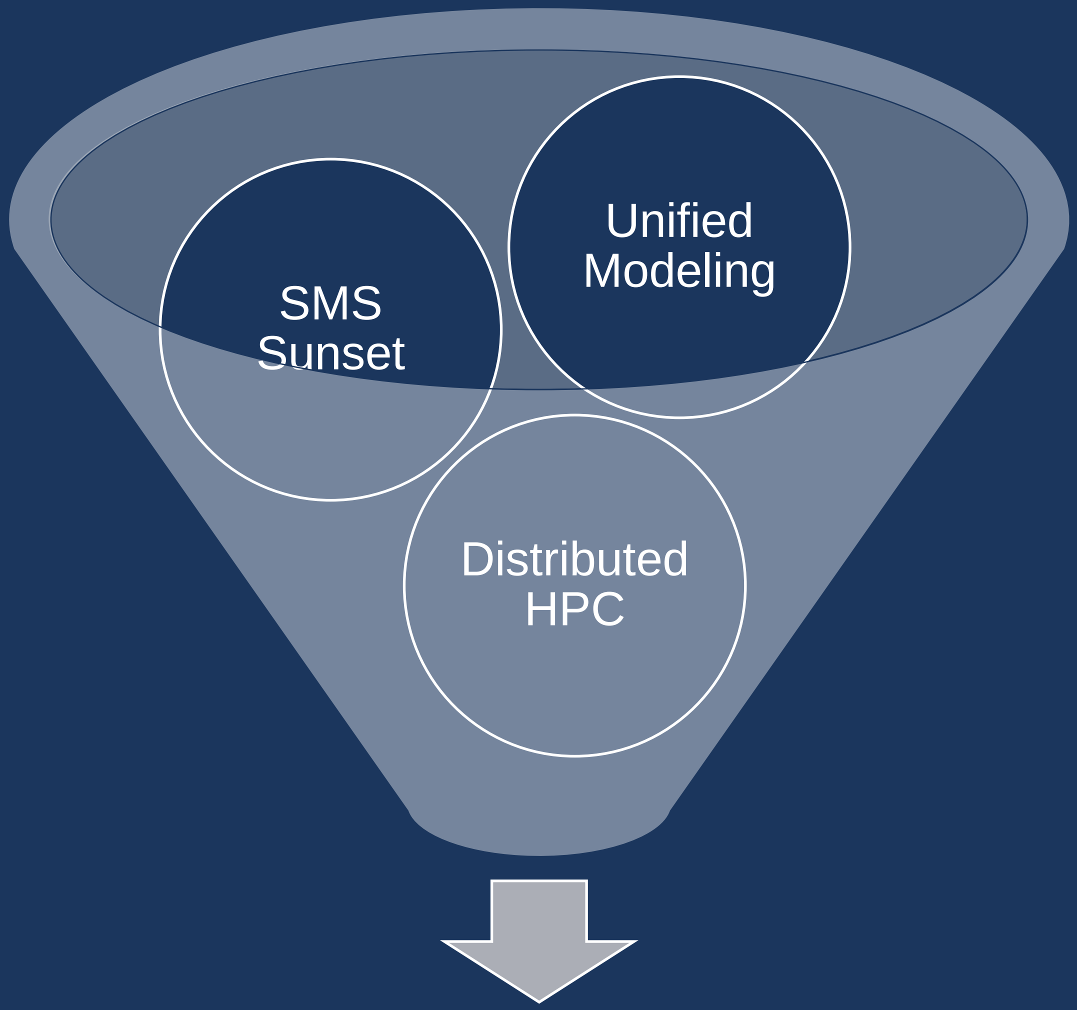
Workflow Modernization for the United States Navy METOC Modeling Enterprise

Timothy Whitcomb
Marine Meteorology Division
Naval Research Laboratory, Monterey, CA



