CSEG Update

Mariana Vertenstein

CESM Software Engineering Group

Outline

1. Inter-component: addition of new component – MOM6
2. Inter- component: new interoperability for CTSM – LILAC
3. Inter-component: regional CESM capability - CRESM
4. Intra-component: configuring MARBL in the ocean host model
5. Workflow in CMIP6

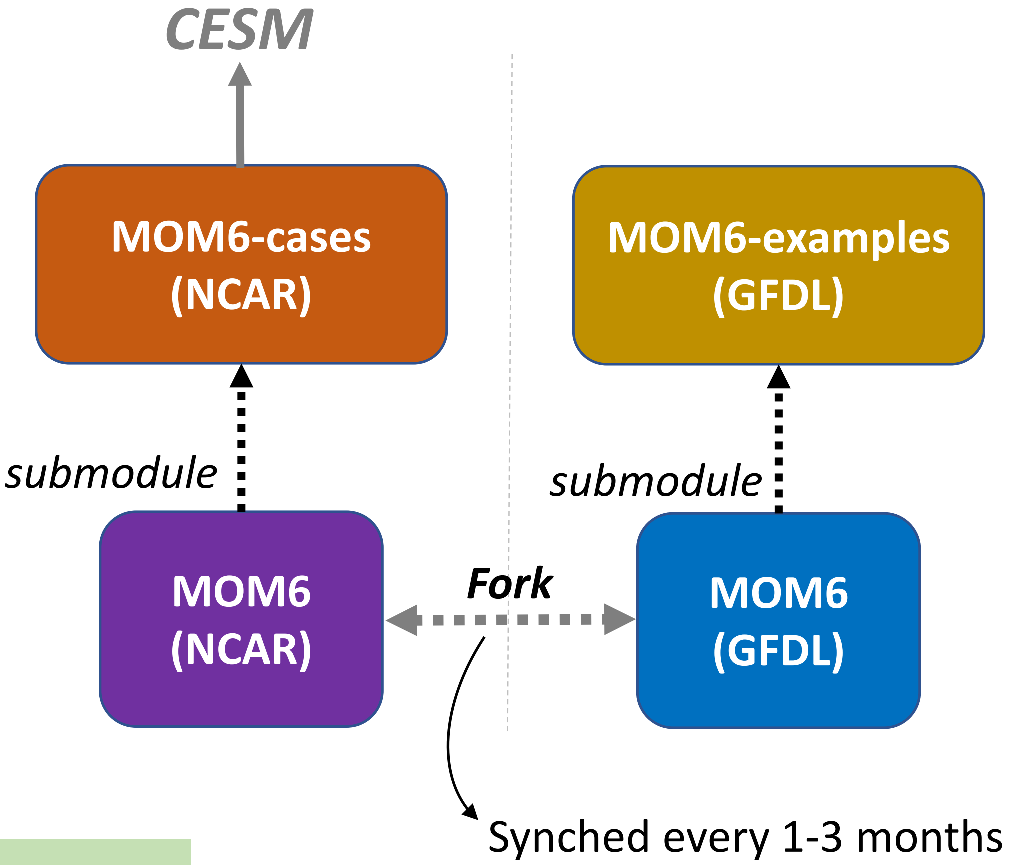
Addition of new component to CESM - MOM6

Alper Altuntas
Gustavo Marques
Frederic Castruccio
Scott Bachman

Repository Organization

Two new *layered* GitHub repositories based on GFDL's repository organization to facilitate CESM coupling and to streamline collaboration with GFDL:

- github.com/**NCAR/MOM6**:
  - Core MOM6 source code + MCT/NUOPC caps
- github.com/**NCAR/MOM6-cases:**
  - CIME compatibility is here (**not** MCT/NUOPC caps)
  - FMS inclusion is here
  - Encapsulates NCAR/MOM6 as a submodule
  - Analogous to (yet independent of) GFDL's MOM6-examples

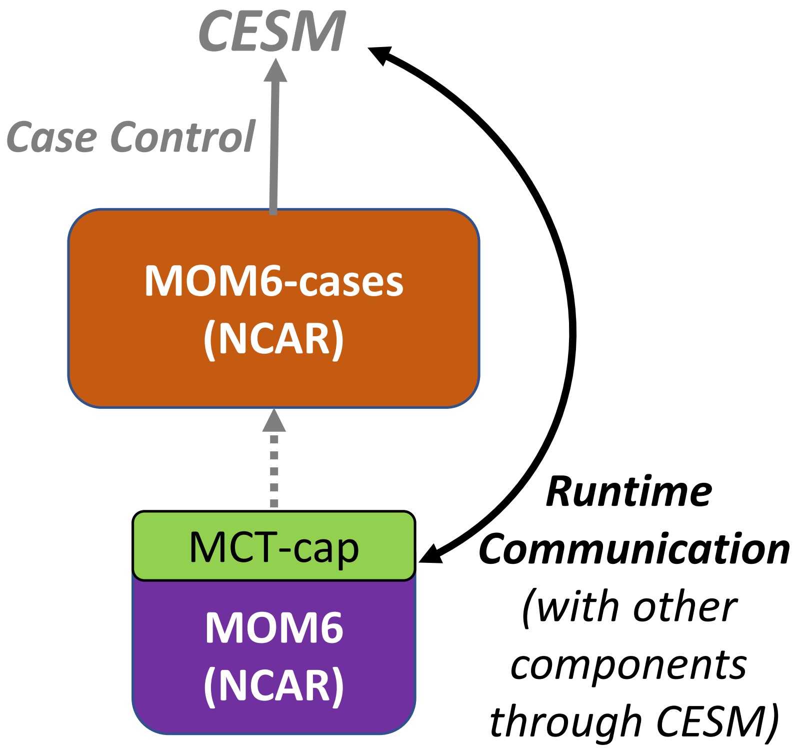


Online Wiki: **https://github.com/NCAR/MOM6-cases/wiki**
Installation, usage, quick-start, etc.

MOM6 in CESM

- **Compliance with CIME case control system:**
  - Interfaces MOM6 with CIME scripts.
  - Contains FMS code base
  - Controls build, configuration, model inputs, outputs, archiving, etc.

- **Compliance with CIME MCT driver:**
  - Added to MOM6 repository.
  - Makes use of CIME ***and*** FMS (GFDL) libraries.
  - Implemented MCT interfaces
    - ocn_init_mct, ocn_run_mct ocn_final_mct
  - NUOPC cap will go here as well
  - Fully Doxygenized.



