Using the WRF-ARW on the UWM *mortimer* HPC
Guide for WRF-ARW Version 4.0.3
February 19, 2019

Introduction

This guide is designed to facilitate basic compilation and execution of the WRF-ARW v4.0.3 model core using OpenMPI on the UWM *mortimer* HPC. I cannot cover every option available to you when installing or running the model; for all of the gory details, see the WRF-ARW User's Guide, available online at:

http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_v4/v4.0/contents.html

I also recommend reading over and familiarizing yourself with the information found in the UWM HPC User's Guide, available online (from on-campus machines only) at:

http://www.peregrine.hpc.uwm.edu/Webdocs/uwm-rc-user-guide.pdf

Note that the steps outlined in this User's Guide are generally applicable to both earlier and newer versions of the WRF-ARW model; however, there may be subtle differences in compilation options between each version of the model. Always read the installation information from the appropriate version of the WRF-ARW User's Guide first!

What You Need to Know First

Compile Environment: We will use the Intel compilers to build WRF on the UWM HPC. The Intel compilers are available through a module on the UWM HPC that will set all of the necessary path, library, and execution variables for you automatically. To load this module, run:

```
module use /sharedapps/ICC/modulefiles/
module load icc/15.2
```

MPI Libraries: You will be using OpenMPI to run parallelized (multiple CPU) jobs on the cluster. To correctly configure OpenMPI, you need to load the appropriate module. This is done by issuing the following command *after* you issue the `module load icc/15.2` command listed above:

```
module load openmpi/1.10.7
```

NetCDF Libraries: Before compilation and execution of the model code, you'll need to point to a netCDF installation on the server. This is also done using modules:

```
module use /raid-08/LS/evans36/modules/
module load netcdf-4.6.2
module load hdf5-1.10.4
```

The HDF5 library must be loaded for the netCDF4 compression capabilities to be enabled.

PnetCDF Libraries: If you wish to use the parallel netCDF file output option (not recommended), you must also load the pnetcdf module:

```
module load pnetcdf-1.11.0
```

GRIB2 Libraries: To use GRIB2-formatted data as inputs to WRF-ARW, you'll need to point to the GRIB2-related libraries on the server. To do so, load the following module:

```
module load grib2-libs
```

I recommend adding all of the above commands to your ~/.bashrc file so that they are loaded automatically each time you log in. To do so, issue the following command:

```
nano ~/.bashrc
```

In the nano text editor, simply add the commands from above to the end of the file, save the file (Ctrl-O), exit nano (Ctrl-X), and then log out and log back in to the supercomputer. There are other variables that you may need to set along the way during WRF or WPS compilation and/or run-time; I will alert you to those where appropriate.

Mortimer Access Considerations

Remote Access: mortimer is accessible *only* from machines with uwm.edu IP addresses. Thus, if you wish to connect to mortimer from off-campus, you will first need to connect to the UWM VPN (Virtual Private Network). Three steps are involved. First, visit https://remote-access.uwm.edu/, login with your ePanther credentials, and download and install the appropriate GlobalProtect VPN client. The second is to e-mail Dan Siercks and request that your ePanther ID be given VPN access credentials to mortimer. The third, completed each time you wish to access mortimer, is to run the VPN client and login using your ePanther credentials. Once connected to the VPN, you should be able to access mortimer as if you were on campus.

Data Transfer: To transfer data from mortimer to your laptop or desktop using sftp or scp, you must specify the remote port that you wish to access. For your data, which resides on /raid-08, this is port 22008. This port is specified with the -P ##### flag, i.e.,

```
sftp -P 22008 <user>@login.mortimer.hpc.uwm.edu
```

Visualization: By default, only command-line resources are available on mortimer. Graphical, or XWindows, resources are available only on a dedicated visualization node. This node is accessed through mortimer (on the login node rather than a computational node), by issuing the following command to access the visualization node:

```
ssh -Y vis-1
```

Note that you must be on a Linux/Unix machine, one of the Macs in EMS W434, or using the MobaXterm package for Windows to be able to use graphical resources via the visualization node.

If you wish to use the ImageMagick (display, convert, etc.) programs on the visualization node, you must first issue the following command (all on one line):

```
module load /sharedapps/pkg-
2018Q2/etc/modulefiles/pkgsrc/2018Q2-non-exclusive
```

Part I: Obtaining & Compiling the Model

Connect to a Compute Node on mortimer

To begin, we wish to connect to a compute node on mortimer. Because the WRF model requires a lot of memory in order to successfully compile, the typical "slurm-shell" command will not work for this task. Instead, we manually connect from the mortimer login node using srun:

```
srun --mem-per-cpu=8gb --ntasks=1 --pty --preserve-env $@ $SHELL -l
```

Subsequently, your command prompt will change slightly, reflecting that you are now on a compute node (compute-#-##) rather than the login node.

Obtaining the WRF Model Code

You can the WRF model code using git. Change into the directory into which you wish to install WRF-ARW and WPS; this should be a subdirectory within your work or "data" directory. Next, issue the following two commands:

```
git clone https://github.com/wrf-model/WRF
git clone https://github.com/wrf-model/WPS
```

These will clone the git repositories for WRF-ARW (v4.0.3 at the time of this writing) and WPS.

Installing the WRF v4.0.3 Model Code

Once you have cloned the WRF repository, you need to switch into the newly created WRF directory. First, open the `arch/Config_new.pl` script with a text editor. Change line 250 ($I_really_want_to_output_grib2_from_WRF) from FALSE to TRUE, then save the file and exit the text editor. Next, issue "`./configure`" (without the quotes) to the command line to start model configuration. A list of configuration options will appear. You will want to choose option 15, "`15. (dmpar) INTEL (ifort/icc)`". It will then ask you for a nesting option. Unless you plan on using advanced nesting options with the model, I recommend using option #1, the default, for basic nesting capabilities.