OPSRUN Modernization

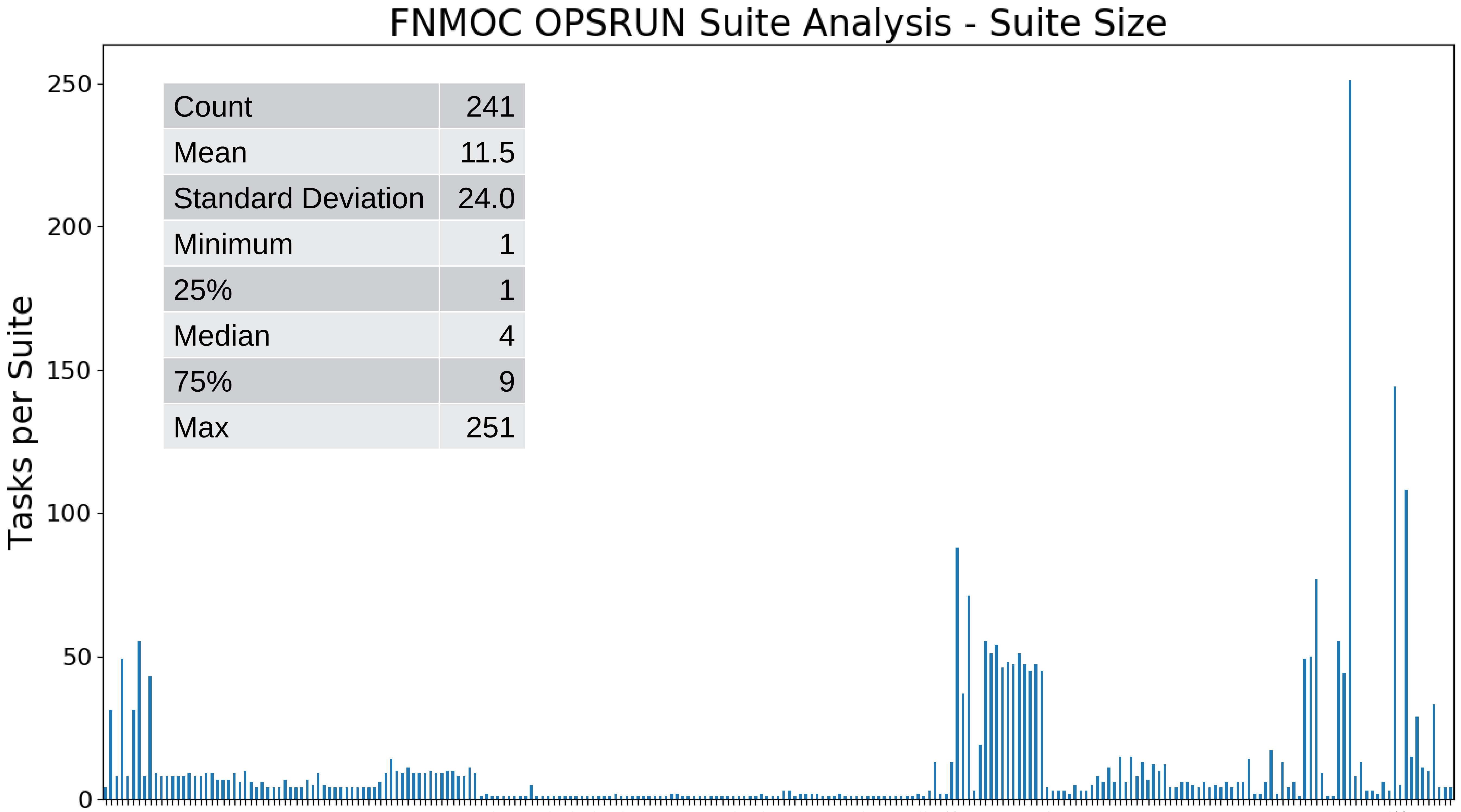
A unique opportunity to examine and re-envision FNMOC operations to reduce errors, provide easier monitoring, leverage new computing capabilities, and control ever-increasing complexity.

OPSRUN Review Process

- Capture overall suites
- Broadly classify suites by purpose
- Analyze suites for size and dependency information
- Use information gained from current operational practice and programmatic runlist analysis to recommend an updated approach and migration strategy.

What is done now?
How is it done now?
Why was it done that way?
Is there a different way to accomplish the same thing?
If there is a different way, is it better?


Data Dissemination
NAVGEM
COAMPS-TC
WW3
Administration
NCODA
Data Processing
Satellite Processing
WRIP
COAMPS
Functional Control
DAF



FNMOC OPSRUN Suite Analysis - Suite Size
Tasks per Suite
0
50
100
150
200
250
Count 241
Mean 11.5
Standard Deviation 24.0
Minimum 1
25% 1
Median 4
75% 9
Max 251

SMS to Cylc Migration

- Porting the existing system to Cylc is suboptimal – seek to *improve*, not just *reproduce*
- Leverage Cylc’s unique capabilities and learnings from the existing OPSRUN review
- Cylc developers have now put together a “Suite Design Guide” with best practices

What is done now?
How is it done now?
Why was it done that way?
Is there a different way to accomplish the same thing?
If there is a different way, is it better?


Cylc Rose Suite Design
Best Practice Guide
Version 1.0 - 23 March 2017
Last updated for: Cylc-7.2.0 and Rose-2017.02.0
Hilary Oliver, Dave Matthews, Andy Clark, and Contributors

SMS Suite Design	Cylc Suite Design
Implicit inter-cycle dependence	**Explicit** inter-cycle dependence
DTG property of a **suite**	DTG property of a **task**
Coarse-grained task definition	**Fine-grained** task definition
Suites constructed by **time**	Suites constructed by **system**


Cyl Rose Suite Design
Best Practice Guide
Version 1.0 - 23 March 2017
Last updated for: Cylc-7.2.0 and Rose-2017.02.0
Hilary Oliver, Dave Matthews, Andy Clark, and Contributors

SMS Suite Design	Cylc Suite Design
Implicit inter-cycle dependence	**Explicit** inter-cycle dependence
DTG property of a **suite**	DTG property of a **task**
Coarse-grained task definition	**Fine-grained** task definition
Suites constructed by **time**	Suites constructed by **system**

Adding true inter-cycle dependency information to the task graph can reduce errors, allows for the possibility of faster catch-up in case of delays, and permits the same suite to be used for retrospective and real-time scenarios.

SMS Suite Design	Cylc Suite Design
Implicit inter-cycle dependence	**Explicit** inter-cycle dependence
DTG property of a **suite**	DTG property of a **task**
Coarse-grained task definition	**Fine-grained** task definition
Suites constructed by **time**	Suites constructed by **system**

Using task-based DTG definitions allows tasks within suites to cycle on their natural interval (1 hour, 3 hours, 6 hours, etc.) rather than requiring additional handling to offset from a suite-level DTG.

Cylc Rose Suite Design
Best Practice Guide
Version 1.0 - 23 March 2017
Last updated for: Cylc-7.2.0 and Rose-2017.02.0
Hilary Oliver, Dave Matthews, Andy Clark, and Contributors

SMS Suite Design	Cylc Suite Design
Implicit inter-cycle dependence	**Explicit** inter-cycle dependence
DTG property of a **suite**	DTG property of a **task**
Coarse-grained task definition	**Fine-grained** task definition
Suites constructed by **time**	Suites constructed by **system**

Fine-grained task definitions produce suites that are more *complex* but much less *complicated*. Errors can be caught and handled earlier, task parallelism can be increased, resource utilization is often improved, and scripting for fine-grained tasks is significantly simpler.


