CHPC Student Cluster Competition 2022

WRF

The **W**eather **R**esearch and **F**orecasting model is a state-of-the-art mesoscale **N**umerical **W**eather **P**rediction (NWP) system designed for both atmospheric research and operational forecasting applications. It features two dynamical cores, a data assimilation system, and a software architecture supporting parallel computation and system extensibility. The model serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers.

Obtain copies of WRF and its Pre-Processing System source codes from GitHub <https://github.com/wrf-model/WRF> and <https://github.com/wrf-model/WPS> (the benchmarks have been tested against **WRF 4.2**). Alternatively copies of the **WRF 4.0** tarballs have also been included in your competition folder.

Configuration, building and installation instructions can be found at <https://www2.mmm.ucar.edu/wrf/OnLineTutorial/> , which detail the following dependencies to be installed or made available in your environment:

* Your choice of **C** and **Fortran Compilers** and implementation of **MPI**
* **HDF5** library required for **NetCDF4** compression support. (Tested with version 1.10.5)
* Optionally you can make use of **Parallel HDF5**, which will process file I/O in parallel, as opposed to serially
* **NetCDF** library is used for large network file I/O with gridded model data. (Tested with version 4.6.3)
* Optionally you can make use of **Parallel HDF5** and netCDF libraries, which will process network file I/O in parallel, as opposed to serially
* If these are not already installed on your system, you may additionally require **Jasper**, **libpng** and **zlib**.

Take care to ensure that you are working within a consistent environment. Failure to manage your environment, libraries and dependencies properly will impede your team from being able to successfully build and compile the application. You can compile for basic nesting.

During the compilation process you will be presented with a choice of options to select for a column (type of compiler and architecture) and a row (type of parallel build):

* The first column is for serial (single processor) builds.
* The second column is OpenMP (threaded, shared-memory), builds with up to 40 maximum processes.
* The third and fourth columns are for (MPI only) and (OpenMP + MPI)

In an analogous way to the optimization experiment from the SWIFT benchmark, where you optimized for the most efficient communication pattern between MPI processes, with each containing multiple POSIX (pThreads) utilizing shared memory. In this experiment however, you will be required to find an optimal mix of MPI process and OpenMP threads, (or your configuration may perform best without OpenMP threads).

Clean build the executable on multiple processors, and for debugging purposes you can usefully redirect and store standard out and standard err to a log file.

$ ./compile - j N em\_real >& build\_wrf.log

$ ./configure

Once WRF has finished building, you must compile and configure WPS, and from within this subdirectory, you may need to create the following symbolic link:

$ ln –sf ungrib/Variable\_Tables/Vtable.GFS Vtable

# Benchmark 1: Single Domain Case

Extract the GEOG and DATA into your WRF/GEOG and WRF/DATA folders, respectively, then copy namelists.wps to WRF/WPS. From your WPS folder, link the weather data files and build the geographical land usage interpolation and metrological data:

$ ./link\_grib.csh ../DATA/\*

$ ./ungrib.exe >& ungrib.log

$ ./geogrid.exe >& geogrid.log

$ ./metgrid.exe >& metgrid.log

Before you can run the benchmark, you must copy the benchmark input files namelist.input into the run/ benchmark subdirectory, create symbolic links to the metrological input data and execute the last interpolation step. From the run/ benchmark subdirectory:

$ ln –sf PATH/TO/met\_em.d01\*

$ ./real.exe >& real.log

This will generate the boundary condition file for the (outer) domain wrfbdy\_d01, and the initial conditions for the domain wrfinput\_d01, (there will be an additional input file in the case for nested domains). There will be a wrf.exe binary and a symbolic link to them under the main/ and run/ folders, respectively.

You must determine the optimum ratio of OpenMP Threads to MPI Ranks, and in this regard useful environment variables to manipulate include OMP\_STACKSIZE and OMP\_NUM\_THREADS. Configure an appropriate SLURM batch script and reroute the output to an appropriately named logfile.

time mpirun –genv OMP\_NUM\_THREADS <OpenMP Threads> \

-np <Num Nodes><MPI Ranks> \

2>&1 | tee singleDomain.log

You will notice rsl.error.xxxx and rsl.out.xxxx files, corresponding to each of the xxxx MPI ranks. It may be useful to monitor or tail the output of the .0000 logs. Submit the rsl.error, rsl.out, singleDomain.log logs and the SLURM batch configuration file.

# Benchmark 2: Nested Two-Domain Case

Repeat the above experiments, to optimize the ratio of OpenMP Threads to MPI Ranks for the nested Two-Domain Case. Please take note that you do not need to recompile the application, unless you are optimizing your application binaries through:

* Testing various implementations of Compiler and MPI, and/or
* Dependency and library versions, and/or
* Compiler optimization flags, and/or
* Any other means to change the application binary,