Once configuration is complete, open the configure.wrf file that is created with a text editor. Change what comes after the equal sign for both DM_FC and FC – lines 155 and 157 – to read mpif90. Save this file, then exit out of the text editor.

If you wish to use this WRF installation with the Data Assimilation Research Testbed software, a couple of additional edits to the WRF Registry are needed before compiling the model. First, open Registry/Registry.EM_COMMON with a text editor. Search for h_diabatic. Change rdu to read i012rhdu. Next, search for refl_10cm. Change hdu to read i012rhdu. Save this file, exit the text editor, and return to the main WRF directory.

Finally, compile the model by issuing the following command:

```
./compile em_real
```

In all, compilation will take approximately 60 minutes to complete. The job may appear to hang when compiling several portions of the model code; this is okay. A successful compilation will produce "ndown.exe," "real.exe," "tc.exe," and "wrf.exe" files in WRF/main. After this has completed, return to the login node by typing exit and pressing enter.

Installing the WRF Pre-Processor Code

Installing the WPS code is similar to installing the WRF model code itself. Once you have cloned the WPS repository, switch into the newly created "WPS" directory. Ensure that this directory is on the same level as the WRF directory but is not *in* the WRF directory.

To enable GRIB2 support for WPS, ensure you have loaded the grib2-libs module introduced in the outset of this document. Next, move to a compute node, just as was done above for WRF. After having done so, issue a "`./configure`" command to the command line to start model configuration. A list of configuration options will appear. Choose the option that reads most similar to, "`Linux x86_64, Intel compiler  (dmpar)`" (currently option 19).

After configuration is complete, open up the configure.wps file with a text editor. Edit the entry for NCARG_LIBS to read like the following:

```
NCARG_LIBS = -L/raid-11/LS/evans36/software/ncl/lib -lncarg -lncarg_gks -lncarg_c \
             -L/usr/lib64 -lX11 -lXext -lpng -lz -lcairo -lfontconfig -lpixman-1 \
                          -lfreetype -lexpat -lpthread -lbz2 -lXrender -lgfortran -lgcc
```

This change is necessary to ensure that utilities requiring the NCAR Graphics or NCL libraries are built. After making these changes, save configure.wps and exit back to the command line. Once this is done, compile the code (`./compile`). Compilation may take 1-3 minutes. You can safely ignore any warnings to the effect of "ignoring unknown option '-f90=ifort'." Once compilation has completed, look for "geogrid.exe," "ungrib.exe," and "metgrid.exe" in the current directory; this is your sign of success. If one or more of these programs are missing, check your configure.wps file to ensure that the library paths are all set correctly (as above), then run ./clean -a and start the process anew. Once done, type exit and press enter to return to the login node.

Part II: WRF Pre-Processor (WPS)

What does the WRF Pre-Processor do?

The WPS has three tasks: defining model domains, extracting initialization data for the model simulation from GRIB files, and horizontally interpolating that data to the model domain and boundaries. All of this is accomplished through the use of a namelist file and a few command line options. If you are working through this tutorial and using the WRF-ARW model for the first time, I recommend that you do so alongside the related material within the "Basics for Running the Model" section of the WRF-ARW Online Tutorial, available at:

http://www2.mmm.ucar.edu/wrf/OnLineTutorial/

Step 1: Defining a Domain

Defining a domain is done through the `geogrid.exe` program of the WPS. Options for the domain are set in the `namelist.wps` file. Open this file in some text editor. The first two sections, `&share` and `&geogrid`, are the only two sections of this file to worry about now.

In `&share`, assuming you are not creating a nested model run, change max_dom in 1. Change start_date and end_date to the appropriate dates and times. These take the form of 'YYYY-MM-DD_HH:MM:SS'. The interval between input times of your model data is specified by interval_seconds; for three hourly data, this will be 10,800. Note that unless you are doing a nested run, only the first option in each list matters. Information on nested runs can be found at the end of this document.

In `&geogrid`, e_we and e_sn define the size of your domain in gridpoints, with e_we defining the east-west size and e_sn defining the north-south size. Change these to appropriate values. geog_data_res defines the horizontal resolution of the geographic data files that you wish to use to setup your domain and has four options: 30s, 10m, 5m, and 2m. Generally 5m is fine for a grid spacing of about 20 km or larger; switch down to 2m or 30s for lower grid spacing values. If you want to include inland lake information with the 30s data set, you can use "30s_with_lakes" in lieu of 30s. There are other options available, and I recommend that you review the WRF-ARW User's Guide for details.

dx and dy control your grid spacing; generally they should be equal to one another and are given in meters (default = 30000 = 30 km). map_proj deals with the desired map projection of your run; lambert, mercator, or lat-lon are suggested for most tasks, with the Lambert projection favored for mid-latitude simulations and the Mercator projection favored for simulations at tropical latitudes. ref_lat and ref_lon are the center point of your domain in latitude and longitude, respectively. Note that for west longitudes, ref_lon should be negative. The remaining fields vary depending upon which map projection you use. For instance, the Lambert projection requires three additional fields: truelat1, truelat2, and stand_lon. truelat1 and truelat2 define the "true" latitudes for the Lambert map projection; unless moving to the southern hemisphere, the default values should be fine. stand_lon specifies the longitude parallel to the x-axis for conic and azimuthal projections; this value should generally be close to that of ref_lon. The Mercator projection requires just one additional field, truelat1. The Latitude-Longitude projection requires three additional fields: pole_lat, pole_lon, and stand_lon. This projection is recommended only for global runs and for such runs should not require any changes to pole_lat and pole_lon.

