CCPP Training
College Park, MD, March 12-13, 2019

Building NEMSfv3gfs with CCPP

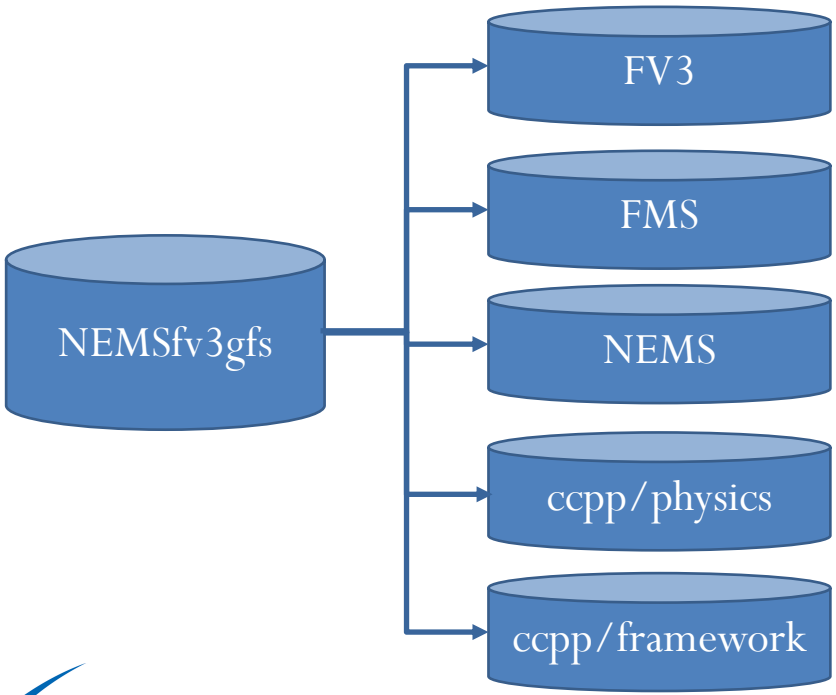
Laurie Carson
Global Model Test Bed

Building NEMSfv3gfs with CCPP

- Code repositories and how to get the code
- Types of builds
- System requirements, compilers, libraries
- How-to instructions (see also: Practical Session exercises)

Code repositories

- The repository structure for CCPP development in NEMSfv3gfs mirrors the Vlab repository structure, with the addition of the CCPP repositories



Repository (GMTB development version)	Branch name
https://github.com/NCAR/NEMSfv3gfs	**gmtb/ccpp**
https://github.com/NCAR/FV3	gmtb/ccpp
https://github.com/NCAR/ccpp-physics	master
https://github.com/NCAR/ccpp-framework	master
https://github.com/NCAR/NEMS	gmtb/ccpp
https://github.com/NCAR/FMS	GFS-FMS

How to get the code

- The authoritative repositories are located on github.com in the NCAR organizational space
  - Some repositories are private (NEMSfv3gfs, FV3, NEMS)
  - Some repositories are public (ccpp-physics, ccpp-framework, FMS)
  - Send a request to gmtb-help@ucar.edu to request access to the private repositories
- Clone a local copy of the repository to begin working, including submodules

```
git clone --recursive -b emc_training_march_2019  https://github.com/NCAR/NEMSfv3gfs
```

