Sharing Physics Between WRF and MPAS with CCPP

Dave Gill
Laura Fowler
Ming Chen
Jimy Dudhia
Jihyeon Jang
Wei Wang
Kelly Werner
Cheryl Craig
Steve Goldhaber

NCAR
UCAR

June 9, 2020 **Boulder, CO**

Big Picture: Sharing

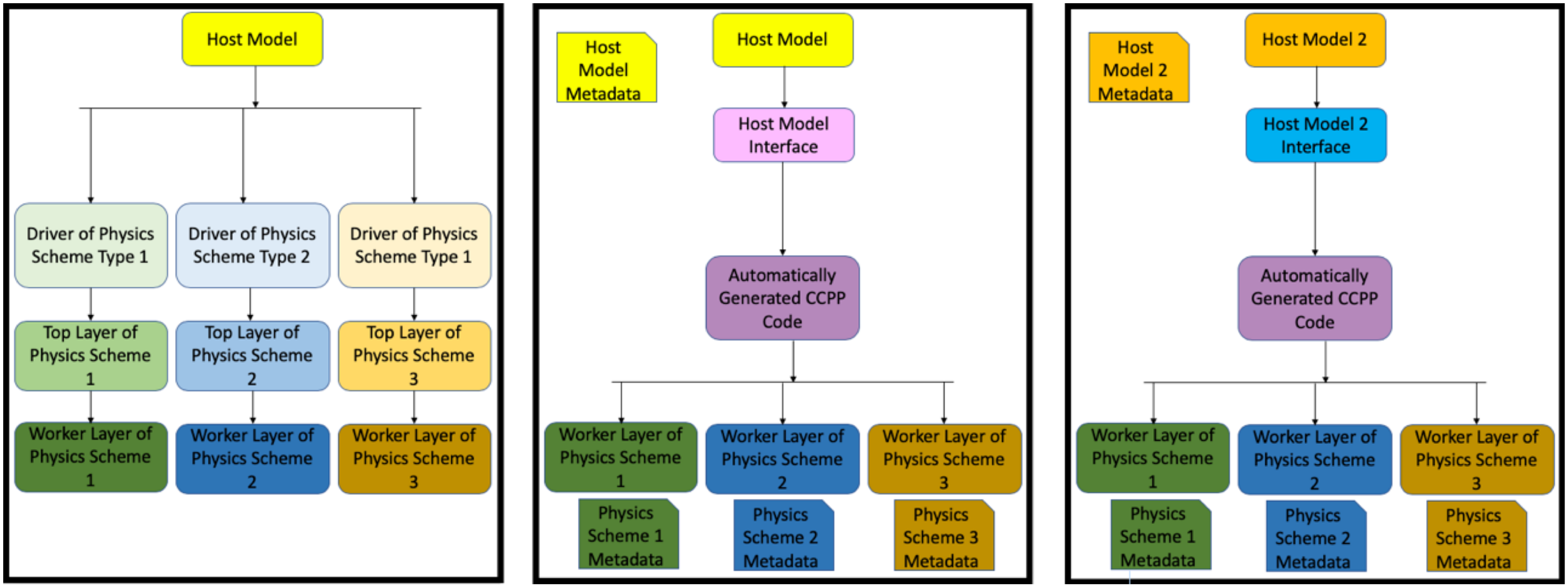
- MPAS and WRF share
  - Support technology: http://forum.mmm.ucar.edu
  - Pre-processor input from WPS
  - Repo management (git) and hosting services (github)
  - Basic tenets of gitflow-esque workflow
  - A couple of physics suites (officially)
  - Hybrid vertical coordinate and moist theta in WRF came from MPAS

- Want to more easily transfer physics between MPAS and WRF
- A goal is to allow physics transfer between MPAS and other models

- Several groups within NCAR (ACOM, CGD, MMM) and NOAA are working on (and are in various stages of implementing) a strategy for more widely sharing physics

CCPP = Common Community Physics Packages

Big Picture: Why are we doing this



Original interface to physics and host model.

The CCPP interface permits the same physics to be used with both the original and a new host model.