Cylc Rose Suite Design
Best Practice Guide
Version 1.0 - 23 March 2017
Last updated for: Cylc-7.2.0 and Rose-2017.02.0
Hilary Oliver, Dave Matthews, Andy Clark, and Contributors

SMS Suite Design	Cylc Suite Design
Implicit inter-cycle dependence	**Explicit** inter-cycle dependence
DTG property of a **suite**	DTG property of a **task**
Coarse-grained task definition	**Fine-grained** task definition
Suites constructed by **time**	Suites constructed by **system**

System-level suites provide reduced complexity (as many suites are typically very similar) and allow for easier testability (e.g. beta or preops) and isolation for scalability of suite controllers.

DoD Security Considerations

- Cybersecurity is an immediate and growing concern within U.S. DoD
- Software clearance governed by STIGs (Security Technical Implementation Guide)
  - https://www.stigviewer.com/stig/application_security_and_development/
- PKI authentication (hard and soft certificates) in almost all circumstances

U.S. NAVAL RESEARCH LABORATORY

Suite Design Aspirations

- Use built-in directories (e.g. $CYLC_SUITE_SHARE_DIR, $CYLC_TASK_WORK_DIR) for *everything*
    - Allows for built-in suite isolation for deployment without risk of conflicts.
    - Allows for easy porting to separate platforms
- Use data movement tasks (particularly at startup) to move or link data into the suite workspace
    - Allows platform-specific operations to get data into shared workspace
    - Documents *precisely* the static and dynamic data dependencies of the application/process the suite is for
- Consistent nomenclature (e.g. *stage* for moving data into the suite workspace)
- Consistent task color/shapes for visualization

Prescribed Directory Setup

- All suites should use $CYLC_SUITE_SHARE_DIR with the following top-level layout:

```
$CYLC_SUITE_SHARE_DIR
        /static
        /dynamic
                /$CYLC_TASK_CYCLE_POINT
```

- Use the cycle point as the top-level division in the dynamic share directory
- If tasks require data from multiple cycle points, can either access directly or via intermediate connector tasks
- Top-level cycle points allow for very easy housekeeping and cleanup

Designing Suites for Multiple Platforms

Multi-Platform Suite Design

- NRL and FNMOC run on many different computational platforms
- Examples of this year alone:
  - On-premises Linux clusters
  - Cray XE6
  - Cray XC-30
  - Cray XC-40
  - Cray XC-50
  - SGI ICE X
  - HPE SGI 8600

Multi-Platform Suite Design

- Cylc configuration parsing allows for repeated configuration sections
  - Repeated configuration items: over-ride
  - Repeated graphs: *append*
- Place all platform-specific configuration in a separate file
  - Batch system information
  - Parallel/serial run commands (e.g. aprun –n 1 on Cray)
  - Timing data
  - Additional/alternate tasks (e.g. CCM on Cray)
- At least one instance of platform-specific configuration for each separate platform (maybe more, e.g. ops vs. r&d)

Inline Processing with Zero Code Changes

Inline Output Processing

- Use functions/scripts that parse log files (e.g. from a forecast model) and emit Cylc messages back to the suite controller
- Allows for inline processing with no source code modification
- System designers must allow their programs to be run inline (e.g. output or data conversion) with a defined stop time (and start time!) to allow parameterized windowing
- Definition of output windows is accomplished using Jinja2 parameterization to allow the same suite definition to be used with multiple output window specifications
- Jinja2 filters for duration allow backwards compatibility with programs that do not support ISO8601 notation

