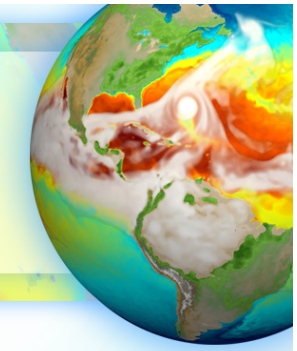
Impediments to regular testing of Scientific Software
or
CI vs. HPC Centers



Robert Jacob

E3SM Infrastructure Group Lead

Argonne National Laboratory

2019 Collegeville Workshop on Sustainable Scientific Software

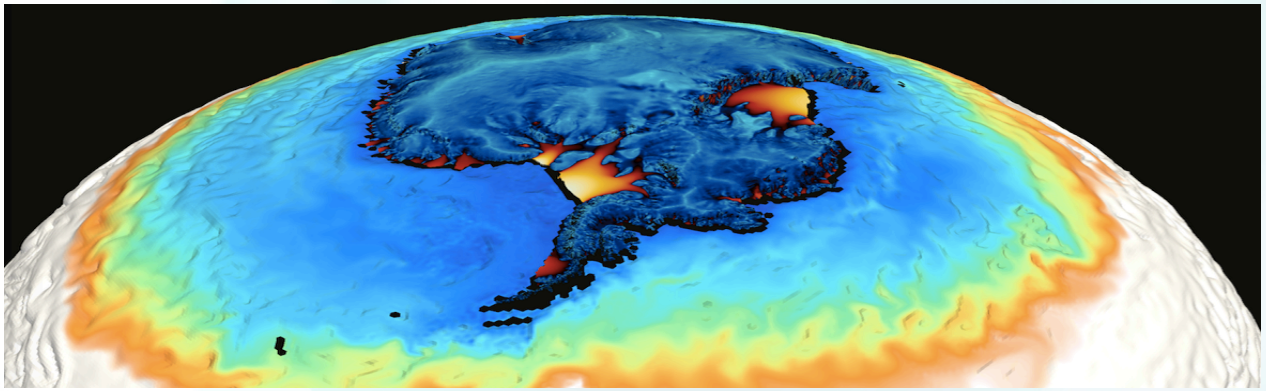
Collegeville, MN.

