WRF Software and Tools Update

Dave Gill
Brian Bonnlander
Kelly Keene
Jianyu Liu

Kevin Manning
John Michalakes
Jordan Powers
Huang Wei
WRF Problemos ... Est Omnis Divisa in Partes Tres

• Independence from your Sys Admin

• Output in Itty-Bitty Pieces

• Szip or Extended-Rice Algorithm

• Whacking that which be Deemed Unnecessary

• Using “WTF” in polite company

• The Secret Handshake
Building WRF and WPS and Libs

• Kelly Keene has posted and supports a scripting system that builds the required libraries for WPS and WRF, and then builds WPS and WRF

• The libs:
  – NETCDF and MPICH (for WRF and WPS)
  – JPEG, JASPER, PNG, ZLIB (WPS Grib2 support)
Building WRF and WPS and Libs

- The builds are designed for desk-top systems that tend to not have structured system administrator support
- Users must have the standard Unix development environment available, such as make
- Several compilers are available: GNU, INTEL, PGI, as well as the vendor XLF compiler
Building WRF and WPS and Libs

- The user selects from available compilers found by the script and then the package takes 15 – 30 minutes to build the libraries
- The WPS configure.wps file is modified to use the new Grib2 support libraries
- Files with recommended “env” settings are constructed
Building WRF and WPS and Libs

- This is the TOP line in the WRF FAQs
- Available for 3.4.1 and 3.5
- A short README is included in the tar file
Stitching Model Output Together

• Yunheung Wang (CAPS) developed and Kevin Manning improved a scheme that joins “split data” back together

```plaintext
&time_control
history_interval_s = 150, 60, 60,
io_form_history    = 102
/
```
Stitching Model Output Together

• Running on 20 cores could produce the following WRF model decomposition and output:
Stitching Model Output Together

• With large domains, model output can dominate the total wall clock time
• When running on 800 cores, there are 800 files per output time written
• Files get constructed with names such as `wrfout_d01_2010-06-23_15:00:00_0000`
Stitching Model Output Together

• The only purpose is timing performance
• Works well with multiple domains and when restarts overlap with model output times
• The joining program is DM parallel
• For a 2000x2000x100 WSM6 domain, 2 minutes per time period with 8 cores manufactured the single file
• Scripts exist to run the joining program concurrently with WRF
Stitching Model Output Together

• Single file input:
Timing for processing wrfinput file (stream 0) for domain 1:
320.15085 elapsed seconds

• Multiple file output:
Timing for Writing wrfout_d01_2010-06-23_12:00:00 for domain 1:
0.90883 elapsed seconds
NETCDF4 Compression

• Huang Wei and Jianyu Liu have put in a simple way to get impressive NETCDF4 compression with WRF model output

• If the user has NETCDF4 libraries that have HDF5 compression included, then a single “env” variable is all that is required
NETCDF4 Compression

• Prior to running `./configure` ...

```
setenv NETCDF4 1
export NETCDF4=1
```
NETCDF4 Compression

• This is fully supported in WRF 3.5
• File sizes tend to be about half of the original size
• The compression works well with fields which contain similar values (such as near-zero quantities for many of the hydrometeor fields)
• NOTE: All of the NETCDF tools need to support the compression: ncview, ncBrowse, ncl, nco
(De)Selecting Model Output Fields

• Several years ago John Michalakes provided a simple run-time option to add and remove fields from WRF streams

```
&time_control
iofields_filename = "myoutfields.txt"
/

-:h:0:W,PB,P
```
(De)Selecting Model Output Fields

- Particularly helpful when ncview shows:

  - (144) 2d vars
  - (21) 3d vars
(De)Selecting Model Output Fields

• Removing half of the unwanted or never used 3d arrays cuts your file sizes in half
• Default values for “history” that are in the Registry do not obligate users
WRF Testing Framework: WTF

- Brian Bonnlander has put together a testing framework for the WRF model
- Runs on yellowstone and janus (large Linux systems with queues)
- Ported to Darwin desktops
- Simple README documentation
WRF Testing Framework: WTF

• Lots of “different core count” tests conducted:
  - em_b_wave - 10 (serial, SM, DM)
  - em_quarter_ss – 16 (serial, SM, DM)
  - em_chem – 6 (serial, DM)
  - em_real – 25 (serial, SM, DM) + 27 (serial, DM)
  - nmm_real – 9 (serial, DM)

• With GNU + PGI + Intel, over 600 tests and 400 bit-wise comparisons conducted weekly
WRF Testing Framework: WTF

• The WTF script system is available for download
• Idea is to eventually test all physics and dynamics options in WRF
• For version 3.6, code contributors will be required to run WTF and other tests
• http://www.mmm.ucar.edu/wrf/users/testing.html
Helpful Policy Documents

• Several existing documents have been updated to reflect the latest policies for getting software into WRF
• These include lists of tests, recommendations, best practices, schedules, and the mechanics of proxy interaction with the WRF repository
Helpful Policy Documents

• Testing policies

http://www.mmm.ucar.edu/wrf/users/testing.html
Helpful Policy Documents

• WRF Software Administration
  
  http://www.mmm.ucar.edu/wrf/users/code_admin.html
Helpful Policy Documents

• Coding standards for contribution

http://www.mmm.ucar.edu/wrf/users/contrib_info.html
Helpful Policy Documents

• The WRF team is intending to offload much of the preliminary testing of delivered software back onto the original contributors
• Automated testing and simple case studies are provided on the testing web pages
• Some minimal coding standards are going to be (more strictly) enforced
• Purpose: reduce the amount of time it takes to get code into the WRF repository
The Big Three Issues

• Building WRF

• Reducing some WRF bulkiness

• Getting code into WRF