Types of builds

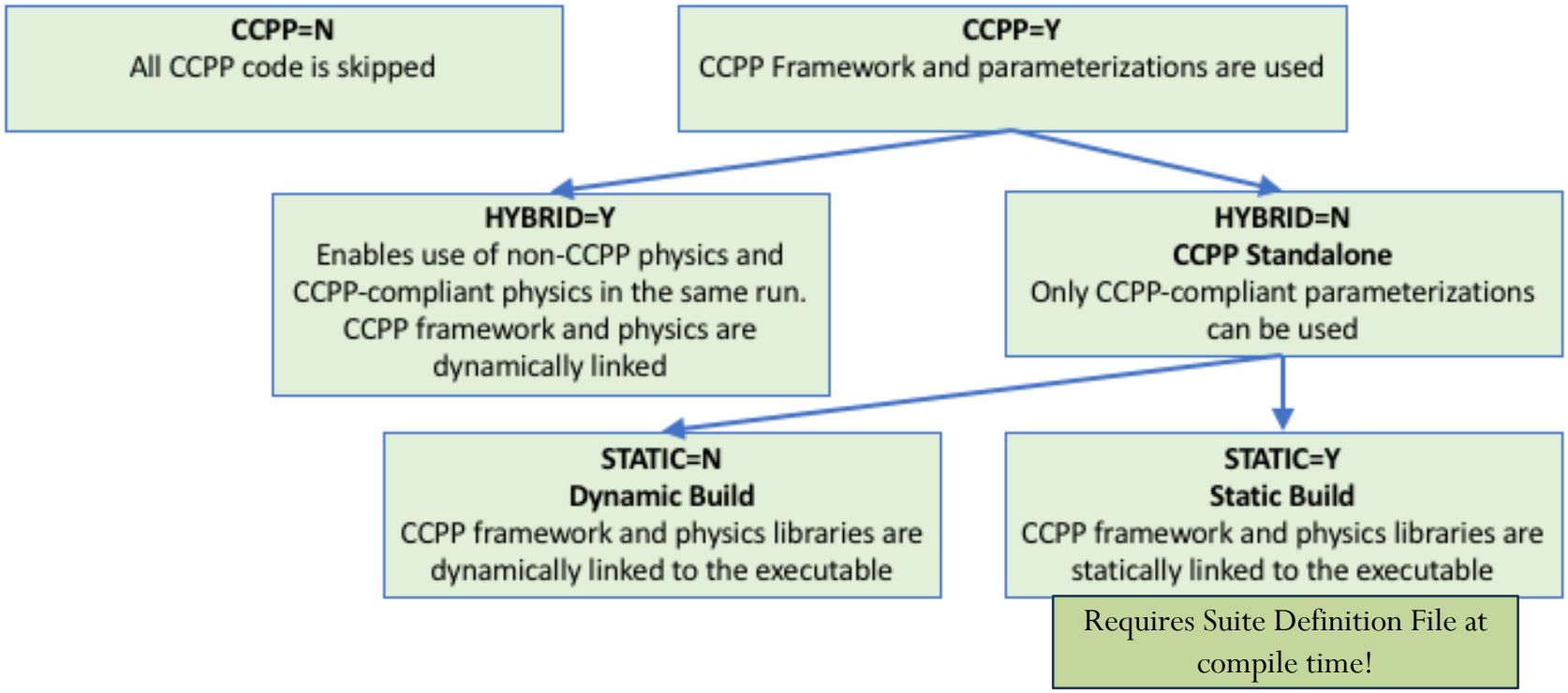
Using FV3GFS as the host model, there are several modes of operation

- **Non-CCPP:** The code is compiled without CCPP enabled and matches the official NEMSfv3gfs codebase in VLab.
- **Hybrid CCPP:** The code is compiled with CCPP enabled and allows users to combine non-CCPP physics and CCPP-compliant physics.
- **Standalone CCPP (non-Hybrid)**: The code is compiled with CCPP enabled and restricted to CCPP-compliant physics.
  - **Dynamic CCPP:** This option is recommended for research and development users, since it allows users to change physics schemes at runtime by making adjustments to the CCPP suite definition file and the model namelist.
  - **Static CCPP**: The code is compiled with CCPP enabled but restricted to CCPP-compliant physics called by a suite that is chosen at compile time.

This selection is specified by supplying various command-line options to the **compile.sh** script

compile.sh $PWD/../FV3 theia.intel "CCPP=Y HYBRID=N STATIC=Y SUITE=suite_FV3_CPT_advanced.xml"

CCPP-Relevant FV3GFS Build Options



Default is CCPP=N
If CCPP=Y, defaults are:
HYBRID=Y
STATIC=N

Non-CCPP

- **CCPP=N**
- The code is compiled without CCPP enabled and matches the official NEMSfv3gfs codebase in VLab.
- This option entirely bypasses all CCPP functionality and is only used for regression testing against the unmodified NEMSfv3gfs codebase.