Finally, geog_data_path defines the path to where the geographic data resides on the server. Set this to '/raid-11/LS/evans36/WRFv4/geog'. Once these variables are set, save the namelist.wps file and return to the command line.

To perform a visual check on your domains' extent to ensure they meet your requirements, ncl and the util/plotgrids_new.ncl script may be used. First, log on to the visualization node following the instructions earlier in this document, then change into your WPS directory. Load the ncl module:

```
module use /raid-08/LS/evans36/modules/
module load ncl-6.5
```

Finally, run the script:

```
ncl util/plotgrids_new.ncl
```

This will produce a display containing your domain(s) with the relevant underlying geography. To exit from this window, simply click anywhere inside of it. If you are satisfied with the domain(s), proceed to run geogrid.exe as below; if not, edit the domain particulars in namelist.wps and use ncl and the plotgrids_new.ncl script to re-visualize the domain(s) until you are satisfied. Once done, disconnect from the visualization node (i.e., return to the login node).

Once ready, you can run geogrid.exe using a slurm-shell, i.e.,

```
slurm-shell
./geogrid.exe
```

Once it is done (and it should give you a success message if it successfully completed), check that you have a geo_em.d01.nc file in the current directory; this ensures that it worked correctly.

Step 2: Getting Model Data

Extracting model data from GRIB files is accomplished through the ungrib.exe program. There are two steps you do need to do before running the program: linking the appropriate variable table (Vtable) and linking the appropriate GRIB data.

Residing in the WPS/ungrib/Variable_Tables/ directory are a series of Vtable.xxx files, where the xxx denotes a model name. These files tell the ungrib program about the format of the data files to be degribbed. If you are using the GFS model, for instance, you'll note a Vtable.GFS file in that directory. In the main WPS directory, issue the following command to link this Vtable file:

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

where you would substitute for GFS as appropriate for your initialization data set. For data sets that do not have a corresponding Vtable file in the default WRF installation, you will either need to create your own using the GRIB packing information from your data set or find one that

someone else has already made for that data set.

Next, you need to link your model data GRIB files to the WPS directory. You can obtain GRIB data used to initialize the model from NCEP's NOMADS data server and/or UCAR's Research Data Archive for most data sources. If you need help finding data, please ask! Download these files to the supercomputer, identify where you have placed these files on the supercomputer, and issue the following command:

```
./link_grib.csh /path/to/model/data/model_data*
```

where you will replace /path/to/model/data with the appropriate path and model_data.t00z* with the appropriate file name format of the data files that you wish to link. This will create a series of `GRIBFILE.xxx` files in the WPS directory.

Before running the ungrib program, clear out all old `GRIBFILE.`, `FILE:`, and `PFILE:` files that may exist to avoid inadvertent errors when running the model. Finally, run `ungrib.exe` in a similar fashion to geogrid.exe. If all goes well, you'll see a success message on screen and multiple files of the format `FILE:YYYY-MM-DD_HH` will be present in the WPS directory.

Step 3: Interpolating Model Data

Finally, to horizontally interpolate the model data (obtained in Step 2) to the domain (obtained in Step 1), the `metgrid.exe` program is used. At this point, except in rare circumstances, you will not need to change any of the variables in the namelist.wps file.

Instead of running `metgrid.exe` directly, as with `geogrid.exe` and `ungrib.exe`, we use a job submission script to run metgrid.exe on multiple CPUs. A job submission script specific to our model configuration takes the form:

```
#!/bin/sh
#SBATCH -n ##
#SBATCH --mem-per-cpu=2gb
module use /sharedapps/ICC/modulefiles/
module load icc/15.2
module load openmpi/1.10.7
module use /raid-08/LS/evans36/modules/
module load netcdf-4.6.2
module load hdf5-1.10.4
module load grib2-libs

mpirun ./metgrid.exe
```

The second and third line tell the SLURM scheduler, which is used by mortimer to schedule jobs on multiple CPUs, how many processors to use (replace ## with some number greater than one; for metgrid, I recommend 16) and to use 2 GB of RAM for each processor. We saw an instance of the mem-per-cpu directive in the third line when we used `srun` to connect to a compute node with

8 GB of memory to compile WRF/WPS; in general, however, `metgrid.exe` (and `real.exe` and `wrf.exe` to come) only require <2 GB of RAM per processor. The last line uses the `mpirun` program to run `metgrid.exe` on the specified number of processors.

Place the above lines in a new text file named metgrid.sbatch (or similar), change ## accordingly, and then save the file. To run metgrid, you submit this script to the job scheduler:

```
sbatch metgrid.sbatch
```

Assuming that the necessary resources exist to start your job immediately, metgrid will take 1-30 minutes to complete, with larger amounts of time necessary for longer-duration simulations and/or larger numbers of simulation domain grid points. You may check on its progress by examining the metgrid.log.0000 file in the WPS directory, or by using `squeue` to look for your metgrid.sbatch job. Once metgrid has finished, ensure that you have a series of `met_em.d01.YYYY-MM-DD_HH:00:00` files in the WPS directory. If so, you're done with the WPS and can skip ahead to Part III of this document. If not, check the metgrid.log.0000 file for possible insight into any errors that may have occurred at this step.

