WRF

The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed to serve both operational forecasting and atmospheric research needs. It features multiple dynamical cores, a 3-dimensional variational (3DVAR) data assimilation system, and a software architecture allowing for computational parallelism and system extensibility.

Building WRF requires a C compiler, a Fortran90 compiler, Perl version 5, MPI Libraries, and the NetCDF libraries.

The WRF benchmark can be downloaded from http://web.ncar.teragrid.org/~bmayer/benchmarks/WRFV2.tar.gz

1 Procedure

Follow these steps to build and run the WRF benchmark.

1. In the WRFV2 directory, run **./configure.** On systems already supported by WRF, this script will present a list of compile options for your computer, select the option you wish to compile. A **configure.wrf** file will be created. The **configure.wrf** file is to be submitted with the benchmark results. It may be modified if desired. For other systems, it may be necessary to construct a **configure.wrf** file. Refer to templates listed in the **configure.defaults** file in the **arch** directory.
2. Once a **configure.wrf** script has been generated, compile the code by typing:

```
./compile em_real
```

in the top-level **WRFV2** directory. The resulting executable is **wrf.exe** in the **run** directory. A symbolic link to the executable is created in **test/em_real**, where the model is typically run.

1. When reconfiguring the code to compile using a different configure option, type

```
  ./clean --a
```

before issuing the **configure** and **compile** commands again. The previous **configure.wrf** file will be delete by **clean --a**.

1. Two benchmark cases are included with this distribution of WRF, a 12 km simulation and a 2.5 km simulation. The data for these benchmark cases is located in the **WRFV2/data/[12km, 2.5km]** directories, respectively. To run one of the benchmark cases copy all the files from the relevant data directory into the **WRFV2/test/em_real** directory, which contains a link the **wrf.exe** executable, and run the model from this directory.
2. Unless otherwise directed, it is not necessary to edit the **namelist.input** file that is provided with the case; however, for distributed memory parallel runs, the default domain decomposition over processors in X and Y can be overridden, if desired. Add **nproc_x** and **nproc_y** variables to the domains section of **namelist.input** and set these to the number of processors in X and Y, keeping in mind that the product of these two variables must equal the number of MPI tasks specified with **mpirun** minus the number of I/O processes (if any).

The **namelist.input** file also provides control over the number and shape of the tiling WRF uses within each patch. On most systems, the default tiling will be only over the Y dimension; the X dimension is left undecomposed over tiles. This default behavior is defined in **frame/module_machine.F** in the routine **init_module_machine**. The default number of tiles per patch is 1 if the code is not compiled for OpenMP or the value of the OpenMP function **omp_get_max_threads()**. This default is defined in **frame/module_tiles.F** in the routine **set_tiles2**. The default number of tiles may be overridden by setting the variable **numtiles** to a value greater than 1 in the domains section of the **namelist.input** file. Or the default shape of a tile can be specified by setting **tile_sz_x** and **tile_sz_y** in the domains section of the **namelist.input** file. These will override both the default tiling and **numtiles** if it is set.

1. Run the model according to the system-specific procedures for executing jobs on your system. The model will generate output data to files beginning with **wrfout_d01** and, for distributed memory runs, a set of **rsl.out.dddd** and **rsl.error.dddd** files where **dddd** is the MPI task number. You can use either RSL or RSL_LITE for distributed memory runs (RSL_LITE is recommended). When running with RSL, a file **show_domain_0000** will also be generated, which contains decomposition information. With RSL_LITE the decomposition information appears in the form of patch starting and ending indices at the beginning of each **rsl.error.*** file. For non-distributed memory runs, capture the output from the running program to a file to be returned along with the benchmark results.

2 Validation

Each run of a benchmark test case will produce an output data file named:

wrfout_d01_2001-10-25_03:00:00 for the 12km case

or

wrfout_d01_2005-06-04_06:00:00 for the 2.5km case

and a corresponding reference file named **wrfout_reference** is included in the data directory for each benchmark case. Numerical validation can be examined by running the script **diffwrf** to compare model output against the reference file as:

diffwrf *your_output* wrfout_reference > diffout_*tag*

Sample output from **diffwrf** runs are included in the data directories in the files **diffwrf_conus12.txt** and **diffwrf_large.txt**. The number of digits of agreement in your **diffwrf_*tag*** files should not differ significantly from that shown in this sample output. The **diffwrf** program is distributed and compiled automatically with WRF. For the 12 km case the **diffwrf** version that should be used can be found in the **external/io_netcdf/** subdirectory of the top level WRFV2 directory. For the 2.5 km case the **diffwrf** version in the **external/io_int/** subdirectory should be used. For both versions the executable is named **diffwrf**. Note: if the files being compared by **diffwrf** are identical then no output will be produced.

3 Data

For both the 12 km benchmark and the 2.5 km benchmark the amount of wallclock time elapsed during each time step is written to the **rsl.out.*** files (for distributed memory parallel) or to standard output (non-distributed memory). The performance of the WRF benchmarks will be measured as the average of the times labeled “Timing for main” written either to the file **rsl.out.0000** (distributed memory) or to the file capturing redirected standard output (non-distributed memory). The “average time” can be calculated, for example, using the command:

```
grep "Timing for main" rsl.out.0000 \| awk 'BEGIN{t=0;at=0;i=0;}{t=t+$9;i=i+1;}END{at=t/i;print "\nAverage
Time: " at " sec/step over " i " time steps\n"}'
```

The 12 km WRF benchmark should be run on multiple processor counts: using 1 processor, using the total number of available batch processors in the system, and on each power of two between 1 and the maximum number of processors, (i.e. 1, 2, 4, 8, …, max_pe). The “average time” (as calculated in the above example, or similarly) should be recorded in the K_WRF worksheet of the benchmark results spreadsheet.

The 2.5 km WRF benchmark should be run using 128 processors, the maximum available number of batch processors, and two equally spaced processor counts in between 128 and the max. Offerors may optionally choose a smaller number of processors than 128 for the minimum (e.g. 64 or 32) for the 2.5 km WRF benchmark. The “average time” (as calculated in the above example, or similarly) should be recorded in the K_WRF worksheet of the benchmark results spreadsheet.

For each run return the following files:

- namelist.input
- configure.wrf
- Either of the following
- rsl.error.0000 and rsl.out.0000 (distributed memory parallel)
- terminal output redirected wrf.exe (non-distributed memory)
- diff_*tag* (output from **diffwrf** - see validation section for details)

In addition, submit a **tar** file of the WRFV2 source directory with any source code modifications you may have made. Only one such file is needed unless you used different versions of the code for different runs. Please run **clean --a** in the WRFV2 directory and delete any **wrfinput**, **wrfbdy**, **wrfout**, and any other extraneous large files before archiving. Compress the **tar** file with **gzip**; it will not be larger than 10 MB if you have cleaned and removed data files. It is only necessary to submit one **tar** file of the WRFV2 source directory if you are submitting benchmarks for more than one case and if you have not modified the source code from case to case.