Using the WRF-ARW on UWM's Updated *mortimer* HPC
Guide for WRF-ARW Version 4.5.1
December 2023

Introduction

This guide is designed to facilitate basic compilation and execution of WRF-ARW v4.5.1 using the Intel compiler suite and OpenMPI on UWM's updated *mortimer* HPC. I cannot cover every option available to you when installing or running the model; for the gory details, please see the WRF-ARW User's Guide, available online at:

Welcome to WRF User's Guide! — WRF Users Guide documentation (ucar.edu)

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

Much of this guide is general to basic terminal and supercomputing environments.

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

Compute Nodes: We will compile WRF-ARW on a compute node. To interactively connect to a compute node, issue the following command:

```
srun --nodes=1 --mem=8g --pty --preserve-env $@ $SHELL -l
```

The remainder of this guide assumes that you are on a compute node and have completed all of the steps in this "What You Need to Know First" section.

Compile Environment: We wish to start with a clean compile environment. To do so, purge all modules that you may currently have loaded:

```
module purge
```

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

```
module load icc/20.4
```

MPI Libraries: You will use OpenMPI to run parallelized (multiple CPU) jobs on the cluster. To load the OpenMPI module, run the following after loading the Intel compiler module:

```
module load openmpi/4.1.2
```

NetCDF and HDF5 Libraries: Before compiling and executing the model code, you'll need to point to netCDF and HDF5 installations on the server. This is also done using modules:

```
module use /raid-11/LS/evans36/modules
module load netcdf-4.9.2
module load hdf5-1.14.3
```

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-1.12.3
```

GRIB2 Libraries: To use GRIB2-formatted data as inputs to WRF-ARW (which you will need to do), you must also load the grib2-libs module:

```
module load grib2libs-dec2023
```

Setting the Stack Size: The stack size on Linux and Unix machines controls the default amount of memory that is allocated to new processes (or threads). The default value for the stack size is too low for WRF-ARW, however. We can set it to be unlimited using the following:

```
ulimit -s unlimited
```

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

```
nano ~/.bash_profile
```

In the nano text editor, simply add the commands from above to the end of the file, save the file (Ctrl-O), exit (Ctrl-X), and then log out and log back into 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

<u>*Login Details*</u>: *mortimer* is accessed via SSH as follows:

```
ssh <user>@submit.mortimer.hpc.uwm.edu
```

Activities such as compiling model code can be performed on the same node to which you log in or by submitting the compilation to the job scheduler.

<u>*Remote Access*</u>: *mortimer* is accessible *only* when on-campus or when using the campus VPN. To download the VPN software, visit https://remote-access.uwm.edu/, login with your ePanther credentials, and download and install the appropriate GlobalProtect VPN client. You will need to provide it with your ePanther credentials when you first log in; you will also need to use the Microsoft Authenticator app for multifactor authentication each time you log in.

<u>*Data Transfer*</u>: To transfer data from *mortimer* to your laptop or desktop using sftp or scp, you must directly connect to the disk on which your data reside. If your data reside on /raid-08, this is raid-08; if your data reside on /raid-11, this is raid-11. This is done as follows:

```
sftp <user>@<disk>.mortimer.hpc.uwm.edu
```

where <user> is replaced with your username and <disk> is replaced with the disk (raid-08, raid-11, etc.) on which your data reside.

<u>*Visualization*</u>: 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 best accessed directly in a distinct SSH session:

```
ssh -Y <user>@vis-1.mortimer.hpc.uwm.edu
```

Part I: Obtaining & Compiling the Model

Obtaining the WRF Model Code

You can obtain 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 in your "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 (v4.5.2 at the time of this writing) and WPS (v4.5 at the time of this writing).

Installing the WRF Model Code

Once you have cloned the WRF repository, you need to switch into the newly created WRF directory. First, issue "`./configure`" (without the quotes) to the command line to start model configuration. A list of configuration options will appear. You 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.

Next, we need to make a minor edit to `configure.wrf`. Open this in nano or another text editor, then move to line 171 where the variable `FC` is defined. Remove the word 'time' in front of `$(DM_FC)`, then save the file and exit the text editor.

If you wish to use WRF's moving nest capabilities, heed the warning message that appears after `configure` finishes and copy `share/landread.c.dist` to `share/landread.c` before continuing.

If you wish to use this WRF installation with the Data Assimilation Research Testbed (DART) software, a couple of additional edits to `Registry/Registry.EM_COMMON` are needed. Open that file with a text editor. Search (Ctrl-W) for h_diabatic. Change rdu to i012rhdu. Next, search for refl_10cm. Change hdu to i012rhdu. Save this file, exit the text editor, and return to the main WRF directory.

We're now ready to compile the model. To do so, issue the following command:

```
./compile em_real
```

Compilation will take approximately 60-90 minutes to complete. The compile process may appear to hang for several minutes when compiling several parts of the model code, especially on the cumulus parameterization driver and a module called first_rk_step_part2; this is expected. Successful compilation will produce "`ndown.exe`," "`real.exe`," "`tc.exe`," and "`wrf.exe`" files in `WRF/main`.

Installing the WRF Pre-Processor (WPS) Code

Installing the WPS code follows a similar process to installing the WRF model code. 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.

If the grib2libs-dec2023 module is not loaded (e.g., you logged off of mortimer after compiling WRF and did not add the module load command to your ~/.bash_profile file), lead it before proceeding.

If you are not already on a compute node, use slurm-shell to connect to one. Run "`./configure`" 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).

Next, compile the code (`./compile`). Compilation should take 1-3 minutes. Once compilation has completed, look for `geogrid.exe`, `ungrib.exe`, and `metgrid.exe` in the current directory.

Part II: WPS

What does WPS 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 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 in the "Basics for Running the Model" section of the WRF-ARW Online Tutorial, available at:

ARW OnLine Tutorial (ucar.edu)

Step 1: Defining a Domain

Defining a domain is done through `geogrid.exe`. Options for the domain are set in namelist.wps. 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 `'/tank/data/LS/evans36/WRFv4/geog'`. Once these variables are set, save namelist.wps and return to the command line.

Finally, run `geogrid.exe`:

```
./geogrid.exe
```

You can safely ignore any "Failed to create UCP worker" message that might appear shortly after running this program. 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. If you are using multiple domains, there should be a `geo_em.d0#.nc` file for each domain.

Step 2: Getting Model Data

Extracting model data from GRIB files is accomplished through `ungrib.exe`. 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 ungrib, clear out all old `GRIBFILE.`, `FILE:`, and `PFILE:` files that may exist to avoid inadvertent errors when running the model. Finally, run `ungrib.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 namelist.wps.

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 and supercomputer configuration takes the form:

```
#!/bin/sh
#SBATCH -n ##
#SBATCH --mem-per-cpu=2gb
module load icc/20.4
module load openmpi/4.1.2
module use /raid-11/LS/evans36/modules/
module load grib2libs-dec2023
module load netcdf-4.9.2
module load hdf5-1.14.3
module load pnetcdf-1.12.3
ulimit -s unlimited

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. The mem-per-cpu directive specifies how much memory per CPU to allocate to compile WRF/WPS; in general, `metgrid.exe` (and `real.exe` and `wrf.exe` to come) only requires <2 GB of RAM per processor. The last line uses the `mpirun` program to run `metgrid.exe` on the earlier-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 simulation durations and/or larger numbers of domain grid points. You may check on its progress by examining the `metgrid.log.0000` file in the WPS directory, or by using `squeue -u <username>` (replace `<username>` with your username, without the brackets) to look for your 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 namelist.wps, 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 of `e_we` and `e_sn` to the desired size values of the inner domain. (Note: the values for these must be an integer multiple of `parent_grid_ratio` plus 1.)
- Change `geog_data_res` as needed, though the default entry will suffice for most uses.
- 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 namelist.wps 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.exe` 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.

duda_wps_advanced.pdf (ucar.edu) (from the January 2021 WRF Tutorial)

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

Namelist.input has several parts, each with multiple variables and multiple options for each of those variables. 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 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 in namelist.wps. Only edit values in the first column (if there are multiple columns for a given variable) for now.

The &time_control section of namelist.input 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 namelist.input is where you will input 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 namelist.input 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:

- 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, respectively. Both rely on radt, which defines how frequently (in minutes) to update the radiation calculation, and this variable 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). 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.
- Boundary-layer parameterizations are specified in bl_pbl_physics. This relies on bldt, which defines how frequently to call the boundary-layer parameterization; setting this equal to zero will match the model time step.
- Cumulus parameterization is handled by cu_physics. This relies on cudt, which defines how frequently to call the cumulus parameterization; setting this to 0 works the same as for bldt.
- 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.