Advanced Uses: Multiple Domains

If you want to set up for a run using multiple domains, it is fairly simple to do so. When editing the namelist.wps file, the following things will be different than as presented in Step 1 above:

- Under &share, set max_dom to 2 (or how many domains you wish to have).
- Edit the second listing in the start_date and end_date options.
- Under &geogrid, change the second listing in parent_grid_ratio to whatever downscaling factor you wish to have for the inner domain. The default of 3 is fine for most circumstances (e.g. will take a 30 km outer domain and create a 10 km grid spacing inner domain).
- Change the second listings of i_parent_start and j_parent_start to where in the outer domain you wish the lower left of the inner domain to begin.
- Change the second listings e_we and e_sn to the desired size values of the inner domain. (Note: the values for these must be some integer multiple of parent_grid_ratio plus 1.)
- Change geog_data_res as needed.
- You will not need to change parent_id from 1 unless you wish to create further inner domains that are not based off of the outer domain.

Note that if you have more than two domains, simply add a comma at the end of the second listing under the options listed above and manually type in your third (and beyond) values.

Advanced Uses: "Constant" Input Data Sources

If you want to use a data set as a "constant" value, such as SST data, simply follow steps 1 and 2 above only for the GRIB files containing this constant data, noting that you will be doing this for just one time. Then, in namelist.wps under the `&metgrid` section, add a line called constants_name, e.g.

```
constants_name = 'SST_FILE:YYYY-MM-DD_HH'
```

where the file name is whatever the output file from the ungrib.exe program is named. In the example above, it is an explicitly named (using the prefix option in &ungrib in namelist.wps) SST data file. If you are using multiple data sets, make sure they have different prefix names so as to not overwrite one data set with the other inadvertently! To do this, edit the `prefix` listing under `&ungrib` in the namelist.wps file to reflect the desired prefix name (often for the constant data set), then change it back when re-running it for the actual model input.

Advanced Uses: Time-Varying SST, Sea Ice, Albedo, and Vegetation Data

For long-duration simulations, or shorter-duration simulations of phenomena such as tropical cyclones that significantly influence fields such as sea surface temperature, you may wish to use time-varying inputs for surface fields that are typically held constant throughout the duration of a WRF-ARW model simulation. While WPS geographic data can describe the climatological evolution of albedo and vegetation fraction through time, external time-varying input data are needed in order for sea surface temperature and/or sea ice to be updated throughout the duration of your model simulation.

Sometimes, these data may be provided in the same input GRIB files you use to provide initial and lateral boundary conditions for your model simulations. In this case, no additional steps are necessary at the WPS stage. Other times, however, you may wish to use external data in lieu of those provided by your atmospheric model data set. In such cases, first use ungrib to process your atmospheric data. Next, link the other data sources' GRIB files to the WPS directory, link the appropriate Vtable to the WPS directory, edit namelist.wps to change the prefix entry under &ungrib to some new descriptive name, then run ungrib for these data. Finally, add the prefix for these data to the fg_name entry under &metgrid in namelist.wps (e.g., fg_name = 'FILE','SST' if you used SST as your new prefix), then run metgrid.exe.

I recommend reading through the WPS advanced tutorial information, particularly that related to METGRID.TBL, if you desire to make use of this option for SST data. Care must be taken to ensure that the datasets are blended appropriately.

http://www2.mmm.ucar.edu/wrf/users/tutorial/201901/duda_wps_advanced.pdf

There are some additions to namelist.input for WRF-ARW when using time-varying surface data sets. Please see the companion section to this one in Part III below for the relevant information. More advanced uses of multiple input data sources (e.g. multiple time-varying data sets, ocean mixed layer depth information, etc.) are detailed within the WRF-ARW User's Guide.

Part III: Configuring and Running the WRF Model

Except in the case of a nested domain or idealized simulation, there are two programs that will be used to setup and run the WRF model: `real.exe` and `wrf.exe`. Both of these programs are housed in the `WRF/run/` directory; change over to that directory now. We'll first use real.exe to take the data from the WPS and get it ready for use in the model, then use wrf.exe to actually run

the model. All of this is accomplished on the command line with no GUI options available. For more information, please refer to Chapter 5 of the WRF-ARW User's Guide.

Step 1: Real-Data Initialization

Before editing any of the files necessary for this step, first link the met_em.d01.* files from the WPS to the current working directory (`WRF/run/`) by issuing the following command:

```
ln -s ../../WPS/met_em.d01.* .
```

From here, we can move on to editing the namelist.input file with the necessary parameters. Many of the parameters in the first few sections of namelist.input will be the same as those in the namelist.wps file from the WPS program, so it might be useful to have those parameters handy at this time.

The namelist.input file has several parts, each with multiple variables and multiple options for each of those variables. In particular, you will see sections headed by &time_control, &domains, &physics, &fdda, &dynamics, &bdy_control, &grib2, and &namelist_quilt; some of these will be edited, others will not. Note that this is not intended to be an end-all listing of the options available to you here, particularly in terms of physics packages. Refer to the section of Chapter 5 of the WRF-ARW User's Guide entitled "Description of Namelist Variables" for more information on all of these options. The meanings of many of these variables are readily apparent, so I will only cover those that are not. As noted before, many of these values are the same as those input in the namelist.wps file from the WPS section. Only edit values in the first column (if there are multiple columns for a given variable) for now.

The &time_control section of the namelist.input file is where you will input the basic model timing parameters. Change history_interval to the time (in minutes) between output times you wish for model output. Otherwise, simply change all values above history_interval (except for input_from_file) to the appropriate values and leave all values below history_interval alone. If you want output in netCDF4-format, add a use_netcdf_classic entry and set it to .false,.

The &domains section of the namelist.input file is where you will be inputting information about your model's domain. Your first option relates to the model's time step. If you wish to use a fixed time step, set the time_step variable to a value (in seconds) that is approximately 6 times as large as your model grid spacing in kilometers. For example, for a 15 km model simulation, set this value to 90. It is helpful, but not necessary, if this time_step value is evenly divisible into 3600, the number of seconds in an hour. (If this is desired, you may modify time_step from 6*grid spacing to some other similar value.)