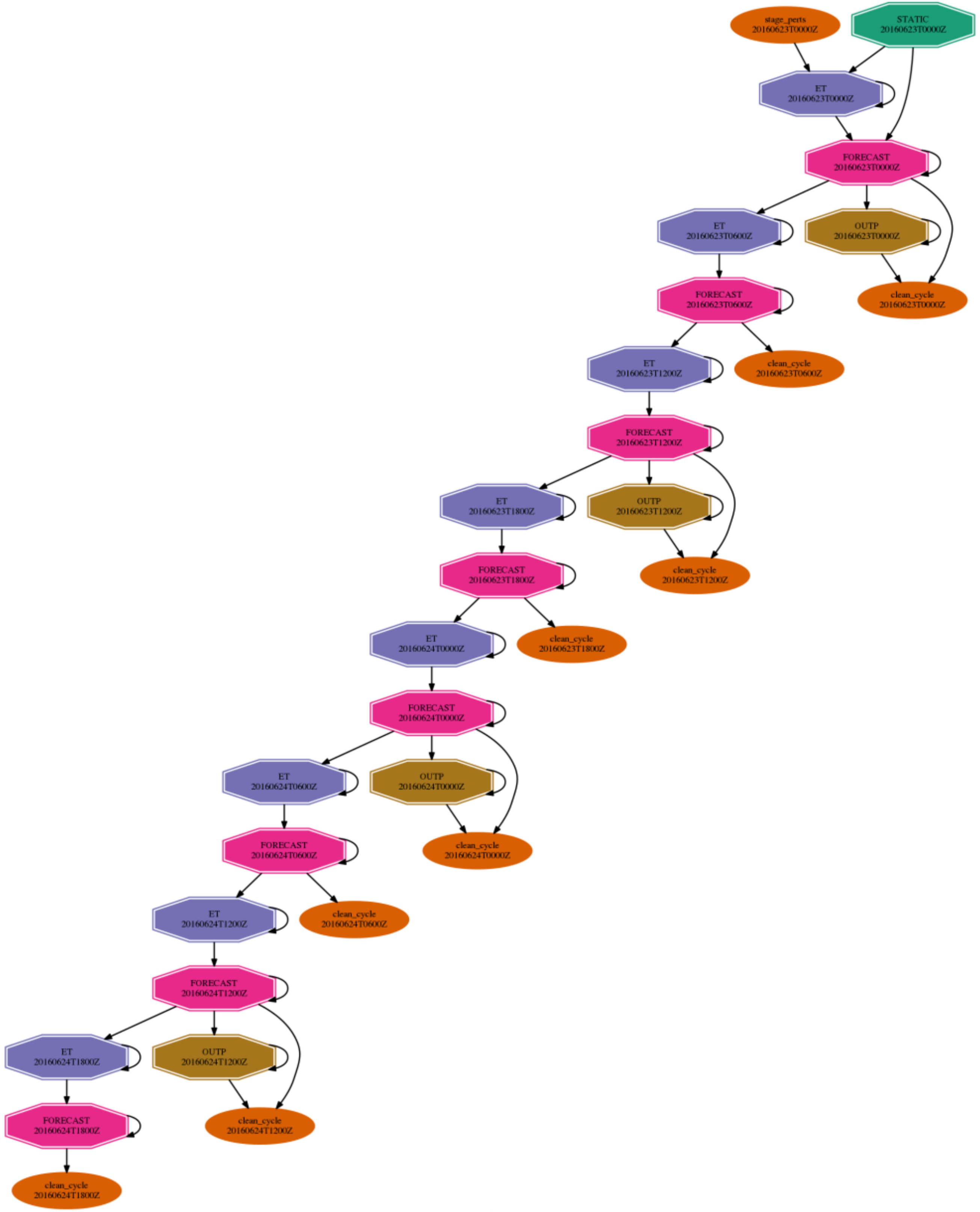
Global Ensemble System (NAVGEM EFS)

NAVGEM EFS Suite Design

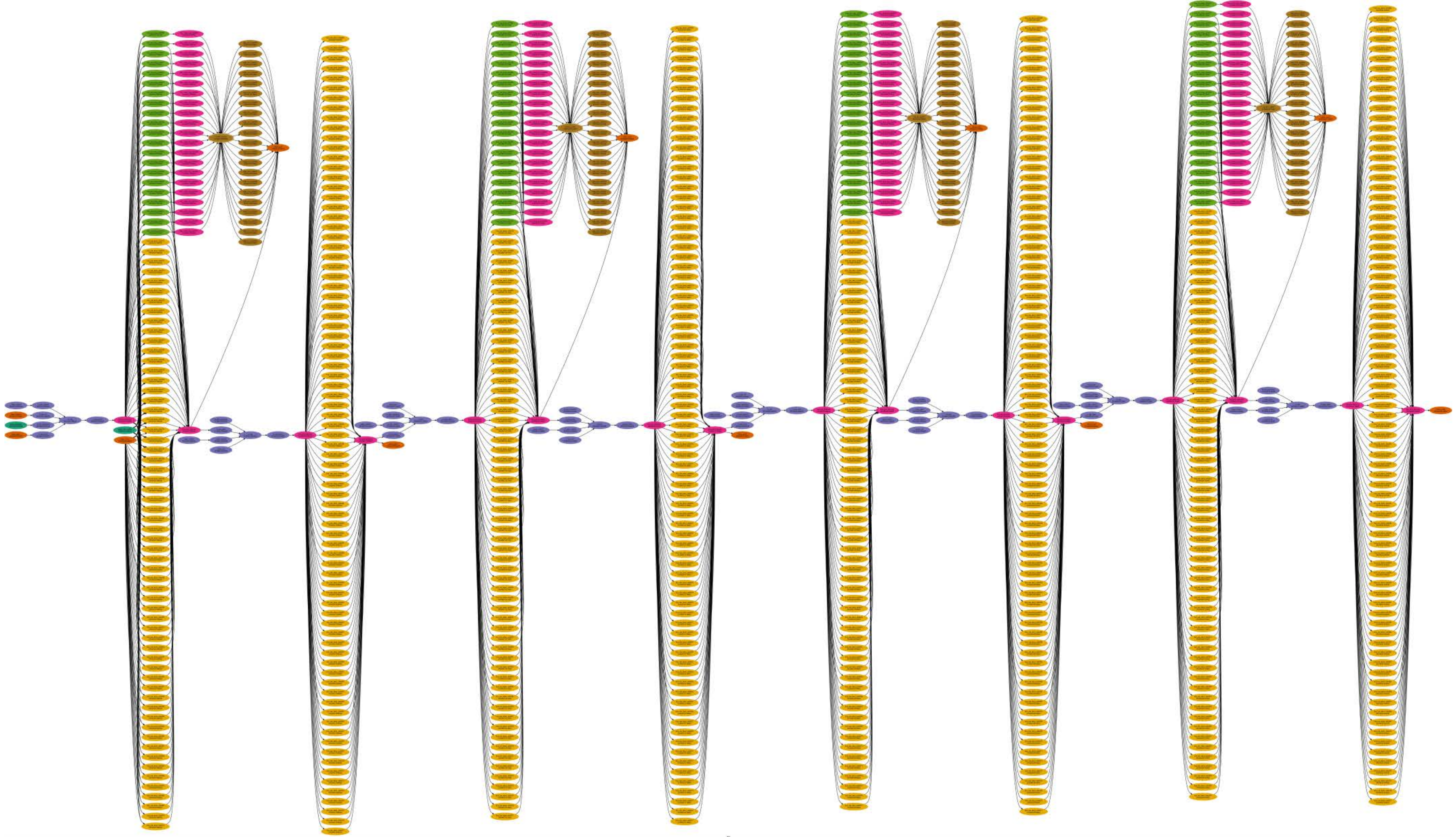
- Ensemble Transform technique for generating ensemble initial conditions
- Runs at a lower resolution than the deterministic forecast model, which requires an interpolated control member
- Use a larger number of members for cycling the ET (currently 80) but cycle fewer members for the ensemble distributed to the wider community (currently 20 members)
- Research and operations use different methods of storing the ensemble members for the ET.
- Research and operations typically use different ensemble sizes