MOM6 in CESM

- **Current Status**
  - Repository organization and synchronization with GFDL established.
  - Addition of MCT cap
  - Implementation of C and G compsets
  - Established operational testing workflow (both internally and collectively with GFDL).
    - continuous integration using Travis for each PR submitted to MOM6-cases
    - MOM6 regression tests (55 in total) run when NCAR and GFDL forks are merged

- **Next Steps:**
  - Complete implementation of parameterizations used in POP (e.g., CVMix , EBM).
  - Implement CIME system testing infrastructure
  - Add lightweight diagnostic tools (based on GFDL's diagnostics approaches)
  - Develop additional compsets and run additional experiments
  - Release a CESM2+MOM6 branch tag by Summer 2018.

Current MOM6-CESM Validation Status

Control experiments (C and G compsets):

- G compsets successfully run for 100 years. (Normal year forcing)
- Ocean coupling interval set to 30 mins.

Grids: (all have 62 vertical levels)

- **gx1v6**
  - Based on 1° POP grid.
- **tx0.66v1**
  - A new grid for MOM6+CESM
  - Tripolar, nominal 2/3° horizontal resolution.

- **Performance (G compsets, gx1v6, 360 cheyenne cores)**
  - POP:~43 years/day MOM6: ~22 years/day

New Interoperability for CLM (CTSM)

LILAC

Lightweight Infrastructure for Land Atmosphere Coupling

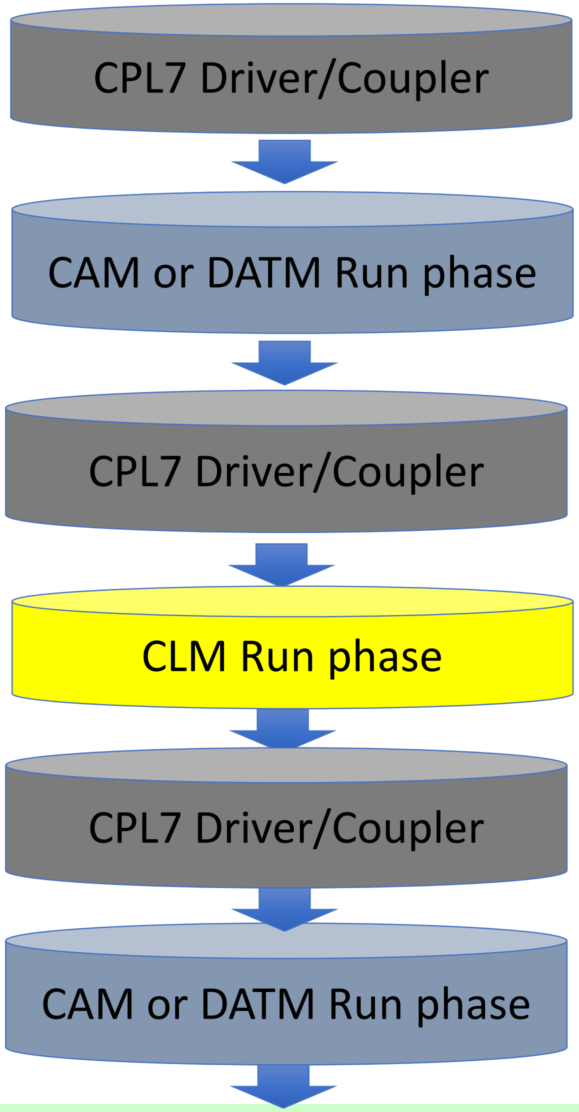
A tool for easy integration of the Community Land Model into multiple modeling systems

Ben Andre and Bill Sacks

Key LILAC Goals:

- Develop lightweight, coupling infrastructure to facilitate an atmosphere model directly calling CTSM from its init and run loo rather than requiring host atmosphere to be a component of CESM to couple to CTSM
- Develop capabilities for CTSM to fill in required missing fields (e.g., aerosol deposition) not provided by host atmosphere model
- Simplify CTSM's tool-chain for creating input datasets
- Develop faster methods for creating spun-up CTSM initial conditions required for target host model
- Target atmosphere components to couple CTSM via LILAC - WRF, MPAS-A, SAM, COSMO