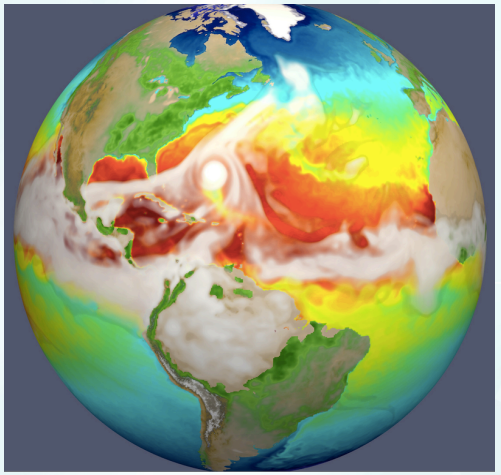
July 23, 2019

E3SM Energy Exascale Earth System Model

U.S. DEPARTMENT OF ENERGY

Energy Exascale Earth System Model

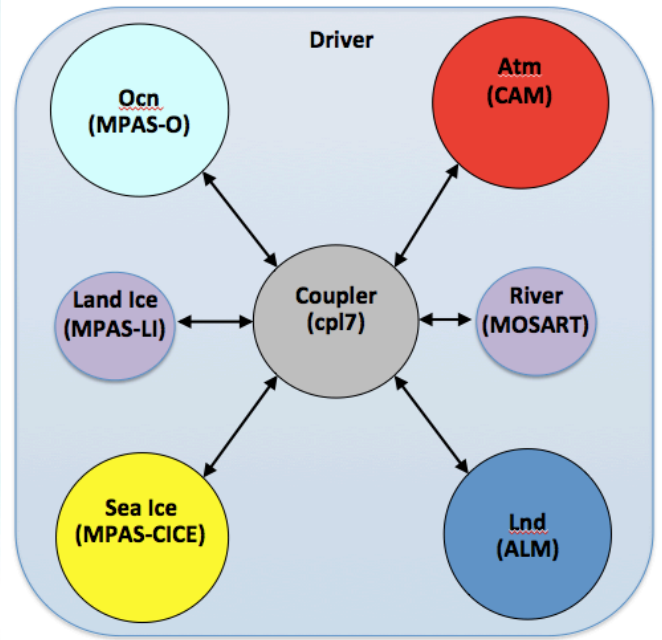




- "A collaboration among the DOE national to develop and apply the most complete, leading-edge climate and Earth system models for the most challenging and demanding climate-change research problems and DOE mission needs while efficiently using DOE Leadership Computing Facilities."
- 8 U.S. national laboratories and 6 partner institutions
- Total effort: ~43 FTE
- Funding: approx $22M/year. DOE Office of Science, Office of Biological and Environmental Research.
- See e3sm.org for more.

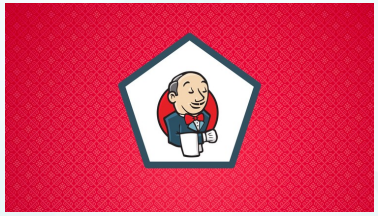
E3SM IG 3 major areas of concern

- Develop, maintain and support software that is needed for E3SM but is **not** part of the main prognostic models (the AGCM, OGCM, etc.). Build system, test system, coupler, data models, diagnostic/analysis software.
- Manage data sets
- Define, document, manage the process and procedures used in all software development within the E3SM Project



CI: Continuous Integration

- Take all the features that are ready to be added to master each day and test them together.
  - Do this before actually merging to master.
  - If it's a pass, can "graduate" to master.
  - If a fail, fix or revert.
  - Have Jenkins or similar service run tests automatically every night



- Necessary for a software base that is modified frequently
  - Testing provides confidence that your modifications behave as expected.

- Ideally: complete code coverage for each iteration of your CI test suite.
  - All execution paths through your software. Use multiple tests to get there.
  - All compiler/machine combinations of interest. Run on a "zoo" of machines.

Non-controversial statement: Automated regular testing is a necessary (but not sufficient) condition for Software Sustainability

CI and E3SM

Does E3SM need CI? Oh yes.

- Is modified frequently
  - Starting August, 2014, E3SM has **averaged** 1 PR/day, every calendar day.
    - Inflated somewhat by per-machine build configs included in source
    - Doesn't count the PRs in from components included as git submodules.
- Has many developers
  - 114 contributors on github.
- Has many simultaneous development goals
  - Components of E3SM (the atmosphere, ocean, land, etc. models) all have their own development goals. Work on parallel tracks and combine to improve simulations in 3 areas: water cycle, biogeochemistry, cryosphere. Major development proposed and reviewed every 3 years.

Expected challenges for CI with E3SM.

- E3SM has many possible execution paths
  - E3SM is actually several climate-science models in one code base. A very “heterogeneous solver”
    - Fully coupled, OGCM with fixed atmosphere, AGCM with fixed SST, active land only, …
  - Each configuration has run/compile time switches to further increase the number of paths.
    - Diagnostics on/off, experimental param. schemes on/off, low/high resolution, etc.

- E3SM is mostly Fortran
  - Experience has shown that large Fortran codes can get different behavior from different Fortran compilers (Intel, GNU, PGI, etc.) and even different versions of the same compiler.
    - Everything from internal compiler errors to runtime errors to climate-changing differences to roundoff.
  - Successful testing with one compiler on one machine is misleading.
  - Also need to test with full optimization flags on and then again with debugging flags.