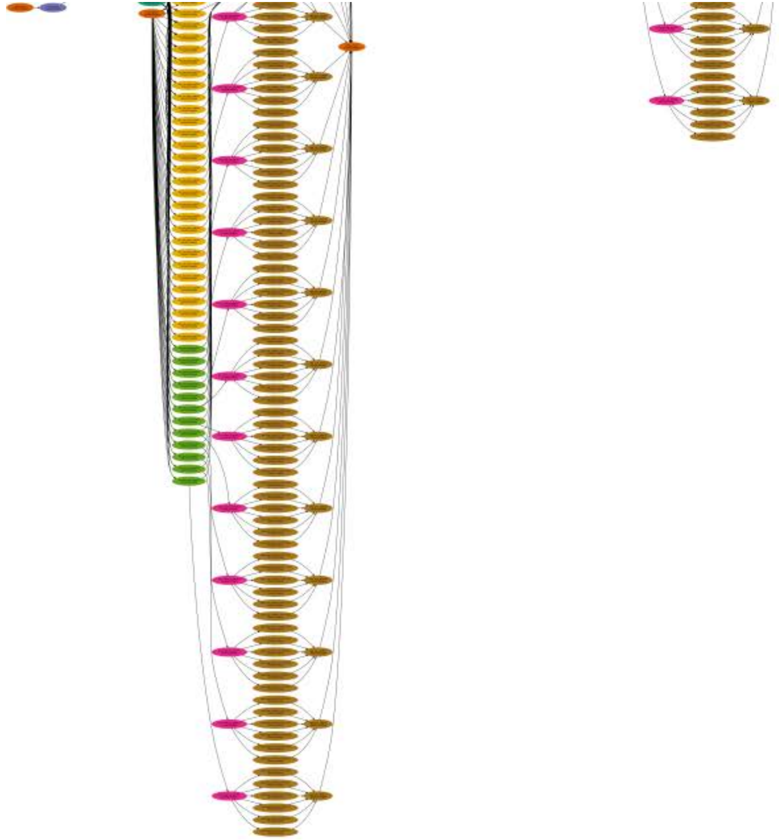
Highlights of the EFS Suite

- Fully parameterized size and forecast duration of the full ensemble
- Parameterized processing to cycle through the full ensemble with long forecast subsets
- Fully parameterized output windows for streamlining output and postprocessing
- Data movement of initial ensemble required on first cycle
  - Investigating automated initialization using very long forecasts with stochastic physics to provide initial perturbations
- Data movement of a control member required on every cycle
  - Recommend moving interpolation into ensemble suite to reduce external dependencies.


stage_perts 20160623T0000Z
STATIC 20160623T0000Z
ET 20160623T0000Z
FORECAST 20160623T0000Z
ET 20160623T0600Z
OUTP 20160623T0000Z
FORECAST 20160623T0600Z
clean_cycle 20160623T0000Z
ET 20160623T1200Z
clean_cycle 20160623T0600Z
FORECAST 20160623T1200Z
ET 20160623T1800Z
OUTP 20160623T1200Z
FORECAST 20160623T1800Z
clean_cycle 20160623T1200Z
ET 20160624T0000Z
clean_cycle 20160623T1800Z
FORECAST 20160624T0000Z
ET 20160624T0600Z
OUTP 20160624T0000Z
FORECAST 20160624T0600Z
clean_cycle 20160624T0000Z
ET 20160624T1200Z
clean_cycle 20160624T0600Z
FORECAST 20160624T1200Z
ET 20160624T1800Z
OUTP 20160624T1200Z
FORECAST 20160624T1800Z
clean_cycle 20160624T1200Z
clean_cycle 20160624T1800Z

suite





Running the forecast model is typically the easy part: it’s the upstream and downstream processing that adds complexity.

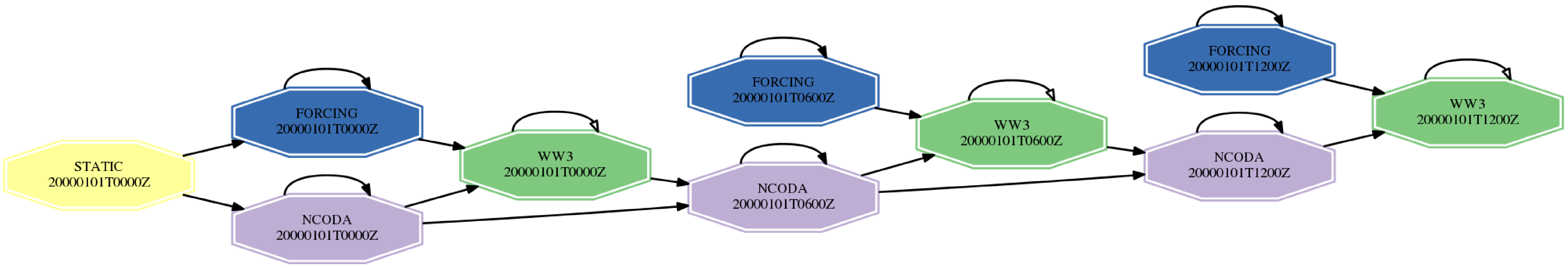
Global Wave Forecasts (WaveWatch III)