CESM run sequencing



CPL7 Driver/Coupler

CAM or DATM Run phase

CPL7 Driver/Coupler

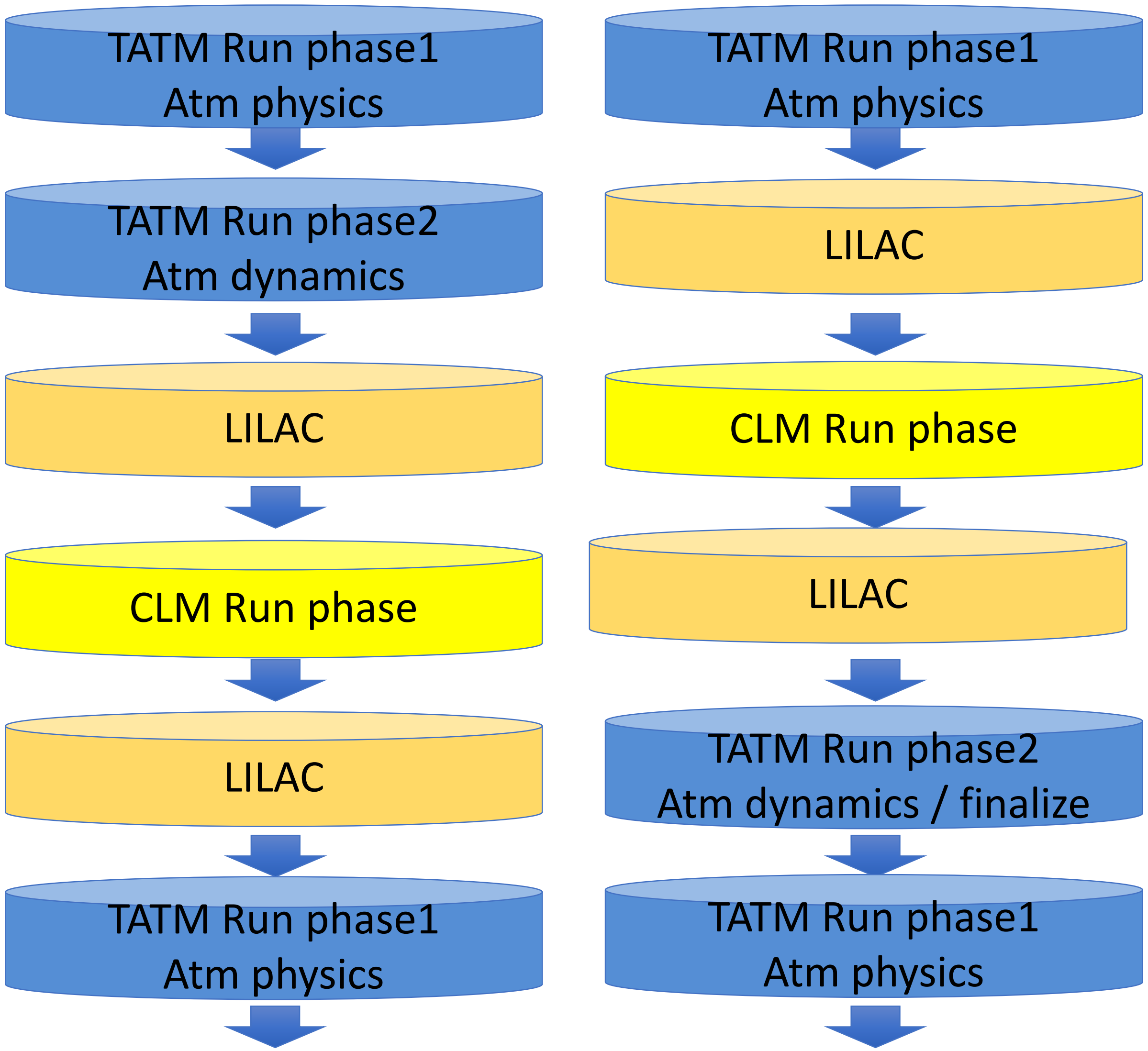
CLM Run phase

CPL7 Driver/Coupler

CAM or DATM Run phase

CPL7 requires one run phase for any coupled component. CAM must *finish* its run phase before communicating with driver.

LILAC enables flexible run sequences



Example 1

TATM Run phase1
Atm physics

TATM Run phase2
Atm dynamics

LILAC

CLM Run phase

LILAC

TATM Run phase1
Atm physics

Example 2

TATM Run phase1
Atm physics

LILAC

CLM Run phase

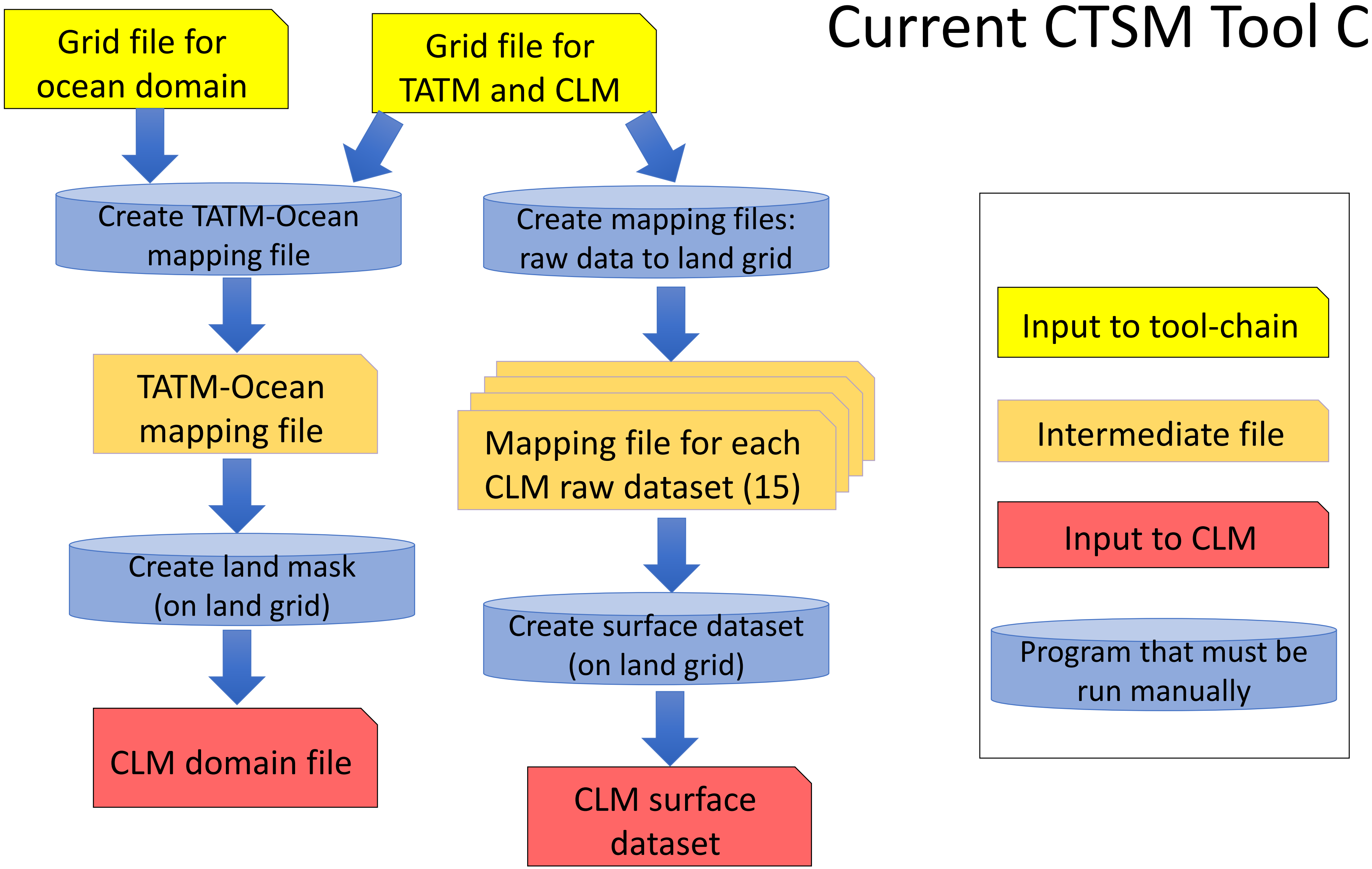
LILAC

TATM Run phase2
Atm dynamics / finalize

TATM Run phase1
Atm physics

With LILAC, atmospheric component is not restricted to one run phase, and CLM can couple to atmosphere anywhere in its run phase.

