Optimizing WRF performance

These recommendations for optimizing the performance of the Weather Research and Forecasting (WRF) modeling system are based on the results of numerous jobs that were run on the **Cheyenne system** by the CISL Consulting Services Group. The jobs included small runs and others with a variety of domain sizes and time steps.

Page contents

Configuring and compiling

CISL experiments showed that using the Intel compiler resulted in *substantially better* WRF model performance than using the GNU compiler. We recommend using the latest Intel compiler and the compiler's default settings as contained in the WRF **configure** script.

> We *do not* recommend using the PathScale compiler to compile WRF.

The script for creating the **configure.wrf** file has options for compiling WRF with some optimization flags that can potentially improve the model's performance. These are the **INTEL (ifort/icc): HSW/BDW** configure options in WRF 4.0.

Experiments showed that using **-fp-model fast = 2** for Intel and **-ofast** for the GNU compiler improved computation speed by approximately 10%. These optimizations, however, may result in less precise computation results.

Users should compile WRF with Distributed-Memory Parallelism (DMPar) for builds with only MPI tasks. Depending on the individual case, advanced WRF users may find some improvement in performance with a hybrid build, using both DMPar and SMPar.

> We *do not* recommend SMPar alone or serial WRF builds.

Run-time options

Hyper-threading significantly reduced simulation performance in some tests performed with 144 and 2,304 cores.

Tests of hybrid MPI/OpenMP jobs, both with and without hyper-threading, showed that hybrid parallelism can provide *marginally higher* performance than pure MPI parallelism.

Run-to-run variability was high, however, and performance could degrade easily if some secondary environment settings are changed. Therefore:

- We recommend hybrid MPI/OpenMP jobs and hyper-threading only for advanced users who are trying to squeeze the very best performance out of their runs.
- For others, we recommend pure MPI *without* hyper-threading (that is, using 36 cores per node) as very easy to do, stable, and still providing good performance.

Processor binding was enabled by default when running MPI jobs on Cheyenne. We used the default binding settings provided by **mpiexec_mpt** in test runs.

Scaling and core count

Computationally, WRF is *highly scalable* and it can be run on extremely large core counts. The initialization, domain decomposition, and I/O, however, *do not scale as well* and therefore can limit the largest reasonable number of cores for a job.

See WRF scaling and timing on Cheyenne (particularly Figure 2) for details.