Global WaveWatch III Suite Design

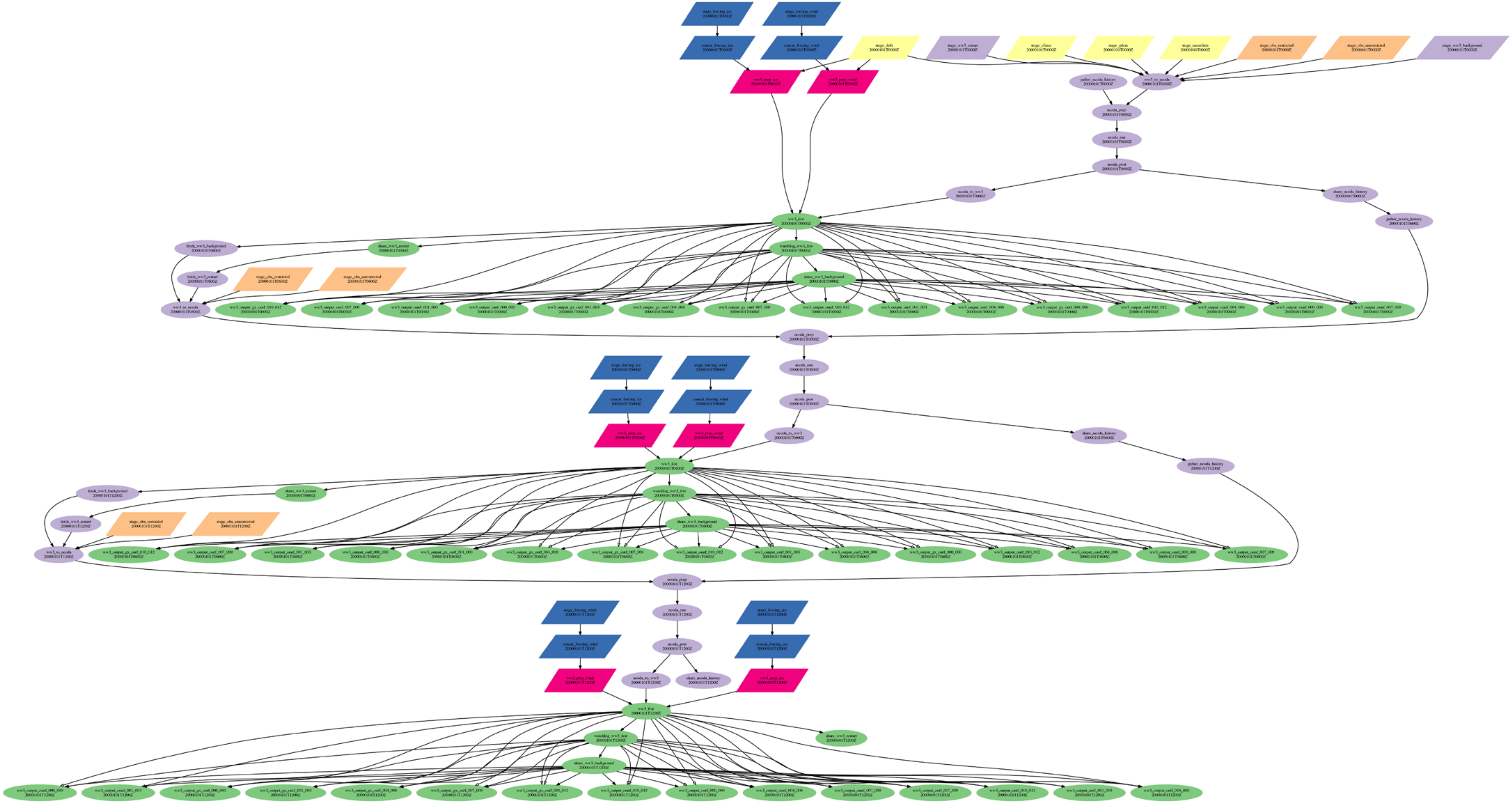
- Requires input of observations for wave data assimilation
- Requires input of ocean and atmosphere fields
- Requires separate output program
- Run in several time chunks to allow output to be produced on time
- Needs to support multigrid configurations (in progress)
- WaveWatch forecast model and output program assume that *everything* is in the current working directory

Highlights of the WaveWatch III Suite

- Fully parameterized sources of forcing data (e.g. winds, sea ice)
- Fully parameterized output windows for streamlining output and postprocessing
- Data movement of forcing data required on every cycle
- Use cylc's `work sub-directory` to allow several tasks to share a working directory (where required)
- Use filesystem links to allow side-by-side execution of output and the forecast model


