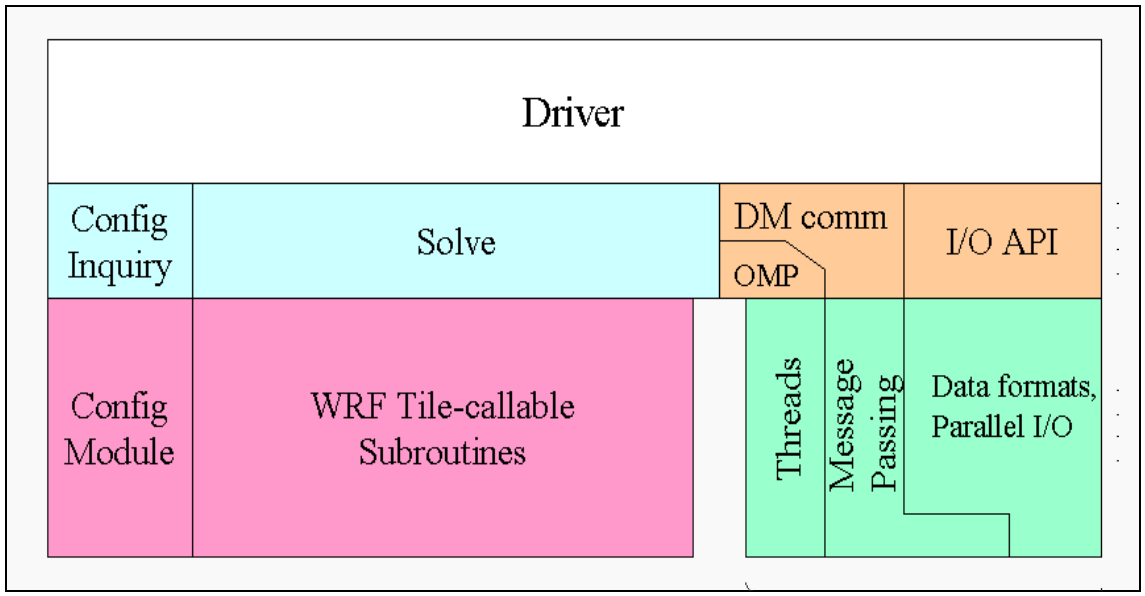

Registry

WRF Software

Dave Gill

- WRF Software Status
- New Infrastructure Features
- Recent Notable Community Efforts
- Best Practices for Developers
- Future Plans

- WRF Software Status
  - Where we are
- New Infrastructure Features
  - What we've done for (to) you
- Recent Notable Community Efforts
  - What you've done, where to steal ideas
- Best Practices for Developers
  - What we want you to keep (stop) doing
- Future Plans
  - What we think we ought to do

WRF Software Status

- Large, recent, mature Community model effort
  - 800,000 lines of Fortran distributed throughout cores
  - More than 250,000 lines of generated code
  - Started in 1998 with a blank sheet
  - Developers Committee and Release Committee primarily responsible for the code and availability to community
  - Several years of fairly reliable adherence to policy for software upgrades

WRF Software Status

- Designed to be broadly extensible
  - Tested weekly on several platforms and with several compiler/OS combinations
  - If you have compiler/OS suggestions (or better yet existing options), please send them along
  - Release testing with vendors provides robust assurances on a large number of platforms, and the benefits of performance options
  - WRF has been coupled using several technologies

WRF Software Status

- Designed to be broadly extensible
  - Designed with three layers (framework, model/physics, and mediation) and a strict contract
  - Runs routinely on small computers, and has been shown to scale on suitable problems to 10^5 procs
  - Development over the past couple of years has included support for cheap commodity graphical processors (GPUs)
  - No app yet forthcoming for iPad, though