Initially Selected Schemes to Port

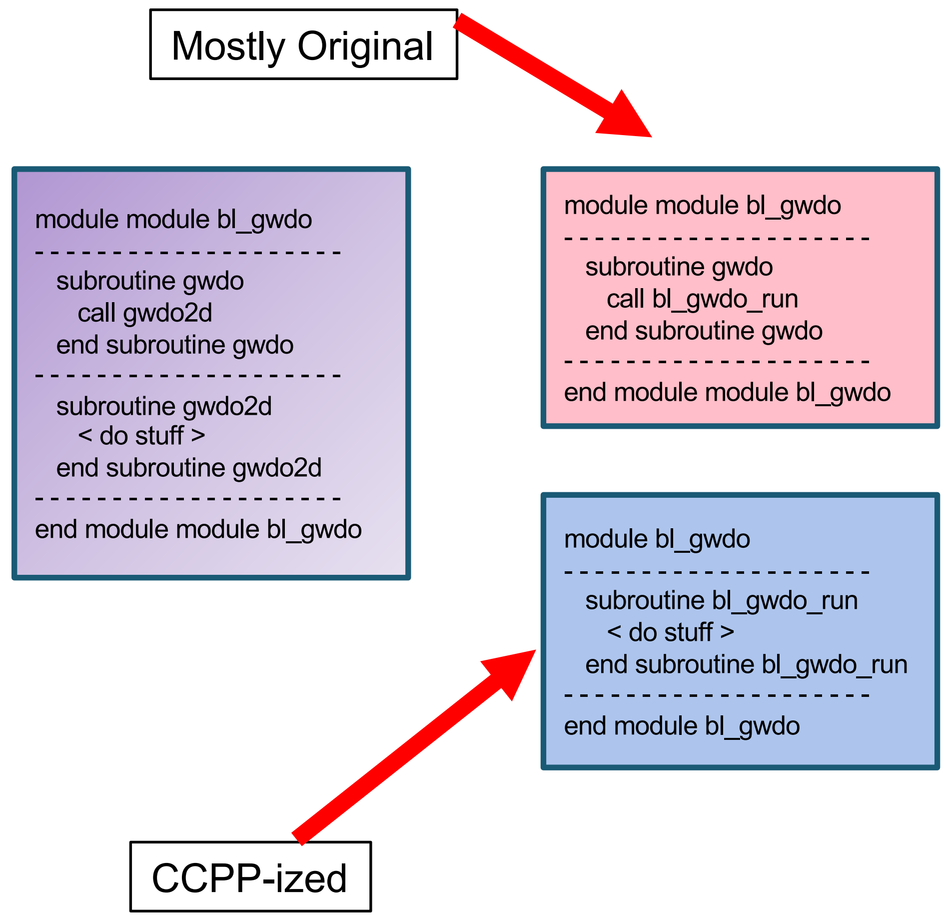
Surface Layer	Xu-Randall Cloud Fraction
YSU PBL	GWDO
New Tiedtke CU	RRTMG LW
WSM6 MP	RRTMG SW

- Choosing to wait for the Land Model.
  - Does not pose troubles for eventual CESM port.
  - Does not pose troubles for MPAS or WRF port.

- CTSM effort is underway at NCAR. LILAC may provide a preferred way to access a supported land model for both MPAS and WRF.

Process / What defines a “Ported Scheme”

- Modifications to scheme for CCPP
  - Split the top-level scheme (usually WRF-centric 3d) apart from the scheme’s 2d (column) routine (still callable from existing MPAS and WRF drivers)
  - Have *_run*, *_init*, *_final*, *_timestep_init*, and *_timestep_final* top-level routine names



Process / What defines a "Ported Scheme"

- Modifications to scheme for CCPP
  - Single horizontal index (if any) in the CCPP-ized routines
  - No horizontal halo-dimensioned fields in CCPP-ized routines

No "j" dimension

No horizontal memory dimensions

```
   real, dimension(its:ite,kts:kte)  , intent(in   ) :: t1, q1, zl
   real, dimension(its:ite,kts:kte)  , intent(in   ) :: prsl
   real, dimension(its:ite,kts:kme)  , intent(in   ) :: prsi
```

Process / What defines a "Ported Scheme"

- Modifications to scheme for CCPP
  - Push as much pre- and post-processing into scheme as makes sense
  - Array assignments in scheme top-level call due to horizontal index reduction are OK and expected
  - Basically: allow removal of top-level of scheme

```
xt24 = mod(xtime+radt*0.5,1440.)
tloctm = gmt + xt24/60. + xlong(i,j)/15.
hrang = 15. * (tloctm-12.) * degrad
xxlat = xlat(i,j) * degrad
```

```
!
   qv2d(its:ite,:) = 0.0
!
```

ZAP: Computation of constants
ZAP: Initializations to zero
ZAP Diagnostics

```
!
     do k = kts,kte
       do i = its,ite
         rthblten(i,k,j) = rthblten(i,k,j)/pi3d(i,k,j)
```

Process / What defines a "Ported Scheme"

- Modifications to scheme for CCPP
  - Interstitial code for helper routines called in top-level (non CCPP-ized) routines

NEW:
Assignments (3d -> 2d)
call ra_rrtmg_sw_mpas_timestep_init
call ra_rrtmg_sw_run
call ra_rrtmg_sw_mpas_timestep_final
Assignments (2d -> 3d)

ORIGINAL:
Starting on line 10000 – 400 lines of diagnostics surrounding the call to SWRAD

