

Basic WRF Tutorial - June 2009

WPS and WRF on *midnight.arsc.edu*

Preparing and Running Simple Test Cases

Pre-requisites

- Basic Unix familiarity (text editing, file commands)
- Basic familiarity with PBS job submission on midnight
- WRF Overview Presentation would be helpful

Setting up your environment

From your local workstation, log on to midnight2

```
ssh -X -Y midnight2.arsc.edu
```

```
[In general, you may also use midnight.arsc.edu, but midnight2 tends to  
be less crowded and, at least for this group tutorial we ask that you  
use midnight2]
```

The version of WRF compiled on midnight uses the Portland Group Programming Environment. You might find it necessary to insure you have this loaded up. To do so, in your compute-node window

```
module purge  
module load PrgEnv.pgi  
module load ncl-5.1.0
```

Note that the *module load ncl-5.1.0* isn't necessary for the WRF routines, but is for some of the graphics utility routines that you might find yourself using.

Setting up new directories for Katrina case study

First, you should change to your \$WORKDIR directory

```
cd $WORKDIR
```

This is a directory with lots of disc space, but it is not backed up, and it will be purged after about a month of inactivity. If you want to save any of this afterwards, you can copy it to your \$ARCHIVE directory (unlimited storage) or your \$HOME directory (limited storage).

A centralized version of WRF resides on midnight. To run a case study you'll need to run a script which will create two new directories in your current working directory (cwd). The script is currently available at

```
/datadir/morton/WRF/WRFV3.1-20090605/userWRFSetup.sh
```

The script expects a single argument - what you want to name your case study domain. In this first tutorial, we're doing a Hurricane Katrina simulation, so, we could run the script as

```
/datadir/morton/WRF/WRFV3.1-20090605/userWRFSetup.sh Katrina
```

If you notice any error or warning messages during the quick execution of this script, you should not proceed past this point - seek help!

Upon successful execution of the script, you'll see a new directory, Katrina, with subdirectories:

WPS - this is where you'll create your domain and perform pre-processing activities

WRFRun - this is where you'll run the simulation

Setting up the domain and input data with WPS

To prepare your case study for execution you'll make use of components in the WRF Preprocessing System (WPS), and you'll do all this from within your new Katrina/WPS directory.

This will consist of three primary steps:

1. Use *geogrid.exe* to define your modeling domain - region of interest, modeling resolution, number of grid points, etc.
2. Use *ungrib.exe* to ingest input files (typically in GRIB format) and convert (ungrib) them to an intermediate format used by WPS.
3. Use *metgrid.exe* to extract relevant data from your intermediate input files and ingest them into the domain created with *geogrid.exe*. This will result in the production of a set of files with input data mapped to your specific domain, and these files will be used later in the *Katrina/WRFRun* directory for creating the initial and boundary conditions used for your simulation.

In all three steps, executables will look at the file *namelist.wps* in your *Katrina/WPS* directory for specific parameters.

geogrid.exe

All you're doing in this step is specifying the properties of your domain – dimensions, location, size, projection, etc. This will require modification of several entries in *namelist.wps*, running *geogrid.exe*, then looking at the output of this step to verify things look right.

In directory *Katrina/WPS* edit *namelist.wps*. Available editors include *vi*, *emacs*, and *pico*.

- For *geogrid.exe*, you only need to worry about a few parameters.
- The multiple columns for some of the fields are used in cases where you're creating multiple nests. In this example we're only using one nest, so we only need to worry about editing the first column – any additional columns will be ignored by the executables.
- In the *&share* section, set **max_dom = 1** - this specifies the number of nests (or domains).
- Then, in the *&geogrid* section, we'll specify the parameters for the domain. The domain for this case study will be 38x35 grid points with 60km resolution, centered at latitude 25N, longitude 85W on a Mercator map projection. Replace the following values in your existing *namelist.wps* as specified below:

```
e_we = 38           # This is the end grid point
                   # in the east-west direction
e_sn = 35           # This is the end grid point
                   # in the north-south direction

geog_data_res = 10m # Use topo resolution of 10 minutes
                   # (approximately 16 km)

dx = 60000          # Distance between grid points (m)
dy = 60000          # Distance between grid points (m)

map_proj = 'mercator' # Map projection

ref_lat = 25.0      # Center latitude (positive is north)
ref_lon = -85.0     # Center longitude (positive is east)

truelat1 = 25.0     # We'll use the ref_lat
stand_lon = -85.0   # We'll use the ref_lon

geog_data_path = '/datadir/WRF/geog'
```

Now, you've set your parameters for *geogrid.exe*.