Set the values from max_dom to e_sn to their appropriate values from namelist.wps. Set e_vert to the desired number of vertical levels. Slightly more complicated is num_metgrid_levels. For this value, open a terminal window to the WRF/run directory and issue the following command:

```
ncdump -h met_em.d01.YYYY-MM-DD_HH:00:00 | grep
                num_metgrid_levels
```

where you put in the desired time of one of the met_em files. In the output from ncdump, look for the num_metgrid_levels toward the top of the screen, then cancel out using Control-C. Edit the namelist.input file variable to match this. For NAM input data, this value will be 40; for recent GFS input data, this value will be 32; and for older GFS input data, this value will be 27. Next, set dx and dy to the appropriate values from namelist.wps. Ignore the rest of the options for now; these are generally only relevant to nested runs.

The &physics section is where you will choose what physics packages you wish to include in your model. Refer to the User's Guide for what numeric values you need to select for each of these parameters. Since v3.9, WRF has handled this via physics suites; however, you can override any of the parameterizations in a given suite by manually adding the relevant entries to the namelist. For example, the mp_physics variable defines what microphysics package you wish to use. Longwave and shortwave physics packages are defined in ra_lw_physics and ra_sw_physics. radt is a time step increment and should be set to the same as dx in kilometers (e.g. set this to 18 for dx = 18 km). Surface physics packages are handled with sf_sfclay_physics (surface layer) and sf_surface_physics (land-surface model). Boundary layer parameterizations are given in bl_pbl_physics. bldt is a time step increment; setting this equal to zero will match the model time step. Cumulus parameterization is handled by cu_physics, with the time step to calls to that package given by cudt. Setting cudt to 0 works the same as for bldt. Set ifsnow to 1. The value for num_soil_layers will depend on the land-surface model chosen; for the NOAH and NOAH-MP land-surface models, this should be set to 4. The urban canopy model may be enabled by setting sf_urban_physics to 1. If modeling a tropical cyclone, the following line may be added to enable the usage of a modified surface flux formulation appropriate for tropical cyclones:

```
isftcflx                            = 2,
```

You may also want to consider employing the 1-dimensional oceanic mixed layer model for such simulations. See the WRF-ARW User's Guide for more information on this option.

Ignore the &fdda section. This handles four-dimensional data assimilation options and will not be used (or maybe even present in the namelist) unless specifically performing data assimilation. In general, you will not need to edit anything in &dynamics either; however, the diff_opt and km_opt variables may be tweaked to modify how the model handles diffusion and eddy coefficients. Refer to the User's Guide for more if you choose to modify those variables. I do recommend changing gwd_opt from 1 to 0, however, to turn off the gravity wave drag option. Otherwise, you should not need to edit any other data in namelist.input.

Step 2: Running the Model

The WRF model utilizes two programs, real.exe and wrf.exe, to prepare to and to actually run the model respectively. These programs are both run in parallel (e.g. on multiple machines) using OpenMPI on the cluster. To do so, you will need an appropriate job submission script. An example is given below for real.exe; you may use the same one for wrf.exe if you change all instances of "real" within the text to "wrf."

```
#!/bin/sh
```

```
#SBATCH -n ##
#SBATCH --mem-per-cpu=2gb
module use /sharedapps/ICC/modulefiles/
module load icc/15.2
module load openmpi/1.10.7
module use /raid-08/LS/evans36/modules/
module load netcdf-4.6.2
module load grib2-libs
module load hdf5-1.10.4

mpirun ./real.exe
```

You will need to replace ## with the number of processors on which to run real.exe (or wrf.exe). This number should ideally be some multiple of 8 up to and including 96. The usage of more than 96 CPUs is generally discouraged to avoid excessive use of limited cluster resources. You will want to make sure, however, that you use an identical number of processors for real.exe and wrf.exe as well as for any and all simulation(s) you may conduct for a given research project.

Please also note that not every simulation will require 2 GB of memory per processor (the --mem-per-cpu=2gb flag); some will require less, while some will require more. You are encouraged to check the memory usage of each job using:

```
ssh –t compute-#-## top
```

where #-## is replaced with a compute node on which your job runs. This can be identified using the `squeue` command and looking for the entry corresponding to your running job, then identifying the nodes listed in the last column of that entry. You'll want to look for the value in the "VIRT" column for one of your processes (e.g., real.exe, wrf.exe, or metgrid.exe) and set the value of --mem-per-cpu to a value somewhat higher than the actual usage.

Once the job submission script is ready, submit it to the cluster using sbatch, e.g.,

```
sbatch (name-of-submission-script)
```

After running sbatch, you will want to exit from the slurm-shell prompt by hitting Ctrl-D on the keyboard (or, alternatively, type exit and hit enter). You can monitor the progress of real.exe by looking at the rsl.error.0000 file in the WRF/run directory or by running `squeue –j #`, where # = the job number returned to you upon running sbatch. Once real.exe has finished, you run wrf.exe in a similar fashion utilizing a nearly identical job submission script. Model progress can be examined in the same manner as for real.exe. Once the model has completed, you are ready for post-processing.

Common Errors: SIGTERM Statements in the rsl.* Files

If you are getting SIGTERM errors in your rsl.* files that keep the model from successfully completing, check to ensure that all relevant parameters between your namelist.wps and namelist.input files are identical. If these are identical, then you may be running into an issue where

your simulation requires more memory than is available on the selected nodes (i.e., you've already requested the maximum available per node). To increase the amount of available memory, simply add additional processors in multiples of 8 to the real.exe and wrf.exe job submission scripts.