Process / What defines a “Ported Scheme”

- General Modifications
  - Clean up INTENT – missing, particularize, wrong
  - Reduce argument list to required minimal set
  - Fix internal comments
  - Single argument *mostly* = single field
    - Unless multiple fields are not distinguishable within scheme
    - Derived types are allowable, but break portability
    - So, interstitial code would be OK for MPAS DDTs
  - Most of the OPTIONAL arguments were not

- A cookbook for developers to follow
  - Automated code depends on an accurate metadata file
  - Constructing the metadata file takes time

The Metadata File

- For each of the CCPP-ized files, a metadata file is required
- For each CCPP-ized subroutine, a metadata section is required in that file
- Each argument in a CCPP-ized routine has a metadata entry

- The mapping of the host model's variables (also with a metadata file) and the physics scheme's variables are through the unique identifier of the field's name
- The ordering of the arguments in the subroutine matches the ordering of the variables in the metadata file

Physics Scheme and Metadata

```
subroutine bl_gwdo_run(      &
    sina, cosa,              &
    rublten,rvblten,         &
    dtaux3d,dtauy3d,         &
    dusfcg,dvsfcg,           &
    uproj, vproj,            &
    t1, q1,                  &
    prsi, prsl, prslk, zl,   &
    kpblmax,                 &
    var, oc1,                &
    oa2d1, oa2d2,            &
    oa2d3, oa2d4,            &
    ol2d1, ol2d2,            &
    ol2d3, ol2d4,            &
    g_, cp_, rd_,            &
    rv_, fv_, pi_,           &
    dxmeter, deltim, kpbl,   &
    its, ite,                &
    kts, kte, kme,           &
    errmsg, errflg           )
```

```
[ccpp-arg-table]
  name = bl_gwdo_run
  type = scheme
[sina]
  standard_name = sine_of_map_rotation
  long_name = sine of map rotation
  units = dimensionless
  dimensions = (horizontal_loop_begin:horizontal_loop_end)
  type = real | kind = kind_phys
  intent = in
  optional = F
[cosa]
  standard_name = cosine_of_map_rotation
  long_name = cosine of map rotation
  units = dimensionless
  dimensions = (horizontal_loop_begin:horizontal_loop_end)
  type = real | kind = kind_phys
  intent = in
  optional = F
```

The Metadata File

- A github repository holds the growing list of known fields (such as dimensions) and conventionally accepted names:

https://github.com/ESCOMP/CCPPStandardNames/blob/master/Metadata-standard-names.md

- Fields with vertical indexing may have “layers” (default) and “levels” (explicitly stated in name)
- Horizontal indexing allows full size (without halo) and threaded size
- The list of known constants is increasing

- Construction of names is often required

The Suite Definition File (SDF mostly TBD)

```
<suite name="MPAS_meso_ref_rad" version="1.0">
   <group name="radiation">
      <scheme>cld_xurandall</scheme>
      <scheme>ra_rrtmg_sw_mpas</scheme>
      <scheme>ra_rrtmg_sw</scheme>
      <scheme>ra_rrtmg_lw_mpas</scheme>
      <scheme>ra_rrtmg_lw</scheme>
   </group>
</suite>

<suite name="MPAS_meso_ref_sfc" version="1.0">
   <group name="surface_layer">
      <scheme>sf_sfclay</scheme>
   </group>
</suite>

<suite name="MPAS_meso_ref_pbl" version="1.0">
   <group name="boundary_layer">
      <scheme>bl_ysu</scheme>
      <scheme>bl_gwdo</scheme>
   </group>
</suite>

<suite name="MPAS_meso_ref_cu" version="1.0">
   <group name="cumulus">
      <scheme>cu_ntiedtke</scheme>
   </group>
</suite>

<suite name="MPAS_meso_ref_mp" version="1.0">
   <group name="microphysics">
      <scheme>mp_wsm6</scheme>
   </group>
</suite>
```

```
<suite name="mesoscale_reference" version="1.0">
   <group name="physics_pre_LSM">
      <suite name="MPAS_meso_ref_rad" group="radiation" />
      <suite name="MPAS_meso_ref_sfc" group="surface_layer" />
   </group>
   <group name="physics_post_LSM">
      <suite name="MPAS_meso_ref_pbl" group="boundary_layer" />
      <suite name="MPAS_meso_ref_cu" group="cumulus" />
   </group>
   <group name="physics_post_dynamics">
      <suite name="MPAS_meso_ref_mp" group="microphysics" />
   </group>
</suite>
```

The “group” element allows MPAS to schedule physics in a requested order

The “group” attribute allows other host models to locate where the call to a particular scheme should go in the time step