A nice, quick 'n dirty approach for viewing your domain is to use the provided utility *util/plotgrids.exe*. Just type

```
util/plotgrids.exe
```

from within the *Katrina/WPS* directory. This program will look at the domain information in *namelist.wps* and create an NCAR Graphics metafile called *gmeta*. This file is viewable with the *idt* program:

```
idt gmeta
```

idt will not work correctly if you haven't previously loaded the *ncl* module.

Once you're happy with the domain, save *namelist.wps* and run (from the *compute-node*)

```
./geogrid.exe
```

Assuming no errors, you should get a final message:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Successful completion of geogrid.                                !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

and you should see a new file, *geo_em.d01.nc* in your directory. This is a netCDF file and contains all of the information needed to proceed. Additionally, you can view it with any netCDF viewer to make sure you created the domain you intended.

For this tutorial, a quick and dirty way of viewing the output is with *ncview*.

```
ncview geo_em.d01.nc
```

Or, if you're familiar with *ncdump*, you might just use that tool to browse the dimensions for a quick verification.

ungrib.exe

In order to set up initial and lateral boundary conditions for a WRF run, we need data. Typically, we use the output files of a coarser-resolution weather model and, through a series of steps,

extract and interpolate this data to create a file of initial conditions and a file of lateral boundary conditions.

The *ungrib* process simply takes the output files from the coarser-resolution weather model and converts them to a standard, intermediate format for use in subsequent steps.

The data that we'll use to drive our Katrina Case study will be the output of the AVN global weather model. Location of input data for the Katrina test case is:

```
/datadir /morton/TutorialData/KatrinaAVN
```

The files here are GRIB output files from the AVN global weather model, run with a start time of 2005-08-28_00Z for 72 hours. These files represent the output at 6-hour intervals on a 1x1 degree grid (approximately 100x100km near the equator).

Steps:

1) Modify *namelist.wps*

In the *&share* section, set the **start_date** and **end_date** fields for this case. Recall that since **max_dom** is set to 1, we only need to worry about the first column - other columns will be ignored.

```
start_date = '2005-08-28_00:00:00' # 00 Zulu, on 28 Aug 2005
end_date   = '2005-08-31_00:00:00' # 00 Zulu, on 31 Aug 2005

interval_seconds = 21600 # 6 hours - interval of
input data files
```

In the *&ungrib* section, it will not be necessary to make any changes

2) Link to Vtable

To convert the AVN input data to a WPS intermediate form, we need a table that describes what variables are in the input data and where they're found. This map is referred to as a Vtable, and we need one for each type of input. A collection of common Vtables is available in the directory *ungrib/Variable_Tables*. The Vtable that works with the AVN data is *Vtable.GFS*.

We'll make this the default Vtable for this case study by making a link to it in the top-level *Katrina/WPS* directory. So, in directory *Katrina/WPS*:

```
ln -s ungrib/Variable_Tables/Vtable.GFS Vtable
```

You'll now have a file called *Vtable* in your directory, which is actually a link to the *Vtable.GFS* in *ungrib/Variable_Tables*. This is a text file, so you can look at its contents if you're curious.

3) Link to input files

Finally, we need to make the input files available in the current directory for *ungrib.exe*. These files tend to be huge, so copying them into the WPS directory is not always efficient. However, we can use the WPS utility *link_grib.csh* to make links in the WPS directory to the actual input files. In this Katrina case,

```
./link_grib.csh /datadir/morton/TutorialData/KatrinaAVN/avn*
```

Note that the wild-card on the filenames needs to be chosen so that it will represent ALL necessary files in the input file directory.

After running this, you should see a set of new links in your WPS directory, *GRIBFILE.AAA*, *GRIBFILE.AAB*, ..., pointing to the individual input files.