Current CTSM Tool Chain


Grid file for ocean domain
Grid file for TATM and CLM
Create TATM-Ocean mapping file
Create mapping files: raw data to land grid
TATM-Ocean mapping file
Mapping file for each CLM raw dataset (15)
Create land mask (on land grid)
Create surface dataset (on land grid)
CLM domain file
CLM surface dataset
Input to tool-chain
Intermediate file
Input to CLM
Program that must be run manually

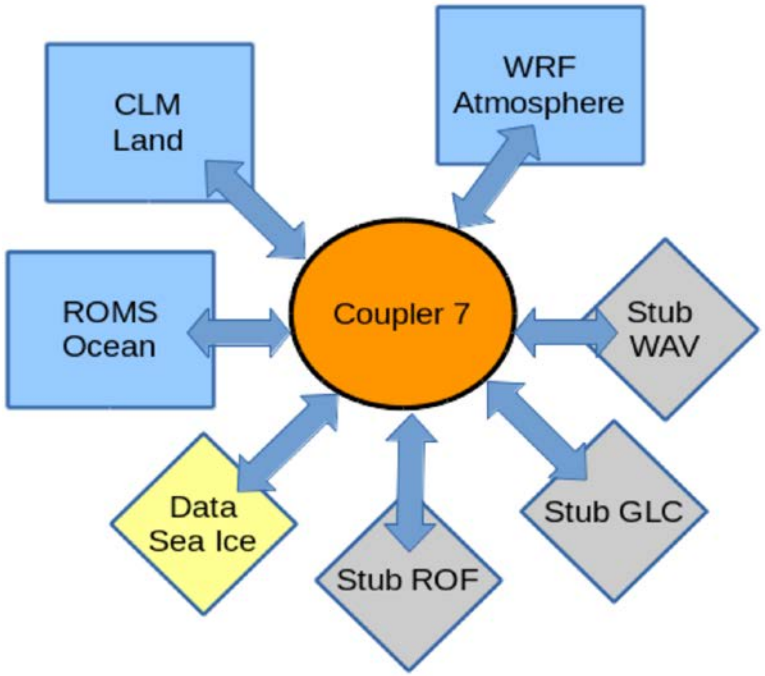
LILAC goal - develop a next-generation implementation of tool chain making it easy for non-experts to perform this task.

Regional CESM

Kate Thayer-Calder (CSEG)
Jaison Kurian (TAMU)
Rafaelle Montouro (CIRES)
Ping Chang (TAMU)

Community Regional Earth System Model (CRESM)

- Joint TAMU (Texas A and M University) and NCAR effort
- Build a regional version of CESM using WRF (atm) and ROMS (ocn)
- All components are coupled with and integrated into the CIME case control system and coupling architecture
- Provide a more user-friendly experience than typical for regional models.
- Allow for higher resolution modeling modeling in both time and two way atmosphere-land-ocean feedbacks

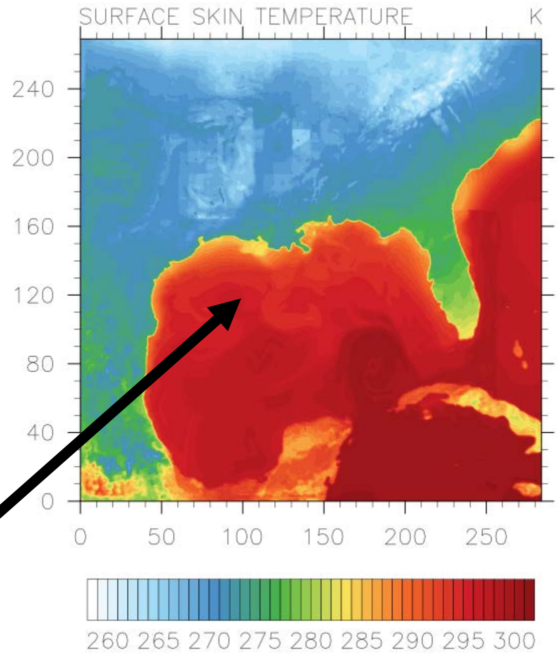


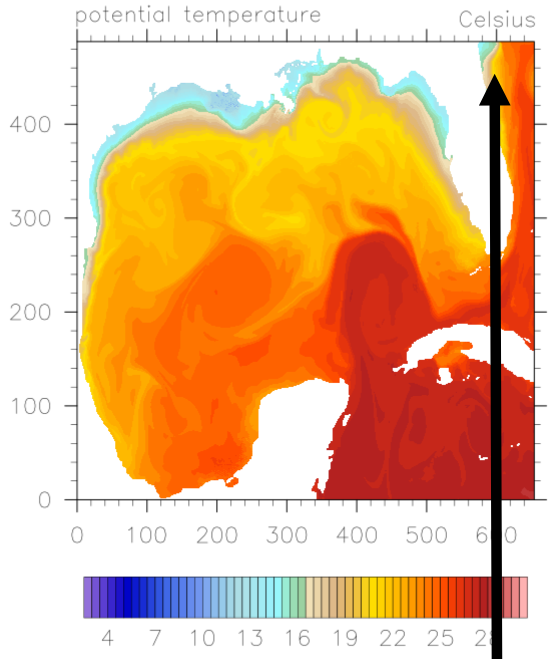
CRESM Status

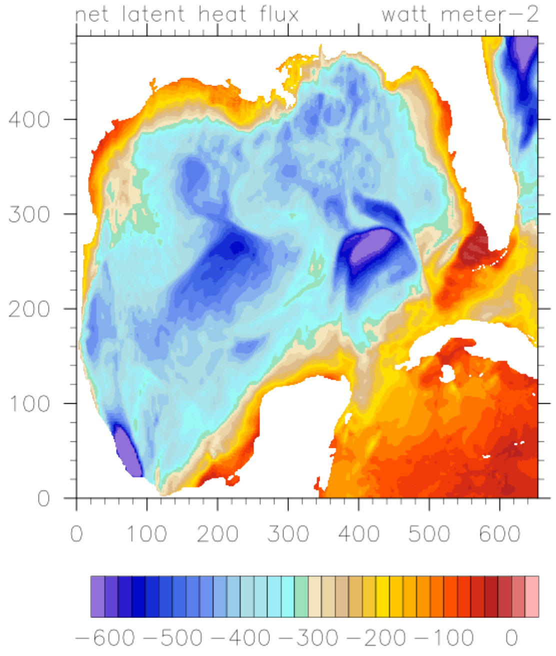
- Current development version supports several configurations including:
    - ➞ WRF atmosphere, CLM land, Data Ocean (PK compsets)
    - ➞ fully CROMS Ocean, Data Atmosphere (PR compsets)
    - ➞ coupled WRF, CLM, and ROMS (PB compsets)
- Current – all compsets are now working – but need a clean-up and testing phase.
- Future – addition of user-friendly support for more grids and boundary forcings, scientific applications (e.g. tropical waves), a more recent WRF version