Advanced Uses: Adaptive Time Steps

With the adaptive time step option, the WRF model will modify the time step up and down as the model integrates in order to find the most efficient yet computationally stable time step. Oftentimes, this speeds up model integration by 25-60%. Note that this option is discouraged for research applications. To use this option, simply add the following line (including the ending comma) immediately below the max_dom option within the &domains section of the namelist:

```
use_adaptive_time_step              = .true.,
```

Advanced Uses: Two-Way Nesting

Most options to get a two-way nest going are handled with one single run of the WRF model and through the namelist.input file. When editing this file, you will note multiple column listings for some of the variables; these extra columns handle information for the inner nest(s). Edit these variables to match the desired values for the inner nest, using the values for the outer nest as a guide. Variables that you did not edit for the single domain run but will need to be edited for a nested run include input_from_file, fine_input_stream, max_dom (the total number of nests), grid_id (1, 2, 3, etc.), parent_id (generally one less than the grid_id), i/j_parent_start (where in the outer domain you want the inner grid lower left hand corner to be), parent_grid_ratio (generally 3 is a good number), parent_time_step_ratio (generally at or near the parent_grid_ratio), and feedback (1 is yes, where the inner grid writes back to the outer one; requires an odd value for parent_grid_ratio). Also, num_metgrid_levels needs to be changed for the nests as well to the number of WRF model levels; see the procedure above to see how to check this.

Notice that I did not discuss input_from_file and fine_input_stream in the previous paragraph. There are many interrelated options to consider for these two variables. The first option is to have all of the fields interpolated from the coarse domain rather than created on their own. This is probably the fastest method, but also may not lead to as accurate of results as otherwise expected. In this case, input_from_file would be .false. and you don't need to worry about fine_input_stream. The second option is to have separate input files from each domain, with input_from_file set to .true.; unfortunately, this means that the nest has to start at the same time as the outer domain. The final option also has input_from_file set to .true., but requires you add a new line after input_from_file for fine_input_stream and set it to a value of 2 for all domains. This allows you to start the nest at a different time than the initial time.

For a nested run, you run `real.exe` and `wrf.exe` as before. Make sure you link over any necessary met_em.d02 (or .d03, etc.) files to the working directory before running real.exe.

Advanced Uses: Time-Varying SST, Sea Ice, Albedo, and Vegetation Data

If you desire to use time-varying surface datasets, new entries must be added to namelist.input

prior to running real.exe and wrf.exe. The first three of these new entries go at the bottom of the &time_control section (i.e., before the ending /) and take the form:

```
io_form_auxinput4    = 2
auxinput4_inname     = "wrflowinp_d<domain>"
auxinput4_interval   = 360, 360, 360,
```

Note that auxinput4_inname should appear exactly as it does above. If the input data are available at a different frequency than every 6 h, change auxinput4_interval to match this frequency (where auxinput4_interval is given in minutes).

A new entry must also be added to the &physics section of namelist.input. This takes the form:

```
sst_update = 1
```

Note that sf_ocean_physics should be set to 0 if this option is activated.

Other Advanced Uses

Recent versions of the WRF-ARW model have added a number of additional options to the model, including the ability to use a digital filter to aid in model initialization; apply nudging (to some specified data set) over some specified time interval; use a 1-D or 3-D ocean model primarily for tropical cyclone simulations; and many, many others. In addition, a number of new physical parameterization packages have been added, many of which have their own additional namelist parameters that can be tweaked to (theoretically) improve model performance for specific forecasting applications. A complete listing of all that is possible with the WRF-ARW model is available within Chapter 5 of the WRF-ARW User's Guide. If you are in doubt as to whether a particular option should be utilized for your simulations, or what particular selections following from using such an option should be for your simulations, please ask.

Part IV: Post-Processing

Post-Processing Using ARWpost and GrADS

WRF-ARW may be used with the ARWpost post-processor. ARWpost converts the WRF model output data into a GrADS-readable format. In the same level directory as the WRF and WPS directories, untar (using a `tar –xzvf ARWpost.tar.gz` command) the post-processor code obtained from the WRF-ARW User's Site. This will create a new ARWpost directory.

Now, switch into the ARWpost directory. Next, run the configuration program by typing `./configure`. Choose option 2 for Linux using the Intel compiler. Once configuration is complete, open configure.arwp with a text editor. Change the entry for LDFLAGS in line 35 so that it reads:

```
LDFLAGS         =     -L/raid-11/LS/evans36/software/netcdf-4.6.2/lib -lnetcdf -lnetcdff
```

Save the file and exit the text editor. Next, compile the post-processor by opening up a slurm-shell and then running `./compile` (assuming that all modules have been loaded and environment variables set prior to doing so). If everything goes according to plan, you'll get an ARWpost.exe file in the current directory. At this point, you may reset all of your environment variables to their defaults (such as by logging out and back in again).

Options for the ARWpost program are handled in the namelist.ARWpost file. First, a note about the namelist.ARWpost file. Options that are active in the namelist are found between the &____ label and the ending / for that section; options that are inactive (i.e. commented out) are founded after the /. Make sure that you have the right options set in your active options sections when editing this namelist file. With that said, let's explore the namelist.ARWpost file section by section. In the &datetime section, set the appropriate values for start_date and end_date. The value for interval_seconds will be equal to that from the history_interval variable in the namelist.input file from running the model, except here in seconds instead of minutes. In the &io section, input_root_name refers to where your wrfout_d01* files are located. Set this equal to the full path there plus wrfout_d01 and it will read in the correct file. The output_root_name variable refers to what you want to call the GrADS output control and data files; set these accordingly.

