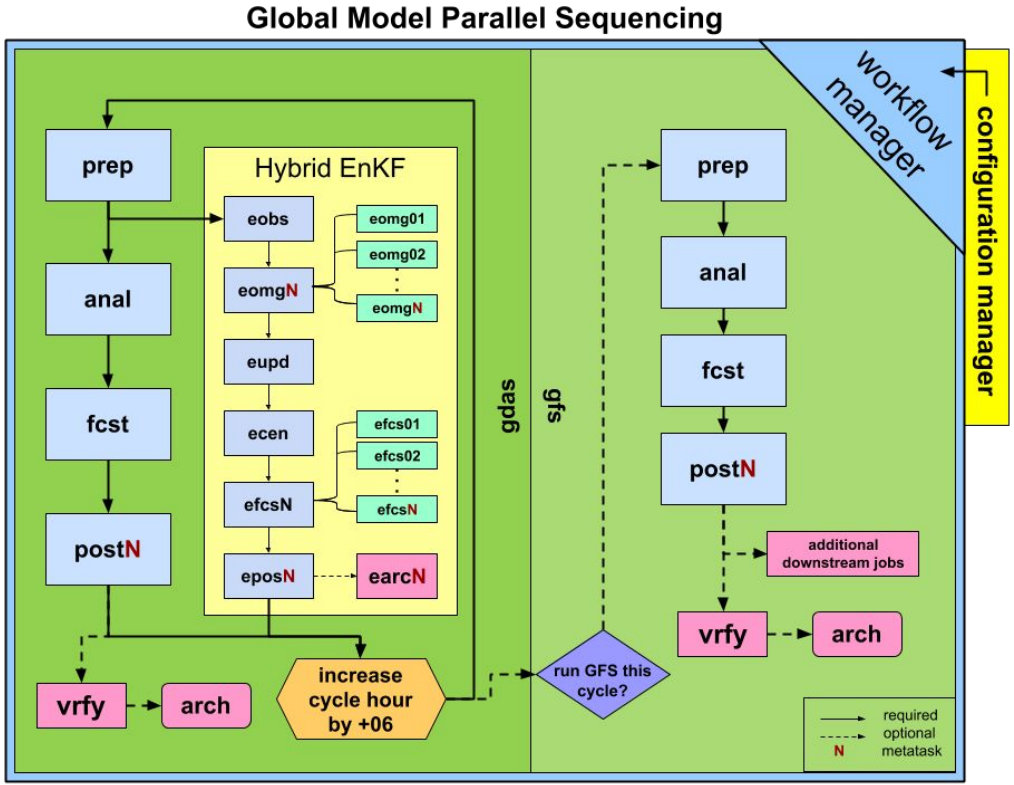
Workflow Requirements

An R2O perspective

What is a workflow from an NWP perspective?

- A number of interdependent jobs that are run at regular intervals in a predefined order based on the application
- Together these jobs typically make up a single cycle (here cycle refers to a single completion of a workflow process from beginning to end) and are run in order in an automated process, given certain conditions are satisfied
- A workflow system is designed so that you can run parts of a cycle, a single cycle, or multiple cycles
- In an operational framework a workflow that executes a cycle is typically run ad infinitum

Global Workflow (an example)



Parts of a workflow system

- Workflow manager/engine -- A system that is used to control all the steps in a particular cycle of the workflow and interface with the HPC scheduler, Examples are
  - ecFlow (from ECMWF)
  - Cylc (from NIWA/UKMO)
  - Rocoto (from GSL/NOAA)
- Application scripts/codes -- A collection of scripts called by the workflow engine to complete the underlying tasks associated with an end to end system. These scripts / codes are independent of the driving workflow manager
- Configuration System -- A set of tools that based on choice of application and workflow engine
  - Defines the tasks and order in the workflow, resource requirements, workflow engine setup (e.g. xml, ecf)
  - Sets up the necessary directories, gathers initial data, configures a baseline experiment

Desired Features in a workflow system

- Requirement for R2O workflow: Scientists should be able to update their components without spending considerable time or effort on the workflow
- Well documented
- Modular (reusable common functions to ensure optimal use of resources)
- Reliable with thorough error checking and logging
- If possible, can restart from failure points and give bit-identical results
- Should be easy to set up for different kinds of configurations
- Easy to add new features and capabilities (critical for multiple groups working together)
- Easily configurable and portable to multiple platforms and architectures
- Can run both DA and forecast cycles
- Can run in deterministic or ensemble mode
- Can work with coupled systems
- Can be transitioned to operations

Workflows, applications and model suites

- A model suite is defined as a collection of codes that are used to propagate an initial earth state forward in time
- UFS model suites are umbrella repositories that connect one or more component repository to make a model suite
- An application is defined as the end to end system that includes
  - The model suite
  - The workflow
  - All the configuration files that make up the application
- Two applications can share the same model suite (and maybe the same workflow)

UFS weather model

UFS weather model umbrella repository has hierarchical code structure

- All the repositories are currently located in GitHub with public access to the broad community
- Each component has its own authoritative repository

ufs-weather-model	https://github.com/ufs-community/ufs-weather-model
|	
|------- FMS	https://github.com/NOAA-GFDL/FMS
|------- FV3ATM	https://github.com/NOAA-EMC/fv3atm
| |-------- atmos_cubed_sphere	https://github.com/NOAA-GFDL/GFDL_atmos_cubed_sphere
| |-------- ccpp	
| |------- framework	https://github.com/NCAR/ccpp-framework
| |------- physics	https://github.com/NCAR/ccpp-physics
|------- NEMS	https://github.com/NOAA-EMC/NEMS
| |-------- prod_util	https://github.com/NOAA-EMC/NCEPLIBS-pyprodutil
|------- stochastic_physics	https://github.com/noaa-psd/stochastic_physics

Where does the build system sit?

- An application should be able to build the system and run it
- A model suite needs a build system / regression test capability to ensure that individual components when updated work with the full model suite
- EMC philosophy has been
  - Keep the build system with the model suite
  - Use the same build system at the application level
  - This has allowed us to have a separation of concerns
- At model suite level we test
  - Code compilation
  - Bit and restart reproducibility
  - Limited configuration level testing
- At the application level we test
  - End to end solution
  - Multiple configurations needed by the application

Does each application need its own workflow?

- Pro: Each application can develop on its own schedule without worrying about other use cases
- Con: This will lead to an explosion of workflows, with large parts repeating the same thing
- At EMC we are currently coalescing around 3 workflows
  - Global -- for medium range and sub seasonal (need to see if we can make this work for seasonal)
  - Regional -- for short range CAM applications
  - Hurricane -- for hurricane applications
- NOTE : Common tools for these workflows are being unified
- NOTE : Still are a collection of applications that do not fall into these categories (e.g. marine, space weather etc.)

Questions to ponder

- Can we build common tools that can then be configured into a handful of unique workflows?
- Our primary focus has been on cycled workflows. Can we build something that can meet both the R2O criteria -- an end to end system that goes from DA to product generation with cycling, and a research criteria -- testing solutions with incremental systems
- Is it possible to build a workflow system that does everything for the user (passes Graduate Student Test) but at the same time make it easy for multiple developers to work together and make changes?
- Workflows at EMC have historically been shell based. Is it time to look at alternatives like Python?
- How complex should the full system be?