4) Run *ungrib.exe*

You've now modified your *namelist.wps*, linked to the correct *Vtable*, and made links to your input files. All you need to do now is run (from your *compute-node*)

```
./ungrib.exe
```

Upon successful execution, you should see the message:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
! Successful completion of ungrib. !  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

and a set of new files, *FILE:2005-08-28_00*, *FILE:2005-08-28_06*, ... Essentially, you should have one new *FILE:* for each of your *GRIBFILE* links. These new "ungribbed" files contain the same data as the original input files, but in a different format that the remainder of the WPS routines will be able to work with. So, each of these new files contains newly-formatted global weather model output for a particular time.

metgrid.exe

The next step involves extracting and interpolating the ungribbed data from the previous step to match your specific domain. In this case, the ungribbed data constitute global model output at approximately 100km resolution. However, your domain consists of only the Gulf of Mexico

region, at 60km resolution. So, it will be the job of *metgrid.exe* to extract just the data for your region, "filling in" through interpolation the intermediate grid points in your domain.

At this point, you shouldn't need to edit anything in *namelist.wps*, so can simply run (from your *compute-node*)

```
./metgrid.exe
```

Upon successful termination, you should see

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Successful completion of metgrid.  !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

This program will use your ungribbed files and domain information as input, and produce a set of files of the form

```
met_em.d01.2005-08-28_00:00:00.nc, met_em.d01.2005-08-28_06:00:00.nc, ...
```

Each one of these files contains the input data now mapped to your specific domain. These are netCDF files and may be viewed with standard netCDF viewers.

This completes your use of WPS - you defined a domain, then you ungribbed input data files, then used this ungribbed data to create a set of data files containing input data mapped to your new domain. You'll now proceed to setting up and running WRF. It's time to leave the *Katrina/WPS* directory and cd into the *Katrina/WRFRun* directory

Setting up and running WRF

There are two primary steps in setting up and running WRF. First, you'll run *real.exe* which will use your *met_em** files from WPS as input to create a file of initial conditions (*wrfinput_d01*) and a file of lateral boundary conditions (*wrfbdy_d01*). Finally, given the initial and boundary conditions, you will run *wrf.exe* to start your weather simulation.

Run real.exe

Before running *real.exe*, the *namelist.input* file needs to be modified for this case. This can be somewhat involved. Then, you'll copy the *met_em** files from the WPS directory into the WRFRun directory (or make links to them), and run *real.exe*.

namelist.input - this file is accessed by both *real.exe* and *wrf.exe* to read parameters specific to the case-study.

For the first run, even though we've prepared 72 hours of input data, we'll just plan on running a 12-hour case. Although there are many fields in this namelist, the only ones you need to change for this Katrina tutorial are listed below. As with WPS, a number of the entries have multiple columns, corresponding to nested runs, but for this case study you only need to worry about the first column; additional columns will be ignored by the executables.

In section *&time_control*:

```
run_hours = 12

start_year = 2005
start_month = 08
start_day = 28
start_hour = 00

end_year = 2005
end_month = 08
end_day = 28
end_hour = 12

interval_seconds = 21600  (interval between input files)
```

In section *&domains*:

```
time_step = 180

e_we = 38

e_sn = 35

e_vert = 31

num_metgrid_levels = 27  (use ncdump on a met_em* file to find
this)

dx = 60000
dy = 60000
```

Now, before running *real.exe*, create links to the *met_em** files from your directory *Katrina/WPS* into your *Katrina/WRFRun* directory

```
ln -s ../WPS/met_em* .
```

Now, to run *real.exe* - actually, we're going to do this on a single processor for now, so use the *real-serial.exe* executable, instead:

```
./real-serial.exe
```

This program will read numerous parameters from *namelist.input*, and get input data from the *met_em** files. If this was successful, the last line you should see is:

```
d01 2005-08-28_12:00:00 real_em: SUCCESS COMPLETE REAL_EM INIT
```

and you should see two new files in your *Katrina/WRFRun* directory - *wrfbdy_d01* and *wrfinput_d01*. These are the lateral and initial boundary conditions for the simulation you're about to perform. All of the work you've done up to now was aimed at creating these two files, and these two files are ultimately the only input data needed directly by *wrf.exe*.