Community Regional Earth System Model (CRESM)

- Fully Coupled Gulf of Mexico January 2010 test case







Atm/Ocn fluxes
Look reasonable

Day 5 10:00:00

WRF surface skin temp
WRF is seeing the ROMS surface temp

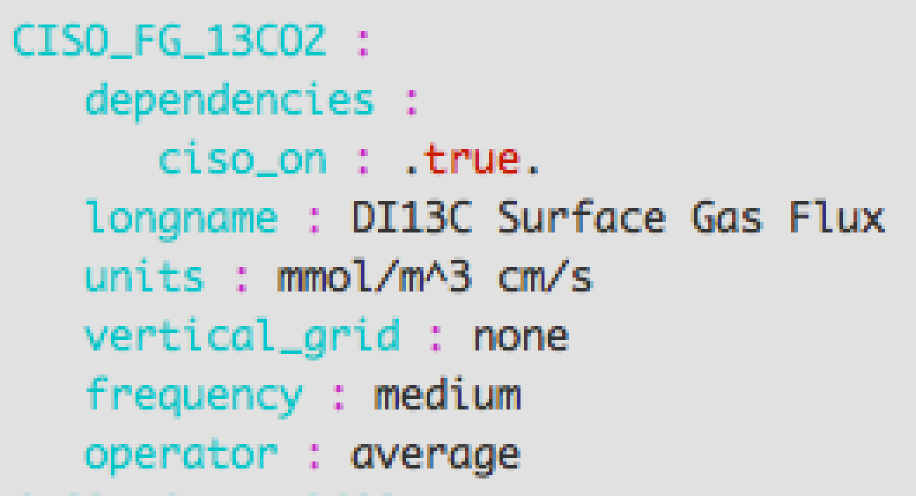
ROMS surface temperature
Smaller ROMS domain (e.g. see Florida) with missing edge data
filled from the data ocean component does not seem to effect results

Interoperability of parameterizations: New python capability in MARBL

Mike Levy
Keith Lindsay
Matthew Long

Python Scripts in MARBL (1)

- New MARBL capability to generate MARBL input parameters (marbl_in)
- MARBL input parameters and diagnostic variables are defined in YAML (not XML)
- Why YAML: human-readable (and comments are allowed!)
- Drawback of YAML: PyYAML not in standard lib
- Solution: developers edit YAML and then run a script to convert to JSON. Python scripts read the JSON (which is in standard lib).
- Longer term goal: auto-generate Fortran code
  - Use JSON dictionaries to generate F90 code that define parameters (lots of repeated code)
  - (Currently, information is duplicated between F90 and YAML files)



```
CISO_FG_13CO2 :
   dependencies :
      ciso_on : .true.
   longname : DI13C Surface Gas Flux
   units : mmol/m^3 cm/s
   vertical_grid : none
   frequency : medium
   operator : average
```

Example from the diagnostics YAML *dictionary; this particular diagnostic is only enabled when running with the carbon isotope tracers.*

Python scripts in MARBL (2)

- New MARBL diagnostic capability
  - MARBL now provides a list of diagnostics with associated averaging frequencies (via python + YAML)
  - Ocean host model (e.g. POP) reads this list and updates its history file configuration to include the MARBL diagnostics appropriately
  - Ocean host model should not need any semantic knowledge of MARBL parameters or its diagnostic - doesn’t know anything about MARBL configuration unless MARBL explicitly shares it
- Example
  - “ciso_on” is a MARBL variable enabling carbon isotopes
  - If “ciso_on” is true, MARBL tells POP to expect 14 more tracer tendencies from MARBL and include additional diagnostics