STATIC
20000101T0000Z
FORCING
20000101T0000Z
NCODA
20000101T0000Z
WW3
20000101T0000Z
FORCING
20000101T0600Z
NCODA
20000101T0600Z
WW3
20000101T0600Z
FORCING
20000101T1200Z
NCODA
20000101T1200Z
WW3
20000101T1200Z

ncoda



ncoda

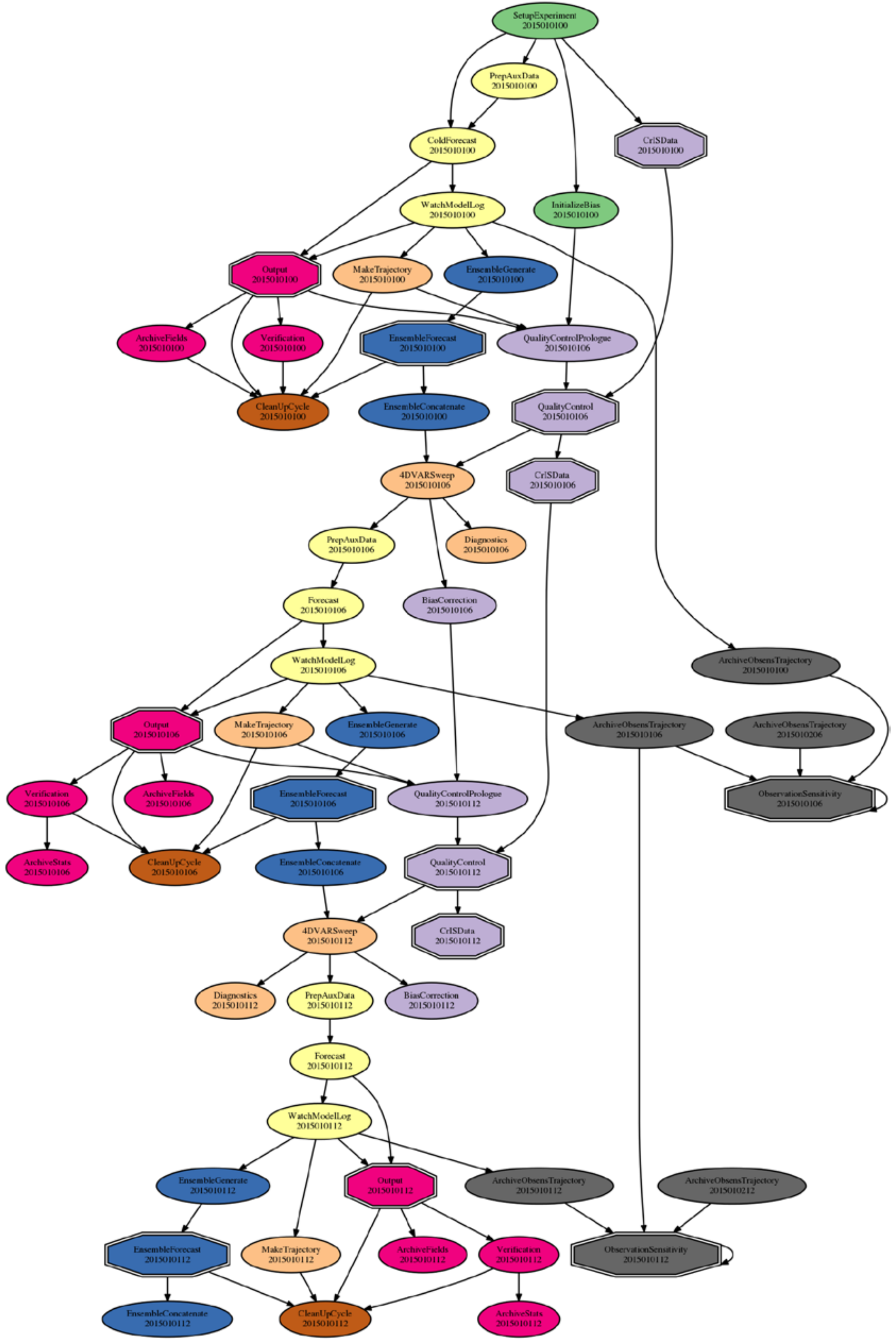
Global Deterministic Model (NAVGEM)