What is output is controlled in the rest of the &io section of the namelist.ARWpost file. There are several options for the variables plot and fields. Generic options are listed by default; other options for plot and fields are given below the mercator_defs variable line. In general, it is probably best to output all available variables, meaning to change plot to 'all_list' and add in all of the variables in the fields definition after mercator_defs to the fields definition immediately below plot. Note that the lines past mercator_defs before &interp are only comment lines; they will not modify your output! I do, however, recommend adding the following line immediately after mercator_defs (but before the /) to split post-processed output into individual files:

```
split_output = .true.
```

There are other options available; see Chapter 9 of the WRF-ARW User's Guide or the README file in the ARWpost directory for more information.

Finally, under &interp, you have the choice of how you want the data output, as controlled by interp_method. The default value of 1 is for pressure level data; 0 is for sigma levels while -1 is for user-defined height levels. The levels to interpolate to, no matter what you choose for interp_method, are given by interp_levels. The default pressure levels are good for most uses; however, you may desire to add additional levels. Examples for pressure and height levels are given in the last three lines of the file; I don't believe you can control specific sigma levels. Finally, if you wish to interpolate data below ground, you can set the extrapolate variable in this section to .true..

Once the namelist file has been edited to your liking, simply run `./ARWpost.exe` on the command line and wait for data to come out. Once it is done and gives you a success message, you can move on to GrADS to view your output. A working installation of GrADS v2.2.1 can be enabled by loading the appropriate module:

```
module use /raid-08/LS/evans36/modules/
module load grads-2.2.1
```

You may wish to add this to your ~/.bashrc file so that it loads automatically on login. These statements tell the supercomputer where the GrADS executables and supplementary data sets (e.g., maps, fonts) may be located.

Post-Processing Using NCL

It is also possible to post-process WRF-ARW data using utilities developed around the NCAR Graphics framework, specifically using NCL. A working version of NCL 6.5 is available on mortimer as a module:

```
module use /raid-08/LS/evans36/modules/
module load ncl-6.5
```

You will also want to add the following entry to your ~/.bashrc file:

```
export NCARG_ROOT=/raid-11/LS/evans36/software/ncl
```

The primary benefit of using NCL rather than ARWpost for WRF-ARW output and visualization is that no post-processing step is necessary; the necessary libraries for reading and manipulating WRF data are built into NCL. The learning curve is somewhat steep, however, particularly if you do not have extensive experience with modern, object-based programming languages. I highly recommend reading the NCL User's Guide (http://www.ncl.ucar.edu) and working through a number of examples to become familiar with NCL before relying on it exclusively for WRF visualization. Also recommended are the WRF Tutorial slides on NCL, available at http://www2.mmm.ucar.edu/wrf/users/tutorial/201901/jaye_ncl.pdf.

Post-Processing Using the Unified Post-Processor (UPP) and GrADS

WRF-ARW model output may also be post-processed using the Unified Post-Processor, or UPP, initially developed by NCEP for use with the GFS and NAM models. It produces output in GRIB format, which can then be read in (after some manipulation) by either GrADS or GEMPAK. The primary advantage to using UPP is the extensive number of derived quantities provided by UPP. The primary disadvantage of using UPP lies in its complexity, particularly with regards to compiling and executing the UPP code.

This guide focuses upon compiling and executing version 3.2 of the UPP code. It assumes that you have added all of the modules loaded to compile and run WRF to your ~/.bashrc file. If you have not done so, please run each module load statement first before continuing.

In the directory in which your WRF and WPS directories reside, issue the following command (all on one line) to download the UPP v3.2 source code:

```
wget https://dtcenter.org/upp/users/downloads/UPP_releases/DTC_upp_v3.2.tar.gz --no-check-certificate
```

You'll need to be on a compute node for this.

Next, unzip and untar this file, (using `tar -zxvf`). Change into the UPPV3.2 directory and run ./configure. You will be presented with a warning that UPP cannot find your WRF directory. Enter 1, to enter the WRF path name manually, then enter the appropriate path name (e.g., mine is in /raid-11/LS/evans36/WRFv4/WRF). You will then be given fourteen compile selections. Choose option #4, "`Linux x86_64, Intel compiler (dm_par)`." Assuming that all of the necessary module files were loaded and environment variables set, type ./compile to begin the compilation process.

Compilation will take a few minutes. To ensure that the compilation completed successfully, check the output on the screen for any errors. Messages akin to "undefined reference to ext_pnc…" and "undefined reference to ext_gr2…" can be ignored; these relate to WRF-ARW pnetcf and grib2 support, respectively. You also will want to check to ensure that a total of sixteen files are present within two directories:

```
UPPV3.2/lib/:  libbacio.a,  libCRTM.a,  libg2.a,  libg2tmpl.a,
libgfsio.a, libip.a, libnemsio.a, ibsfcio.a, libsigio.a, libsp.a,
libw3emc.a, libw3nco.a, libxmlparse.a

UPPV3.2/bin/: unipost.exe, ndate.exe, copygb.exe
```

Presuming that compilation completed successfully, you are ready to read and edit the UPP namelist file, `UPPV3.2/parm/wrf_cntrl.parm`. This file controls the fields that UPP will post-process and the vertical (isobaric) levels on which post-processed data will be available. The first three lines of `wrf_cntrl.parm` can safely be ignored. The remaining lines, two for each output field, control what will be processed and on what vertical levels it will be processed. The first line for each record looks like this:

```
(DWPT TEMP ON P SFCS ) SCAL=( 4.0)
```

The field controlled by that record is listed within the first set of parentheses. In this example, dewpoint (DWPT) temperature (TEMP) on pressure (P) surfaces (SFCS) is the output field. A full listing of output files is available on pages 18 through 33 of the UPP User's Guide. The remainder of this line, the `SCAL=( 4.0)`, controls the accuracy of floating-point output. In this example, `SCAL=( 4.0)` states that the output data will be accurate to within 1/16 (i.e., 1/2^4). Usually, it is fine to leave this number as-is; however, if you wish for there to be greater precision to a given field, set this value higher (e.g., 5.0 or 6.0).