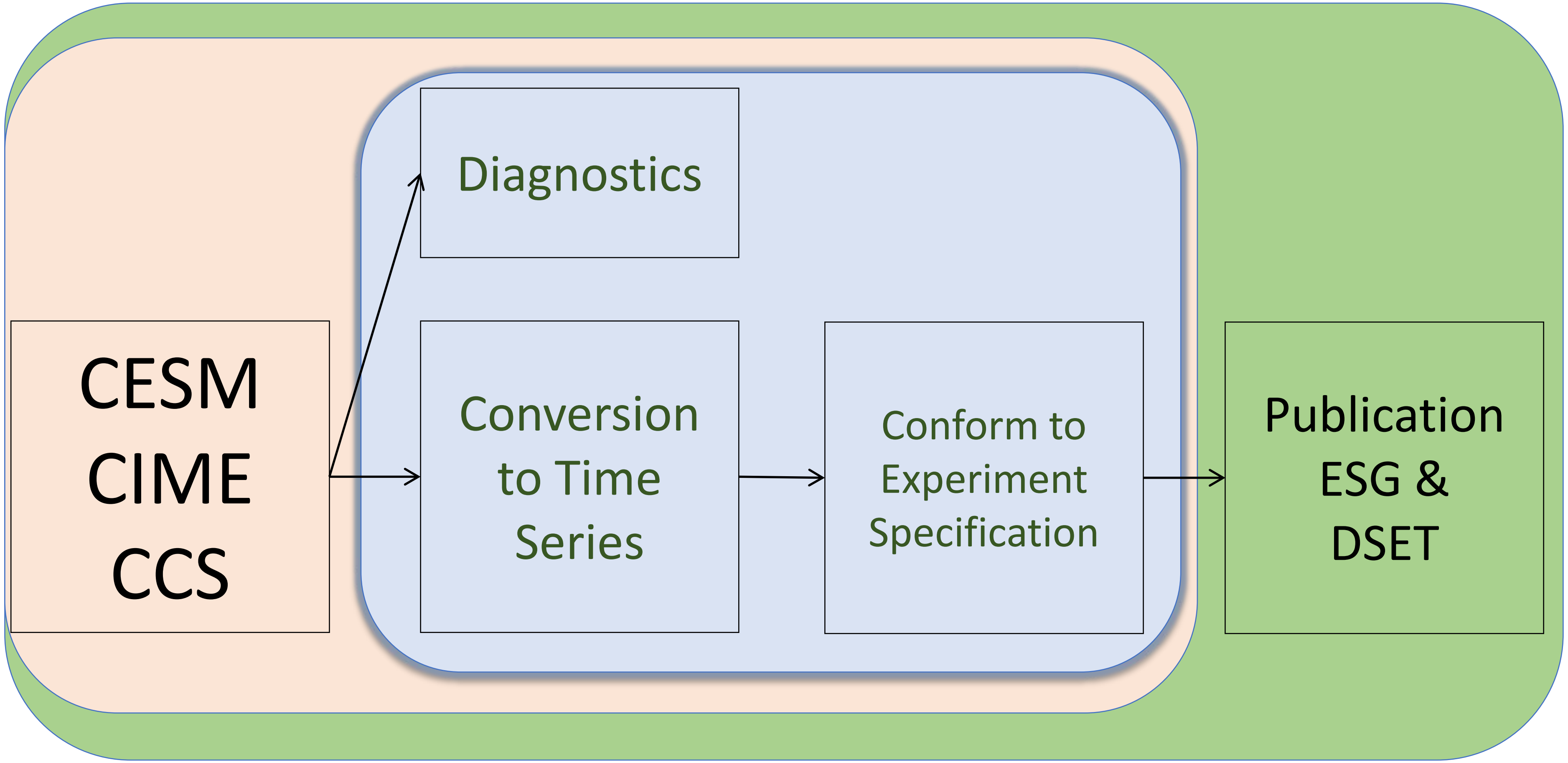
Workflow in CMIP6

Alice Bertini
Sheri Mickelson
Kevin Paul
John Dennis

CESM python based parallel post-processing tools

Cylc Workflow Engine (state machine that is automated)

CESM2.0 Experiment Database (expert knowledge – manual)
experiment case reservations, metadata archival and status information



CESM
CIME
CCS

Diagnostics

Conversion to Time Series

Conform to Experiment Specification

Publication ESG & DSET

CESM python based parallel post-processing tools

http://github.com/NCAR/CESM_postprocessing

- Diagnostics
  - NCL-based atmosphere, land, land-ice, sea-ice, ocean
  - Python-based ILAMB/IOMB land and ocean biogeochemistry
  - High-resolution ocean (0.1 degree)
- Single variable time series generation
- CF-compliant conformer tool
- Support for Cheyenne and Geyser

Thanks to Alper Altuntas(NCAR), Ben Andre(NCAR), Nate Collier (ORNL), Jan Laenerts (CU)

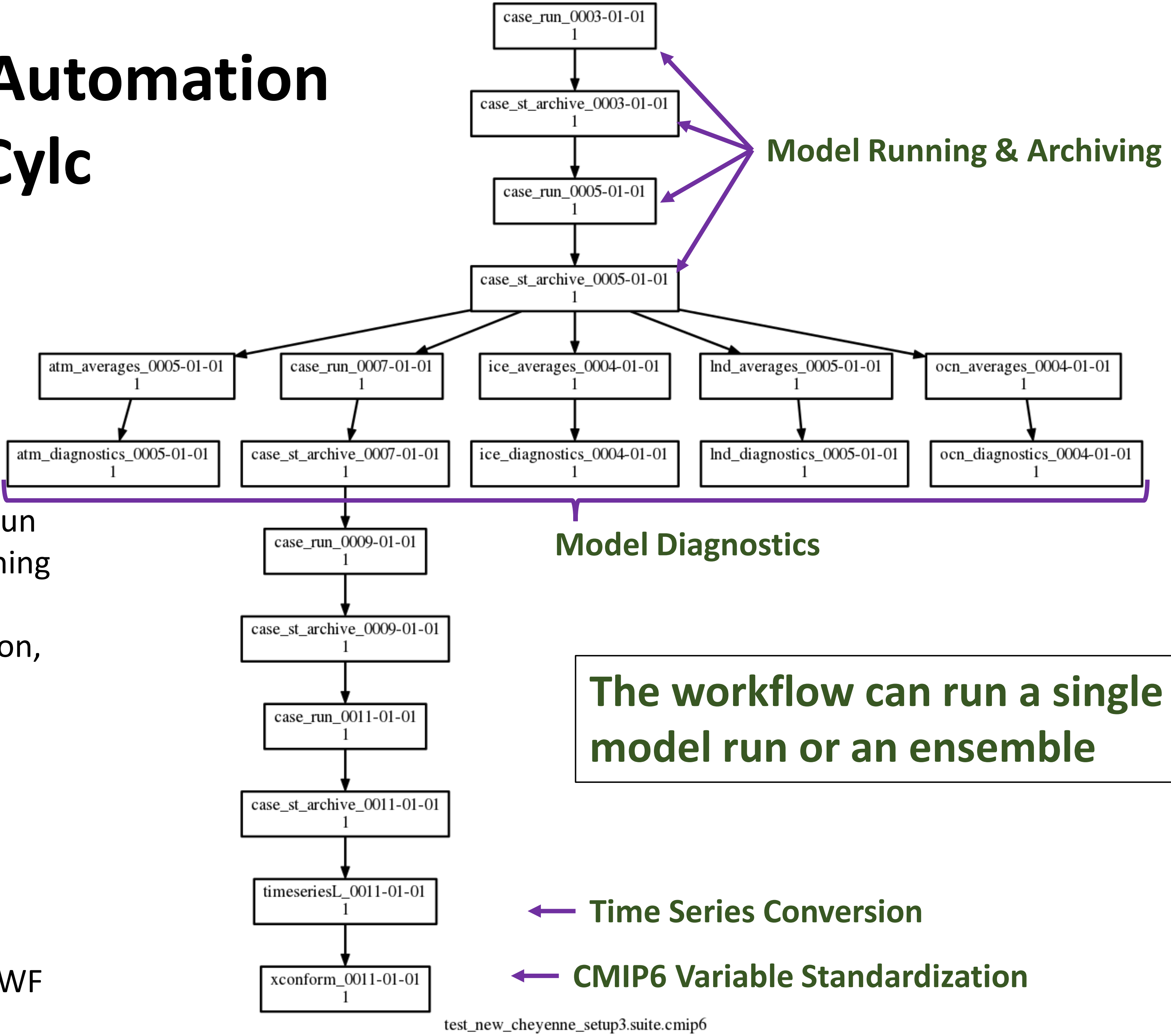
* Updates in the last year

CESM Workflow Automation for CMIP6 using Cylc

You can now run a script that will:

- create the CESM case
- setup the full post-processing environment
- create a Cylc suite for them customized to user’s settings.
- once you build CESM and hit a run button on the Cylc GUI - everything will be ran for you (archiving, diagnostics, timeseries conversion, variable standardization)
- order dependencies will be preserved
- An example workflow is shown here.