Run wrf.exe

For this tutorial, *namelist.input* is already set up, so you can just run WRF. Again, we'll use the serial version for now, so run (from your *compute-node*):

```
./wrf-serial.exe
```

As this is running, after seeing a lot of meaningless text, you'll see the simulation timestepping away. The output of the simulation will be printed at 3-hour intervals to the netCDF file

```
wrfout_d01_2005-08-28_00:00:00
```

This will probably take about 5 minutes to run through the full 12-hour simulation. You can use *ncview* for a crude check of the output. Again, there are numerous other tools for viewing this output, such as *NCL* and *IDV*, both of which are beyond the scope of this tutorial.

Running WRF in Parallel

At this point, you've just completed your first run of a WRF simulation, and we at ARSC are all very, very proud of you. This was a fairly small test-case intended to be used for instruction. In the real world you'll likely be running with more computationally-intensive domains and will need to run some of the WRF utilities on multiple CPU's. The parallel capabilities of WPS/WRF include

- Major utilities in WPS can be parallelized, but for most of my simulations, I have found the serial executables to be quite adequate, particularly when used on a *compute-node*.

There are times when your domain might be too large for a single CPU to handle, and in this case you'll need to resort to parallel executables as a possible solution.

- In the centralized WRF used in this tutorial, *real.exe* and *wrf.exe* are parallelized, and *real-serial.exe* and *wrf-serial.exe* are meant to run on a single CPU. The serial version of *real.exe* is usually sufficient, and this program is not that scalable. On large domains you might realize significant speedup by using *real.exe* on four to eight CPU's.
- There's no problem with using the output of *real-serial.exe* as input for parallel *wrf.exe*. *real.exe* and *real-serial.exe* should produce exactly the same output files for use as input to *wrf.exe*. So, this tutorial will only address the parallel execution of *wrf.exe* on *midnight*. Parallel execution of *real.exe* – if desired - would be done in a very similar fashion.

Parallel execution of *wrf.exe*

Running *wrf.exe* in parallel will consist of four primary steps

- Create a script for the Portable Batch System (PBS). This is the system that will actually allocate resources for your job and run the job when resources are available. If resources are not immediately available, your job will be held in a queue (which you can monitor)
- Submit the job to PBS
- Monitor job status
- Check the output

Creating a PBS script

A template PBS script already exists for you in the *Katrina/WRFRun* directory – *wrfrun.pbs*. A thorough discussion of PBS is beyond the scope of this tutorial, but you should only need to make the following changes to *wrfrun.pbs*

- **#PBS -q standard** *standard* is the name of the queue (see ARSC documentation for other possibilities).
- **#PBS -l select=2:ncpus=4:node type=4way** This option will request a total of eight cpus. In this directive, you are requesting two 4-way nodes, each possessing four cpus (more accurately, we refer to these as *cores*).
- **#PBS -l walltime=00:30:00** This indicates the amount of time – hours:minutes:seconds – that you want the resources for. After this time has expired the job will be killed by PBS. For a 12-hour Katrina simulation, setting this to 2 minutes should be more than sufficient.
- **mpirun ./wrf.exe** This is the command that actually runs *wrf.exe* on the cpus you requested. Note that the previous #PBS lines were used to REQUEST resources, whereas this line is used once the resources have been allocated to your job.

Submitting the job

To submit the job, use the *qsub* command to give the PBS script you just created to the scheduler, which will then take over

```
qsub wrfrun.pbs
```

Upon submission, you will get a message referring to the unique job number that you just submitted. This job number will be used to reference this job when you view its status, look at output messages, etc.

Monitoring the job

Use the *qstat* command to monitor the status of jobs in the PBS system

```
qstat -a
```

or

```
qstat -u <your user-id>
```

A **Q** in the second-to-last column indicates that the job is queued up, awaiting resources. An **R** indicates that the job is running

If your job is not listed, then it's quite likely that it already finished. Hopefully it was a successful execution (for this tutorial, a 12-hour Katrina forecast on 16 cpus will likely take less than a minute), but you should consider that maybe there was an error in your PBS script. Any PBS error messages will be written to the file *wrfrun.pbs.o<job-number>*.