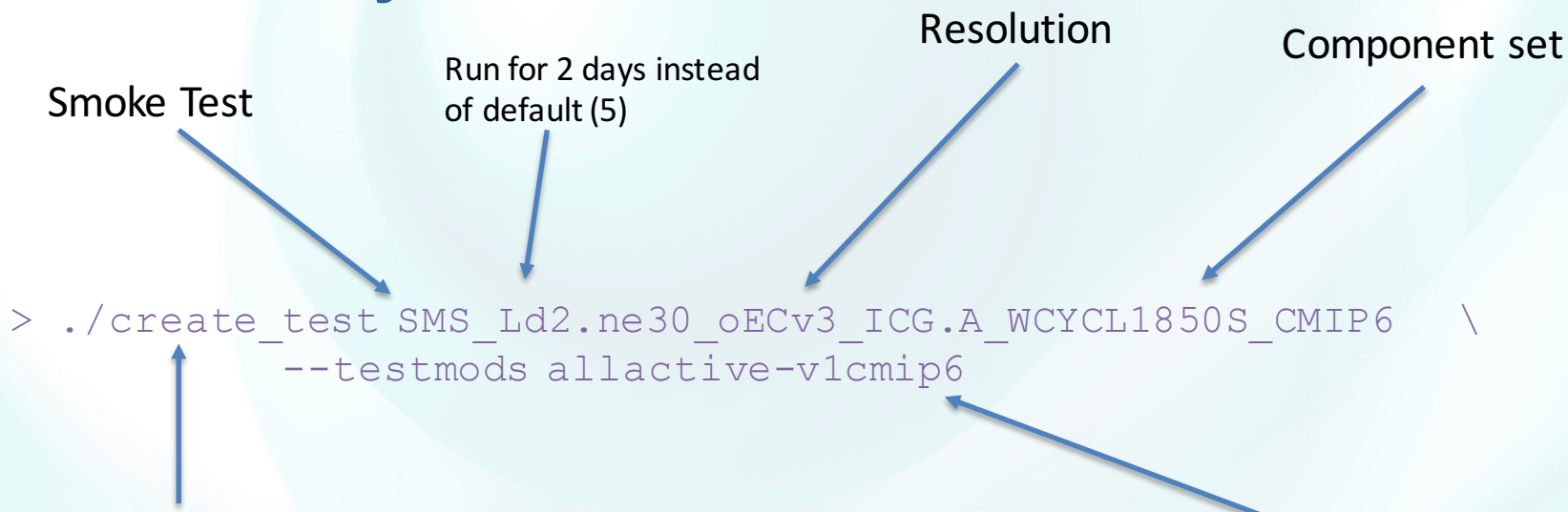
Brief look at how E3SM does testing

- Climate models work hard to guarantee bit-for-bit reproducible results under many circumstances.
  - Necessary to avoid expensive and not-yet-automated testing for “identical” climates.

- E3SM/CESM share the Case Control System from the Common Infrastructure for Modeling the Earth (CIME) which provides these test types and system test infrastructure.

- CIME provides about 30 system test types. Examples:
  - “SMS” smoke test. Run successfully for some amount of simulated time.
  - “ERS” Exact restart. bit for bit when starting from checkpoint.
    - Do 3 runs: one long, 2 short with second using a checkpoint from the first. End at same time. Should be bfb.
  - “PEM” bit for bit when changing task count
    - Do 2 runs, one with default task layout, one with Ntasks/2. Should be bfb.

An E3SM system test



Custom python program to parse, set up and build executable, run test.

Machine is auto-detected from hostname where command is executed (can be specified)

Default compiler and flags are specified in XML config file in source.

Optional set of runtime settings

E3SM test suites

- Combine test types with different compsets and resolutions used by E3SM to form **test suites**
  - e3sm_integration: run overnight on version of the day (VOTD). 71 tests.
  - e3sm_developer: subset of e3sm_integration, run in an hour or so by individual developer on their development cluster (or maybe a big workstation)
  - e3sm_highres: Run high-resolution configs on high-node-count platforms that can accommodate them.

Lets be a naïve E3SM developer...

- I'm developing a key DOE Office of Science application.
- I'd like to follow good programming practices and use CI on the DOE Office of Science compute platforms my application is targeting.
- Lets set up our test suites to run every night on the platforms DOE has made available for its science!

Obstacle 1: No overnight turnaround

- On busy computers at OLCF, ALCF, NERSC, no way to guarantee your test suite will finish in the morning if it is above a certain size.
  - Queue wait times depend on other users and there are a lot of users.
  - “Premium” queues still aren’t a a guarantee and would make testing more expensive.
  - Some queues have a minimum job size to start quickly.