https://cylc.github.io/cylc/
https://github.com/NCAR/CESM-WF



case_run_0003-01-01
1
case_st_archive_0003-01-01
1
case_run_0005-01-01
1
case_st_archive_0005-01-01
1
atm_averages_0005-01-01
1
case_run_0007-01-01
1
ice_averages_0004-01-01
1
lnd_averages_0005-01-01
1
ocn_averages_0004-01-01
1
atm_diagnostics_0005-01-01
1
case_st_archive_0007-01-01
1
ice_diagnostics_0004-01-01
1
lnd_diagnostics_0005-01-01
1
ocn_diagnostics_0004-01-01
1
case_run_0009-01-01
1
case_st_archive_0009-01-01
1
case_run_0011-01-01
1
case_st_archive_0011-01-01
1
timeseriesL_0011-01-01
1
xconform_0011-01-01
1
test_new_cheyenne_setup3.suite.cmip6

Model Running & Archiving

Model Diagnostics

Time Series Conversion

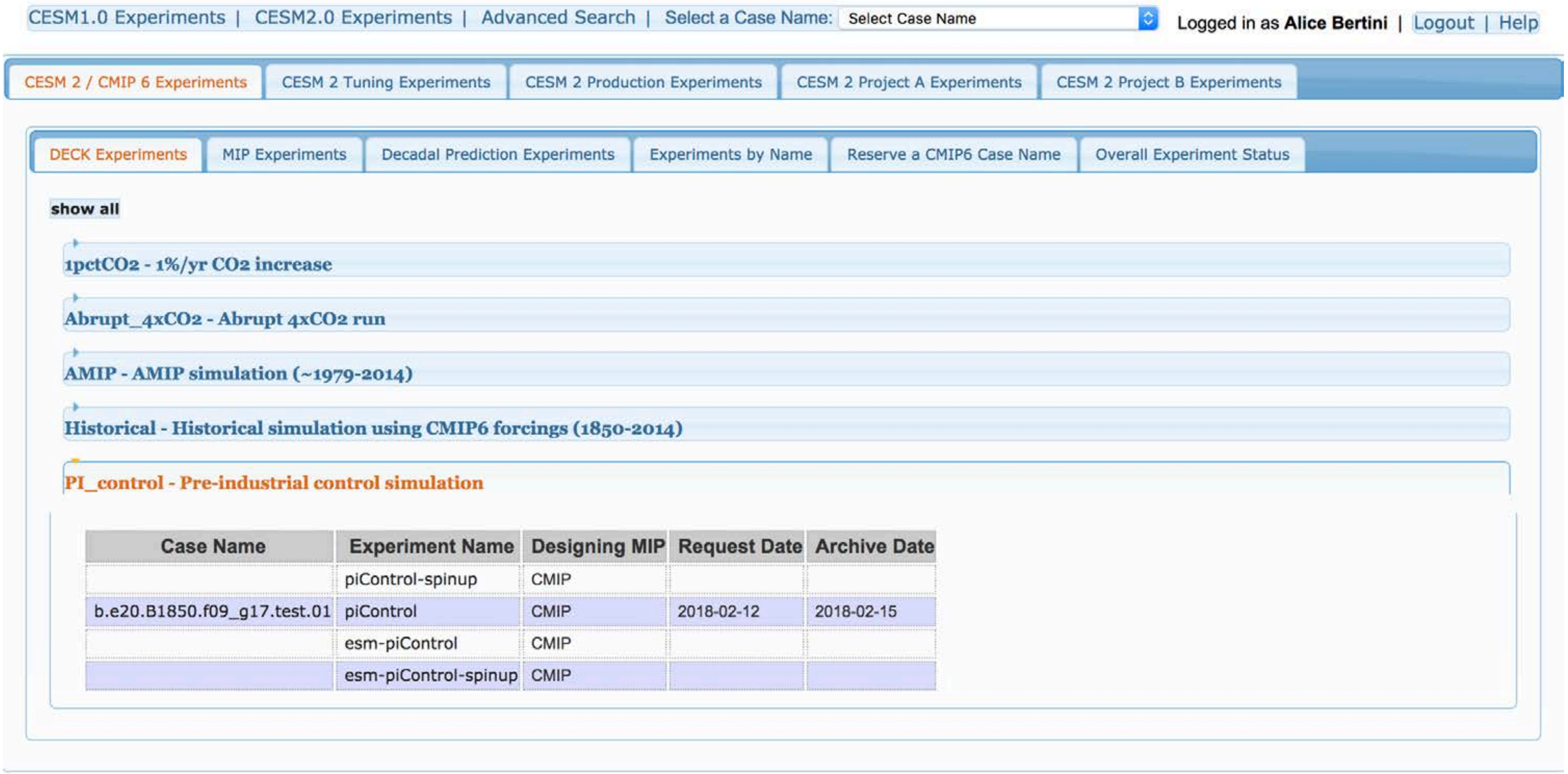
CMIP6 Variable Standardization

The workflow can run a single model run or an ensemble

CESM2.0 Experiment Database

experiment case reservations, metadata archival and status information

http://csegweb.cgd.ucar.edu/expdb2.0 *


CESM1.0 Experiments | CESM2.0 Experiments | Advanced Search | Select a Case Name: Select Case Name
Logged in as Alice Bertini | Logout | Help
CESM 2 / CMIP 6 Experiments
CESM 2 Tuning Experiments
CESM 2 Production Experiments
CESM 2 Project A Experiments
CESM 2 Project B Experiments
DECK Experiments
MIP Experiments
Decadal Prediction Experiments
Experiments by Name
Reserve a CMIP6 Case Name
Overall Experiment Status
show all

1pctCO2 - 1%/yr CO2 increase

Abrupt_4xCO2 - Abrupt 4xCO2 run

AMIP - AMIP simulation (~1979-2014)

Historical - Historical simulation using CMIP6 forcings (1850-2014)

PI_control - Pre-industrial control simulation

Case Name	Experiment Name	Designing MIP	Request Date	Archive Date
	piControl-spinup	CMIP		
b.e20.B1850.f09_g17.test.01	piControl	CMIP	2018-02-12	2018-02-15
	esm-piControl	CMIP		
	esm-piControl-spinup	CMIP		

Experiment database provides ability to reserve a case name, track the status of the various runs that are occurring and interface with the publication of the data

Questions?

