignment 5 - hierarchical parallelism

**Early Due Date (+5%): Fri. 11/22, 11:59pm**
**Final Due Date: Tue. 11/26, 11:59pm**

## Background

Distributed computing both opens up the possibility of an entire network of computers collaborating simultaneously on an single application while also enabling multiple levels of concurrent parallelism.

Using MVAPICH2, an open source standard MPI implementation, combined with global shared memory threading techniques we will implement a hierarchical parallel version of simplified AMR.

## Assignment

Create an MPI version of your SAMR program according to the following guidelines:

- use the global communicator
- use a total of 5 MPI processes
- Your rank 0 MPI process will be your master MPI process. This process will:
  - perform all input and output functions
  - initialize DSVs
  - after each iteration of updated DSV computation collect all updated DSVs and perform the convergence condition test
  - print convergence results (convergence iterations, max and min DSV at convergence, values used for affect rate and epsilon)
  - collect and report appropriate timing information for comparison with your other SAMR implementations.
- ranks 1 - 4 will perform parallel DSV update computations
  - using pthreads or OMP, create (request) one thread for each node core
  - design and implement an appropriate workload distribution
  - for each convergence iteration:
    * receive current DSVs from the rank 0 master MPI process
    * compute updated DSVs in parallel
    * communicate updated DSVs to master MPI process

Provide a short (2 -3 page) report which:

- describes the workload distribution used for this lab
- provides summary comparative runtime results for all implementations of SAMR (labs 1 - 5)
- your descriptions and analysis of any unexpected results
- a brief summary of the applicability of the various parallelization APIs discussed this semester

## Input Data

Use data file testgrid_400_12206 from previous labs 1 through 3.

## Testing and Submission

Please note that OSC run queues, including the interactive queues, may get exceptionally long at this time of the semester. Plan accordingly. Please conduct your MPI tests on the Owens cluster.

Follow the testing and submission guidelines for previous labs, using directory “lab5” for this assignment.

Run your program using 5 nodes (request all cores for all nodes so that you have dedicated processors), one for the master MPI process and one for each remote MPI process.

A common and effective MPI development strategy is to design the communication paths and protocols you will use for your application first, and then implement a “skeleton” prototype that simply performs the communications using known values (using “stub” routines as placeholders for computational subroutines). Once this framework is verified to be working correctly, the actual computational routines are inserted into the communications framework.

- Create a directory "lab5". Within this directory, place:
  - all source code files;
  - makefile
    - your program should build with the command “make” with no parameters.
    - name your executable “lab5_mpi.”
- submit your report in .pdf format via Carmen.

## MPI on OSC

MVAPICH2, an open source standard MPI implementation, is installed and available on the Owens cluster. Use the *module list* command to verify it is loaded, and *module load mvapich2* if necessary. To allocate a proper node and begin an interactive computing session, use: **qsub -I -l walltime=0:59:00 -l nodes=1:ppn=28** (qsub -"capital eye" -"little ell" walltime . . . -"little ell" nodes . . . ).

The qsub command often obtains an interactive session within a few minutes. Note, however, that OSC is a shared resource, and your wait will very with current system load. The load on Owens is anticipated to increase significantly near the end of the semester. Please plan accordingly. We are guests on the OSC cluster, so please practice good citizenship. You can compile your MPI programs on the "login nodes." Only start a qsub interactive session **when you are ready to run your program.** Please "exit" from qsub once your tests are complete. If you need to make significant revisions to your source, exit qsub, edit as required, and then start a new qsub session to re-test. Also, do not forget we have access to the debug and batch queues as well.

The MPI compiler is *mpicc*, which uses the same options as the gcc C compiler.

To execute MPI programs on Owens first start an interactive qsub session, and then use *mpirun -np 5 ./lab5_mpi*. The "-np 5" option will specify the 5 processes (an MPI master process and the four remote MPI computational processes) required for this assignment.