The second line for each record looks like this:

```
L=(00000 01001 01010 10101 01010 10101 01010 11111 11111 10000
                  00000 00000 00000 00000)
```

This line controls the vertical levels on which post-processed data will be available. A value of 0

indicates NO output at a given level; a value of 1 indicates that output WILL be provided at a given level. For surface-level output, only the first digit is of interest. For pressure-level output, the first 47 digits are of interest. In order, these levels are 2, 5, 7, 10, 20, 30, 50, 70, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, and 1000 hPa. The isobaric levels on which you choose to output data, and the fields that you desire to output, will vary depending upon your specific application.

For model-level output, the first N digits are of interest, where N is the number of vertical levels used within the model simulation (i.e., e_vert in your `WRF/run/namelist.input` file). Other variables for specific applications (e.g., soil data, PBL averages, etc.) have slightly different specifications; please see pages 10 and 11 of the UPP User's Guide for more details.

Decide which model fields you desire to output, modify `wrf_cntrl.parm` appropriately, and then save the file. Once this is done, you are ready to post-process the data. To do so, move into the `UPPV3.2/scripts` directory and open the `run_unipost` script with your favorite text editor. A number of environment and script variables must be edited, including:

- `export TOP_DIR` (line 75) – the full directory path to your UPPV3.2 directory.
- `export DOMAINPATH` (line 76) – the full directory path you some working directory where, ideally, you want your post-processed data to reside. You'll want to make sure that this folder exists.
- `export WRFPATH` (line 77) – the full directory path to your WRF installation (the WRF directory).
- `export modelDataPath` (line 81) – change this to the full path to your wrfout_d01 file(s).
- `export startdate` (line 93) – change this to the start time of your WRF-ARW model forecast, in YYYYMMDDHH format.
- `export fhr` (line 94) – usually can keep this at 00 unless you wish to start post-processing at some time after model initialization.
- `export lastfhr` (line 95) – change this to the length (in hours) of your WRF-ARW model forecast or the last desired forecast output time.
- `export incrementhr` (line 96) – change this to match how frequently (in hours) you wish to have post-processed model output.

You will also want make sure that lines 107 and 108 are commented out (# in front) and that line 104 is uncommented (no # in front). We will run this job on one processor on a compute node, and have compiled it with distributed memory access. But, for the job submission to work correctly, we submit it like a serial job.

Once this is done, we are ready to post-process the data. First, create a job submission script (perhaps named process.lsf) in your UPPV3.2/scripts/ directory that looks like this:

```
#!/bin/bash
#SBATCH -J process
```

```
#SBATCH -o process.log
#SBATCH -n 1
module use /sharedapps/ICC/modulefiles/
module load icc/15.2
module load openmpi/1.10.7
module use /raid-08/LS/evans36/modules/
module load netcdf-4.6.2
module load grib2-libs
module load hdf5-1.10.4
cd /path/to/UPPV3.2/scripts
mpirun ./run_unipost
```

Make sure that you modify the cd statement above with the appropriate directory path.

Despite whatever you may set for your `DOMAINPATH` variable above, output will likely appear in your `UPPV3.2/scripts/` directory as `WRFPRS_d01.##`, where ## = forecast hour. If all goes well, you'll get output files for each forecast time requested. This may take a few minutes to an hour or more, particularly for lengthy, high-resolution model simulations. Afterward, you may safely move the `WRFPRS_d01*` files to the directory that you entered as the `DOMAINPATH` variable; in fact, I recommend doing in order to facilitate analysis of multiple cases.

Once this has completed, we need to create GrADS control files for these data. First, load the GrADS module (see the ARWpost section above). Next, issue one of the following commands from your data/working directory:

```
./grib2ctl.pl WRFPRS_d01.%f2 > (casename).ctl (if max fcst hour < 100)
./grib2ctl.pl WRFPRS_d01.%f3 > (casename).ctl (if max fcst hour > 100)
```

where you will want to replace (casename) with some descriptive name for the dataset. Once this has completed, the (casename).ctl file will contain a nearly-complete GrADS control file. Before reading it in, however, we need to edit one line within this file. Open it up with your favorite text editor, then go to the line that starts with 'tdef 1'. Change the 1 to the number of forecast times (i.e., the number of `WRFPRS_d01*` files). Change the '1mo' at the end of this line to the time increment between files (e.g., 1hr, 3hr, etc.).

Next, we need to create a "map" for the output data file so that GrADS can read it in. To do so, issue the following command from this same directory:

```
gribmap -e -i (casename).ctl
```

where you replace (casename) with whatever you used as the prefix for your .ctl file. Once this has completed, you are ready to read in the data using GrADS.

Other Post-Processing Systems

Other post-processing systems for WRF-ARW output do exist. VAPOR is a self-contained tool developed by UCAR for graphics-intensive multi-dimensional visualizations; in many ways, it is

the natural successor to the deprecated Vis5d package. Information on VAPOR may be found in Chapter 9 of the WRF-ARW User's Guide. Newer utilities such as wrf-python exist to visualize and process WRF-ARW output using Python are also available. WRF-ARW model output may also be used with a number of other utilities, such as the HYSPLIT trajectory and dispersion model and LAGRANTO trajectory tool. If you are interested in using WRF-ARW model output with such a utility, I encourage you to consult the documentation for that utility to understand what is necessary to be able to do so.

Conclusions

If you have any questions with WRF model installation, setup, or debugging, please feel free to ask me via e-mail (UWM affiliates only; otherwise, please contact wrfhelp@ucar.edu).