Each task in a parallel *wrf.exe* run will have its own *stderr* and *stdout* streams, going to files *rsl.error.0000*, *rsl.error.0001*, ... and *rsl.out.0000*, *rsl.out.0001*, ..., where the number indicates the task number that produced the file. Usually, it's sufficient to look just at *rsl.out.0000*. During a long run, you can monitor the timestepping by issuing the command

```
tail -f rsl.out.0000
```

As with the run of *wrf-serial.exe*, model output will be written to the file

```
wrfout_d01_2005-08-28_00:00:00
```

Other Things You Can Do

At this point, you've gone through the setup of a WRF run, and you've executed it in serial and in parallel. Here are some other things you might want to try now. As you try these variants, I highly recommend that you use the *userWRFSetup.sh* utility to create new working directories and start from scratch for each new case study. For example

will create a new directory, *Katrina-Case2*, with newly-created subdirectories *WPS* and *WRFRun*. In past tutorials I have found that attendees often get themselves really messed up when they try to modify their existing namelists for a variant of the existing case study. Wait until you feel comfortable with WRF to try that! At this point, it's better for you to start each case study from scratch, in order to gain familiarity with the entire process.

- You actually have 72-hours of *met_em** files in your *Katrina/WRFRun* directory. You can try a run longer than 12 hours. To do this, you would need to modify the following fields in *namelist.input* - run_hours, end_year, end_month, end_day, end_hours. You would then need to re-run *real.exe* (in order to produce a *wrfbdy_d01* that has lateral boundary conditions for more than the 12 hours you originally specified), then *wrf.exe*. If running *wrf.exe* in parallel you should make sure you request enough walltime in the PBS script.
- Perform a simulation for a different region of the planet, using the same input files that you used for the Katrina simulation (in other words, for the 72-hour period starting at 2005-08-28_00Z). Although you could use the *Katrina/WPS* and *Katrina/WRFRun* directories, just modifying things as you go along, I recommend that you start this tutorial from the beginning, creating new WPS and WRFRun directories, running *geogrid.exe*, then *ungrib.exe*, then *metgrid.exe*, then *real.exe*, and finally *wrf.exe*. Here are some guidelines (mostly for the purpose of this tutorial, to keep you from running something too big right now) for setting up your new model domain:
 - *geogrid.exe*
 - Continue to use approximately 60km resolution
 - Try to keep your grid dimensions no larger than 100x100
 - For your projection, polar latitudes often use a *polar* projection, mid-latitudes a *lambert* projection, and tropical latitudes a *mercator* projection. However, you can use whatever you want. If you use a *lambert* projection, you will need to define both a *truelat1* and a *truelat2* in *namelist.wps* – these values define the latitudes at which the plane of your projection intersects the sphere.
 - *ungrib.exe*
 - Since ungribbed files are not related to a particular domain – recall that they simply represent the input files transformed to a format the WPS prefers to work with – you could copy the ungribbed files over from your *Katrina/WPS* directory, though I recommend you do the ungribbing again, just for the practice.
 - *real.exe* and *wrf.exe*
 - For the most part, you should modify *namelist.input* as you did for the Katrina tutorial, though you'll have to enter different grid dimensions in the *&domains* section. For the time_step field, a general rule of thumb

is that the value (which is in seconds), shouldn't be larger than three times your resolution (in km). So, for a 60km grid you would want a timestep size no greater than 180s. In calm conditions, you can often get away with one that's larger, and in active conditions, you might sometimes need to use a smaller timestep. Keep in mind that larger timesteps get you through the simulation faster.

More Information

- ARSC-WRF Google Group – <http://groups.google.com/group/arsc-wrf>
- WRF Users Page - <http://www.mmm.ucar.edu/wrf/users/>
- WRF ARW Users Guide - http://www.mmm.ucar.edu/wrf/users/docs/user_guide_V3/contents.html
- WRF ARW Online Tutorial - <http://www.mmm.ucar.edu/wrf/OnLineTutorial/index.htm>
- ARSC Weather Modeling – <http://weather.arsc.edu/>