Changes MPAS

- The WRF model will not be converted to a CCPP-ized host model

- MPAS will have a transition period, allowing access to physics by the original driver and the new CCPP option
- The existing top-level drivers of each scheme remain in place
  - These are not required when CCPP is used
- Rolling the calls to the *_to_MPAS* and *_from_MPAS* routines into interstitial routines
  - Interstitial routines are by definition associated with the host model
  - Maintains bit-for-bit results

Easy Changes to MPAS to “interface” and “drivers”

mpas_atmphys_interface

- (k,i) -> (i,k,j) -> (i,k) -> (i,k,j) -> (k,i)

mpas_atmphys_interface_ccpp

- (k,i) -> (i,k) -> (k,i)

```
if(.not.allocated(psfc_p) ) allocate(psfc_p(ims:ime,jms:jme) )

rho_p(i,k,j) = zz(k,i) * rho_zz(k,i)
```

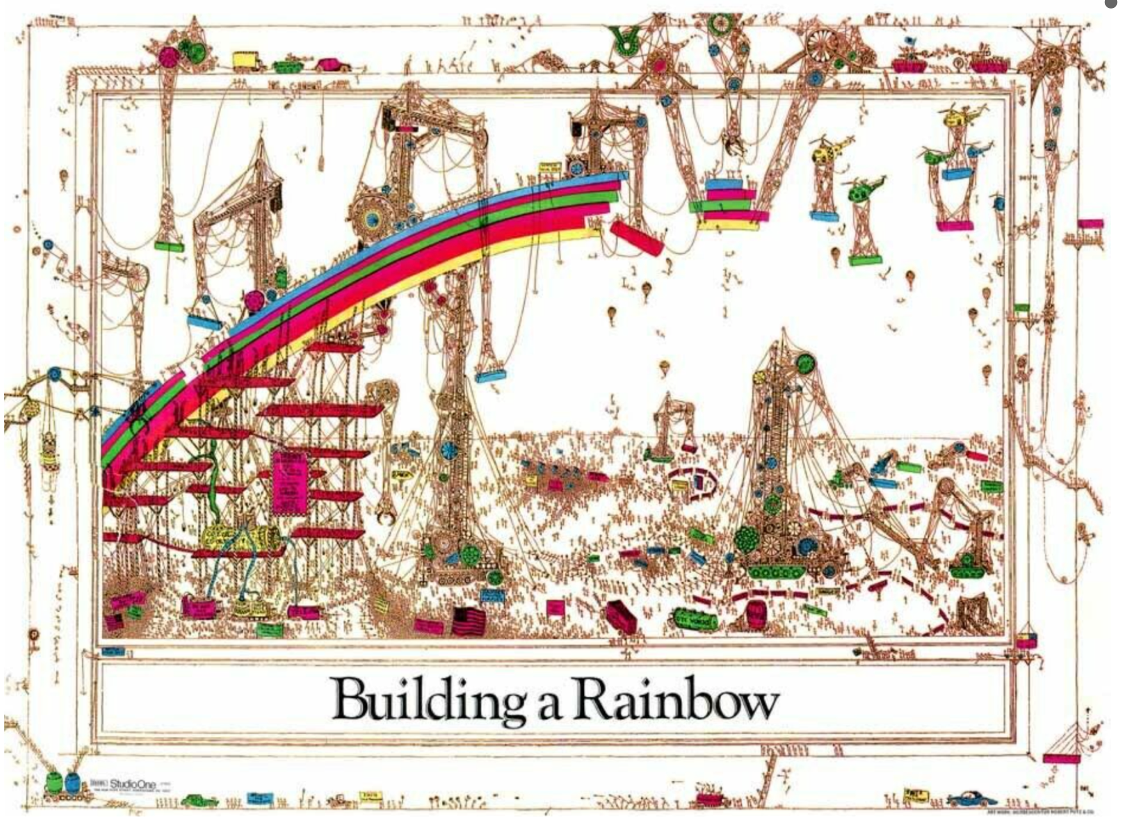
```
if(.not.allocated(psfc_p) ) allocate(psfc_p(ims:ime)         )

rho_p(i,k) = zz(k,i) * rho_zz(k,i)
```

Big Picture: What we are doing

- Scheme sharing between MPAS and WRF
- Purpose is to get MPAS and CAM physics available for each other
- Using a jointly developed utility to provide Common Community Physics Packages (CCPP)
- Conventionalize the top-level physics schemes
- Describe the arguments in the top-level physics schemes in a separate metadata file
- Describe the available fields from the host model is a metadata file
- Map the available fields from the host model and those required by the physics schemes

Status: Importance of Verb Tense

- Started and stopped a few times to get our bearings and scope
- All first suite physics is ported, including non-conventionalized metadata files
- Getting all metadata and CCPP-ized routines to be uniform
- Merging all separate git repositories into a single location



- Figuring out the data flow for the physics diagnostic data, leaning towards including / wrapping calls inside of interstitial routines