NAVGEM Suite Design

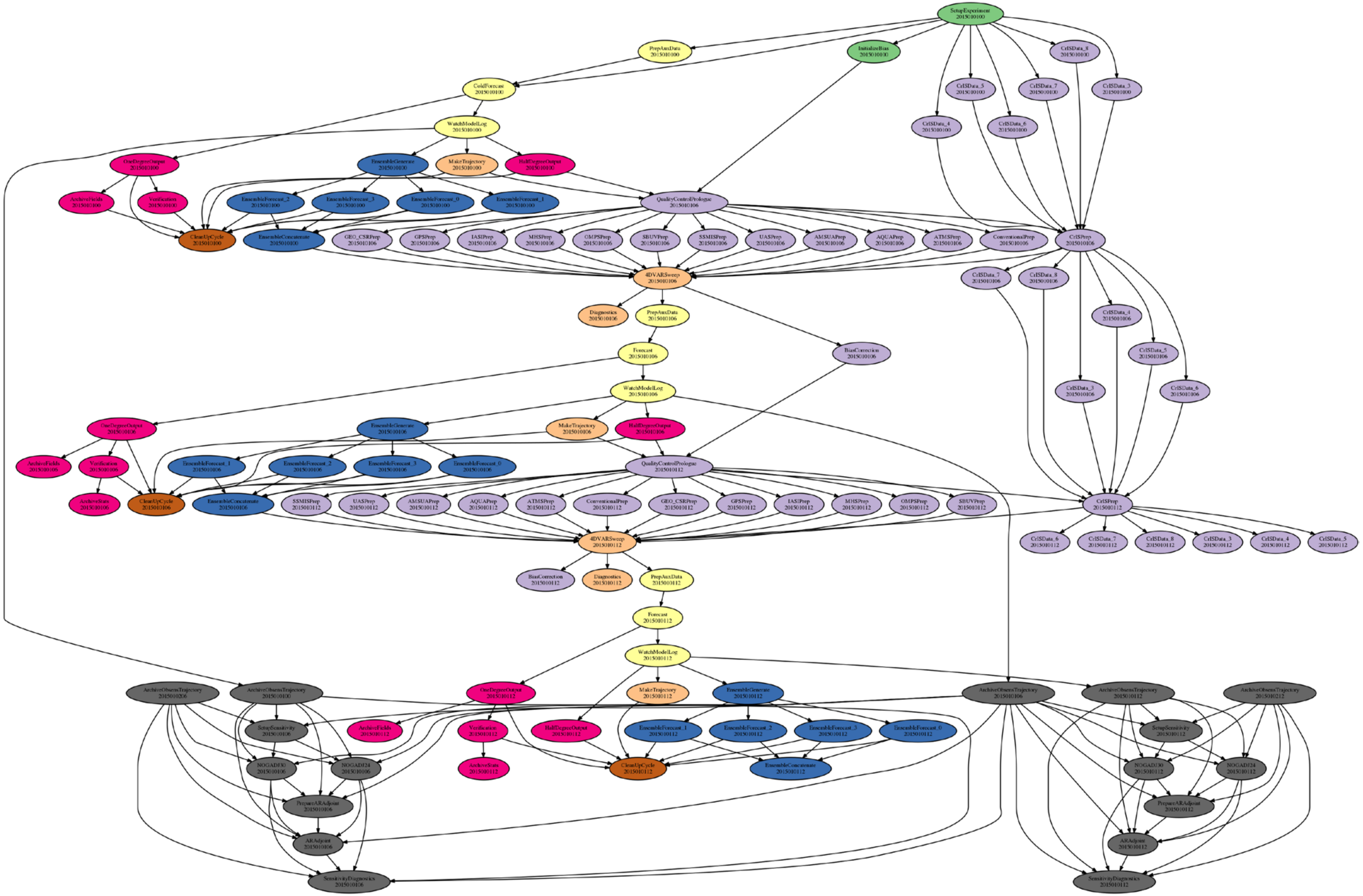
- Requires input of observations for atmospheric data assimilation
- Cycles with a 6-hour window for the 4D-Var data assimilation
- Includes built-in 80-member ensemble for Hybrid 4D-Var
- Requires input of surface forcing data from other systems (e.g. land surface and SST)
- Consists of observation quality control, data assimilation, forecast, verification, and forecast sensitivity to observations (FSOI) calculations
- Operations and R&D support different forecast lead times
    - FSOI calculations also require minimum forecast lead time of PT30H

Highlights of the NAVGEM Suite

- More sophisticated cold-start bootstrapping to accommodate the 4D-VAR data assimilation system
- Parameterized ensemble size for the Hybrid 4D-Var
- Fully parameterized output windows for streamlining output and postprocessing
- Data movement of observation data required on every cycle
- Leverage *future* inter-cycle dependence to support unique configuration of FSOI (e.g. needs a forecast from last cycle, this cycle, and an analysis 24 hours from now)


SetupExperiment 2015010100
PrepAuxData 2015010100
ColdForecast 2015010100
CrISData 2015010100
WatchModelLog 2015010100
InitializeBias 2015010100
Output 2015010100
MakeTrajectory 2015010100
EnsembleGenerate 2015010100
ArchiveFields 2015010100
Verification 2015010100
EnsembleForecast 2015010100
QualityControlPrologue 2015010106
CleanUpCycle 2015010100
EnsembleConcatenate 2015010100
QualityControl 2015010106
4DVARSweep 2015010106
CrISData 2015010106
PrepAuxData 2015010106
Diagnostics 2015010106
Forecast 2015010106
BiasCorrection 2015010106
WatchModelLog 2015010106
ArchiveObsensTrajectory 2015010100
Output 2015010106
MakeTrajectory 2015010106
EnsembleGenerate 2015010106
ArchiveObsensTrajectory 2015010106
ArchiveObsensTrajectory 2015010206
Verification 2015010106
ArchiveFields 2015010106
EnsembleForecast 2015010106
QualityControlPrologue 2015010112
ObservationSensitivity 2015010106
ArchiveStats 2015010106
CleanUpCycle 2015010106
EnsembleConcatenate 2015010106
QualityControl 2015010112
4DVARSweep 2015010112
CrISData 2015010112
Diagnostics 2015010112
PrepAuxData 2015010112
BiasCorrection 2015010112
Forecast 2015010112
WatchModelLog 2015010112
EnsembleGenerate 2015010112
Output 2015010112
ArchiveObsensTrajectory 2015010112
ArchiveObsensTrajectory 2015010212
EnsembleForecast 2015010112
MakeTrajectory 2015010112
ArchiveFields 2015010112
Verification 2015010112
ObservationSensitivity 2015010112
EnsembleConcatenate 2015010112
CleanUpCycle 2015010112
ArchiveStats 2015010112

navgem



navgem

Closing Thoughts

- Cylc adoption is a powerful tool for reducing technical barriers in transitioning scientific research to operations
- Adoption isn’t enough – requires some care for how to write and configure suites for maximum cross-platform and cross-agency compatibility
- Interested in Altair/BoM work using Apache Kafka to facilitate flexible inter-suite triggering
    - Especially in the face of distributed HPC!
- Also reduce barriers for visiting scientists, postdocs, or even laboratory staff unfamiliar with a particular system