New Infrastructure Features

- Run-time IO capability

- Edit the `namelist.input` file, the `time_control` namelist record

```
iofields_filename = "myoutfields.txt"
io_form_auxhist7 = 2 (choose an available stream)
auxhist7_interval = 10 (every 10 minutes)
```

New Infrastructure Features

- Run-time IO capability
- Place the fields that you want in the named text file **`myoutfields.txt`**

```
+:h:7:RAINC,RAINNC
```

- Where "+" means ADD this variable to the output stream, "h" is the history stream, and "7" is the stream number

New Infrastructure Features

- Run-time IO capability

```
-:h:0:PH,PHB,W
+:i:0:CANWAT
```

- Users may add or remove fields with "+" or "–"
- The standard input and output streams to utilize are accessed through the "i" or "h" keyword.
- Select a stream number (example: 0=standard model output and input, 1=input to real from WPS)
- Names are the quoted fields in the "state" entries in the Registry.

New Infrastructure Features

- Run-time IO capability

With great power comes great responsibility. This is my gift, my curse.

Spiderman

- Users MUST now include an `io_form` and an `interval` for all input and output files, otherwise the model will quietly ignore your IO request

New Infrastructure Features

- Parallel build
  - Users may set either an env variable or provide an option to the `compile` command
  - Be careful about overloading your system with multiple simultaneous builds

`setenv J 2` (uses two processes for the make)
`setenv J` (disables parallel build)

`./compile -j 2` (uses two processes for make)

Recent Notable Community Efforts

- Hurricane WRF (HWRF) - example of coupling WRF to an ocean model
- Grell cumulus scheme (G3) - example of non-column physics option and associated communications
- WRF Fire - example of regridding cells for selective and arbitrary resolution

Recent Notable Community Efforts

- Nested DFI (hopefully in 3.2.1) - example of working with timers and framework-level grid structures on multiple domains
- Offline Vertical nesting - example of refining vertical levels in different domains (`ndown` only right now)
- Thompson MP scheme - example of parallel computation of look-up table

Best Practices for Developers

- Getting code into the WRF repository
  - Contact wrfhelp@ucar.edu to be issued a liaison on the WRF Developers committee
  - New features are required to be in the model by mid December, with prospective intentions made known by mid September

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - New schemes are a single Fortran module
  - No module-specific IO, except for initialization
  - No `STOP` or `WRITE` statements, use the defined uitilities for debug information or fatal aborts
  - Even if your scheme is column-oriented, wrap a driver with 3-d capability for the entire tile
  - Make use of optional fields if required for different dynamical cores (with appropriate `#ifdefs`)

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - PLEASE use the `package` capability in the Registry file to associate new variables with your scheme to reduce the memory footprint and meaningless output
  - If you know of limitations in your scheme, help us bulletproof the code by using the new `module_check_a_mundo.F` file

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - Test your schemes with both ARW and NMM, single and multiple domains, moving nests, and various other physics combinations

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - Verify that you code is able to reproduce results on various processor counts, both OpenMP and MPI
  - There should be no parallel sections inside the physics/ model layer
  - Restrict model layer interface arguments to be standard Fortran 77 constructs (i.e. no derived data types)

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - Try to stay away from common blocks, even within the code
  - Have no decomposable data in the `CONTAINS` section of the module
  - Utilize the physical constants provided within WRF
  - Review the two WRF software tutorial presentations

Best Practices for Developers

- If you are a physics developer: follow the existing paradigm
  - Provide an appropriate level of user documentation for inclusion in the user guide (a couple of paragraphs suffice)
  - Provide internal documentation (as well as references) inside the code
  - WRF is public domain software, be careful of the licenses that you include in your code

Future Plans

- The community has identified model coupling as an area of focus
  - The WRF model initially used MCEL and MCT to couple with ocean and wave models
  - There is a project currently underway to couple WRF with a land surface package via ESMF
  - The HWRF ocean model coupling is handled cleanly with MPI
- Let us know of capabilities that you require that are unavailable in the WRF infrastructure