Hybrid CCPP

- **CCPP=Y HYBRID=Y**
- The code is compiled with CCPP enabled and allows a user to combine non-CCPP physics and CCPP-compliant physics.
- Restricted to parameterizations that are termed as "physics" by EMC, i.e. that in a non-CCPP build would be called from **GFS_physics_driver.F90**.
- Parameterizations that fall into the categories "time_vary", "radiation" and "stochastics" have to be CCPP compliant.
- The hybrid option is fairly complex and ***not recommended*** for users to start with and is intended as a ***temporary*** measure for research and development until all physics are available through the CCPP.
- This option uses the existing physics calling infrastructure (**GFS_physics_driver.F90**) to call either CCPP-compliant or non-CCPP-compliant codes within the same run.
- The CCPP framework and physics libraries are dynamically linked to the executable for this option.

Standalone CCPP, dynamic build

- **CCPP=Y HYBRID=N STATIC=N**
- Any parameterization to be called as part of a suite must be available in CCPP.
- Physics scheme selection and order is determined by an external suite definition file (SDF). This is used only at RUN time.
- The existing physics-calling code (**GFS_physics_driver.F90, GFS_radiation_driver.F90**) are bypassed altogether in this mode and any "glue" code previously contained therein is executed from a CCPP-compliant "interstitial scheme".
- **Dynamic CCPP**
  - CCPP framework and physics libraries are dynamically linked to the executable
  - All physics parameterizations are compiled into the library
  - This option is recommended for research and development users, since it allows to change physics schemes at runtime by making adjustments to the CCPP suite definition file and the model namelist.
  - This option carries computational overhead associated with the higher level of flexibility.

Standalone CCPP, static build

- **CCPP=Y HYBRID=N STATIC=Y SUITE="xyz.xml"**
- Any parameterization to be called as part of a suite must be available in CCPP.
- Physics scheme selection and order is determined by an external suite definition file (SDF). This is used at COMPILE time and RUN time.
- The existing physics-calling code (**GFS_physics_driver.F90, GFS_radiation_driver.F90**) are bypassed altogether in this mode and any "glue" code previously contained therein is executed from a CCPP-compliant "interstitial scheme".
- **Static CCPP**
  - CCPP framework and physics libraries are statically linked to the executable
  - The code is restricted to CCPP-compliant physics called by a suite that is chosen at compile time.
  - This option is recommended for advanced users and operational applications, since it limits flexibility in favor of runtime performance and memory footprint.

System requirements, compilers, libraries

- FORTRAN 90+ compiler (ifort v15+, gfortran v5.4+,pgf90 v17.9+)
- C compiler (icc v15+, gcc v5.4+, pgcc v17.9+)
- cmake v2.8.11+
- netCDF v4.x (not v3.x) with HDF5, ZLIB and SZIP
- Python v2.x (not v3.x)
- Libxml2 (tested with 2.2 and 2.9.1)
- The Earth System Modeling Framework (ESMF), the SIONlib, the NCEPlibs, and the netCDF libraries must be built with the same compiler as NEMSfv3gfs.
- The available compilers are currently Intel, PGI, and GNU.

System requirements, compilers, libraries

- NCEP libs (pre-installed on theia, jet, Cheyenne)*

Library name/version	Description
bacio	NCEP binary I/O library
ip	NCEP general interpolation library
nemsio	NEMS I/O routines
sp	NCEP spectral grid transforms
w3emc	NCEP/EMC library for decoding data in GRIB1 format
w3nco/v2.0.6	NCEP/NCO library for decoding data in GRIB1 format

* these libraries are pre-built with the Intel compiler and OpenMP and –fPIC flags to support CCPP. They are not the same builds as those available as modules on theia, jet

System requirements, compilers, libraries

- Other system libs

Library name/version	Description
ESMF v7.1.0r	Earth System Modeling Framework for coupling applications – used for the NUOPC layer for NEMSfv3gfs, NOT for CCPP!
netCDF v4.x	Interface to data access functions for storing and retrieving data arrays
SIONlib (optional) v1.7.2	Library (link) used to read precomputed lookup tables instead of computing them on the fly

How-to instructions

- See Practical Session instructions at:
- https://dtcenter.org/community-code/common-community-physics-package-ccpp/tutorial-practical-instructions

Wrap up

- Authoritative repositories @github.com
- Use "git clone" to get the source code directories
- Understand the various CCPP build options
- Compile (with selected options) using the NEMSfv3gfs compile.sh script
- Run a test case (next section!)

This is the first phase – become familiar with how CCPP integrates with NEMSfv3gfs!

Next steps: modify or add code, contribute new code for review/inclusion (tomorrow!)

QUESTIONS?