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 uses two programs, real.exe and wrf.exe, to prepare to run and to 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" in the text to "wrf."

```
#!/bin/sh
#SBATCH -n ##
#SBATCH --mem-per-cpu=2gb
module load icc/20.4
module load openmpi/4.1.2
module use /raid-11/LS/evans36/modules/
module load grib2libs-dec2023
module load netcdf-4.9.2
module load hdf5-1.14.3
module load pnetcdf-1.12.3
ulimit -s unlimited

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, or up to 128 if you wish to use one of the 128-processor AMD-based compute nodes. You should strive to use an identical number of processors for real.exe and wrf.exe, however, as well as for all simulation(s) you may conduct for a given 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 execute-#### 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 namelist.wps and namelist.input 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%. This option is discouraged for research applications, however. To use this option, simply add the following line (including the ending comma) immediately below the max_dom option in 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 namelist.input. 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 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 many 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, many new physical parameterization packages have been added, some 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 in Chapter 5 of the WRF-ARW User's Guide. If you are in doubt as to whether a particular option should be used for your simulations, or how it should be configured, please ask.

Part IV: Visualization

The recommended way of visualizing WRF-ARW output is using Python and the wrf-python package. This requires installing a suitable Python distribution (typically Anaconda Python), installing the wrf-python package and other necessary packages (e.g., cartopy for mapping), and then creating and running a Python script that calls upon wrf-python to visualize your data.

We will install the March 202 edition of Anaconda Python, the last one which uses Python 3.10 as its default Python version, since wrf-python has not yet been updated to work with Python 3.11. To install this version of Anaconda Python, copy the following link:

```
https://repo.anaconda.com/archive/Anaconda3-2023.03-1-Linux-x86_64.sh
```

Next, connect to a compute node on *mortimer* and run the following:

```
wget <pasted file link>
```

where <pasted file link> is replaced by the address that you copied above, without the brackets around the address. Once this installer has downloaded, make it executable:

```
chmod 744 <downloaded file name>
```

After doing so, run the executable:

```
./<downloaded file name>
```

You should install this into a directory in your data directory (e.g., `/raid-##/LS/username/anaconda`) rather than your home directory.

Installation completes by asking if you want to initialize the Anaconda Python package. *Do NOT do this!* Anaconda Python contains libraries for programs such as netCDF and the GRIB2 libraries that conflict with those we use with WRF-ARW. Letting the installer initialize Anaconda Python will cause WRF-ARW to not work, requiring us to manually undo everything the installer did to fix the issue. Instead, tell it no, then run the eval command that it provides (changing YOUR_SHELL_NAME in that command to bash), followed by the conda config command that it provides.

To use your Anaconda Python installation in the future, I recommend creating and loading a module file. An example is given in /raid-08/LS/evans36/modules/anaconda-py37; you can copy this file to a directory and with a filename of your choosing and edit it from there. You will want to edit the appsroot entry to match the path to your Anaconda Python installation and the MODULEPATH entry to match the directory into which you copied the module file. Save the file once edited, then exit your text editor. Next, issue the following commands:

```
module use /path/to/modulefile
module load <whatever you named the module>
```

where `/path/to/modulefile` is replaced by the full path to your module file. You will need to do this each time you wish to use Anaconda Python, and *you are strongly encouraged to unload the module (module unload <module name>) when you are done using it to avoid conflicts with WRF-ARW-related modules!*

The two main packages that you will need to add to Anaconda Python to use wrf-python are wrf-python itself and cartopy for mapping. To do so, connect to a compute node, load your Anaconda Python module, then use conda install to install the needed packages:

```
conda install -c conda-forge wrf-python cartopy
```

You may also wish to install other packages (such as metpy) at the same time. Anaconda Python will generally install all necessary supporting packages, such that trying to install just those two packages will often also install other packages at the same time.

Example Jupyter Notebooks containing wrf-python code snippets for a wide range of common use cases are available at:

evans36/wrf-python-notebooks: Jupyter Notebooks demonstrating basic wrf-python usage. (github.com)

In general, you can create your Python scripts on the login node or a compute node if you are using a regular text editor to do so. You need to connect to the visualization node to use a graphical editor such as Spyder to write and edit your code, however. You also need to connect to the visualization node to run your code, whether in Spyder or on the command line.

Conclusions

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