Extra Slides

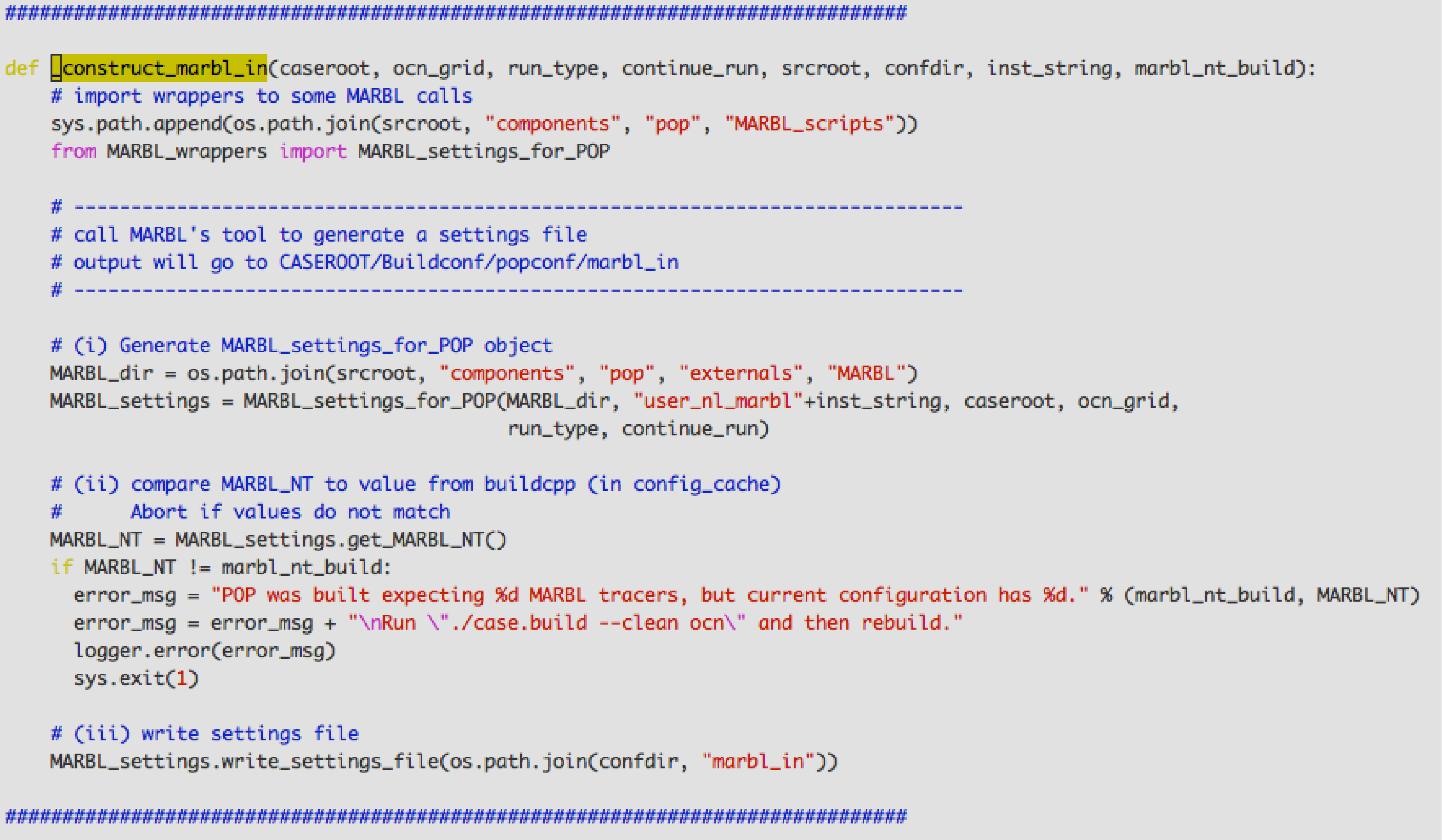
How do CESM Users Change MARBL Configuration?

- Changing parameters in `marbl_in`
  - Edit `user_nl_marbl` in `$CASEROOT` (note that this file is not automatically created by `CIME` scripts otherwise process is identical to any other `CESM` component)
  - `POP` will not allow changes in `user_nl_marbl` that change tracer count if `BUILD_COMPLETE=TRUE`
- Changing diagnostic output from MARBL
  - Python scripts generate `$CASEROOT/Buildconf/popconf/ecosys_diagnostics`, which can be copied to `SourceMods/src.pop/` and edited
  - For consistency, this file also contains ecosystem-related diagnostics that POP generates (e.g. forcing fields and tracer state)

Cylc, CMIP6 and CESM2.0 Experiments Database Workflow Steps

1. Reserve a CESM casename for a CMIP6 experiment using the CESM2.0 web interface
2. Setup a cylc suite for new reserved casename
3. Cylc automatically calls caseroot archive_metadata script to interact with CESM2.0 experiments database
4. Cylc suite sends automatic status email updates to CESM2.0 experiments database
5. CESM2.0 web interface manual interaction for publication to ESG and DSET

Creation of MARBL input parameters



```
###############################################################################

def construct_marbl_in(caseroot, ocn_grid, run_type, continue_run, srcroot, confdir, inst_string, marbl_nt_build):
    # import wrappers to some MARBL calls
    sys.path.append(os.path.join(srcroot, "components", "pop", "MARBL_scripts"))
    from MARBL_wrappers import MARBL_settings_for_POP

    # ------------------------------------------------------------------------------
    # call MARBL's tool to generate a settings file
    # output will go to CASEROOT/Buildconf/popconf/marbl_in
    # ------------------------------------------------------------------------------

    # (i) Generate MARBL_settings_for_POP object
    MARBL_dir = os.path.join(srcroot, "components", "pop", "externals", "MARBL")
    MARBL_settings = MARBL_settings_for_POP(MARBL_dir, "user_nl_marbl"+inst_string, caseroot, ocn_grid,
                                            run_type, continue_run)

    # (ii) compare MARBL_NT to value from buildcpp (in config_cache)
    #      Abort if values do not match
    MARBL_NT = MARBL_settings.get_MARBL_NT()
    if MARBL_NT != marbl_nt_build:
      error_msg = "POP was built expecting %d MARBL tracers, but current configuration has %d." % (marbl_nt_build, MARBL_NT)
      error_msg = error_msg + "\nRun \"./case.build --clean ocn\" and then rebuild."
      logger.error(error_msg)
      sys.exit(1)

    # (iii) write settings file
    MARBL_settings.write_settings_file(os.path.join(confdir, "marbl_in"))

###############################################################################
```

Subroutine in `buildnml` *uses a wrapper to a [MARBL-owned] python class to create* `marbl_in`