- “Debugging” queues offer fast turnaround BUT aimed at single developer doing repeated edit-compile-run-analyze cycles.
  - Limit number of simultaneous jobs per-user.
  - Limit the total number of nodes.
  - Limit the total wallclock time (less then 4 hours)

Obstacle 2: No automatic test runs

- 2-factor authentication means external Jenkins service can't connect to start testing. There are a few ways around that.
  - Ask nicely for an exception for one specific user coming from specific IP address (NERSC)
  - If allowed, use a cron process to start suite (ALCF).
    - Even better, a local Jenkins instance.
  - If forced to, designate a human to log on and run suite (OLCF)

Obstacle 3: Not enough core hours available.

- Core-hour proposal process assumes code is ready-to-run.
  - Must carefully map requested core-hours to planned production simulations (that will result in papers).
  - "computational readiness" is actually a required section in INCITE proposals.
  - Proposals are very competitive. LCFs let anyone in the world apply.
  - You can usually add 10% to your total request for "debugging or mistakes".
- "discretionary" accounts are easy to get but:
  - Available for limited time
  - Designed to help you calculate your core-hour request accurately after some tuning/measuring of your ready-to-run code. Not for CI or long-term development.

Coping strategies 1

- Find cycles from locally available machines.
  - Sandia had 2 “lightly” used clusters with standard CPU nodes (1200 and 1800 nodes)
    - But only Sandia staff can access
  - Argonne had a lab-wide cluster large enough.
    - And its possible for non-Argonne E3SM staff to use.
  - BER spent program dollars on its own hardware (200 and 400-node clusters so far)

- Install similar software stack where possible (Compiler and compiler version)

- Limitations
  - Still not testing at scale of 1K-10K nodes/tasks that we will be running on. Can easily miss flagging code that only has problems on high task counts.
  - Still not testing on the actual hardware/software stack we’ll use in production.
  - Debugging test fails on the Sandia-only machines is painful if the developer is not a Sandia employee.

Coping strategies 2 (because we still don't have enough time)

- Spend time making configurations that are "ultra low" resolution
  - Most straightforward way to reduce resource requirements of tests.
- Spend time figuring out just how many timesteps are needed for a given system test to work.
- Instead of all possible configurations, test most-used configurations
- Don't use all the test types for selected configurations.
- Limitations
  - This is a lot of work
  - May not touch all code paths when running at low resolution.
  - When a new configuration starts to get more use, may not make it in to test suite immediately.
  - Its easy for end-users to configure the model for a case that has never been tested.

E3SM test suite result

- After absorbing all the limitations imposed on our testing
  - Reduce number of test types
  - Reduce number of component configurations
  - Reduce resolution to “ultra-low” in most cases.
  - Reduce total simulation to bare minimum (9 time steps in some cases)
  - Run mostly on available lab clusters.
  - For LCF, NERSC overnight, reduce to absolute bare minimum (e3sm_production)

We have some confidence that our software is working and can be changed with confidence. Still get surprises.

E3SM_Baselines *7 builds* [view timeline]

		Configure		Test			
Site	Build Name	Error	Warn	Not Run	Fail	Pass	Start Time
melvin	e3sm_developer_master_gnu	0	0	0	0	38	18 hours ago
melvin	e3sm_developer_next_gnu	0	2_{+2}	0	1^{+1}	37_{-1}	18 hours ago
sandiatoss3	e3sm_integration_master_intel	0	0	0	0_{-3}	71^{+3}	19 hours ago
sandiatoss3	e3sm_integration_next_intel	0	10_{+10}	0	10^{+10}_{-3}	61^{+3}_{-10}	19 hours ago
cori-knl	e3sm_prod_maint-1_0_intel	0	0	0	0	2	Jun 09, 2019 - 23:16 UTC
anvil	e3sm_prod_next_intel	0	0_{-1}	0	0	2	2 hours ago
cori-knl	e3sm_prod_next_intel	0	0	0	0_{-1}	2^{+1}	Jun 13, 2019 - 03:50 UTC

Desired changes

- At the center level:
  - Provide internal Jenkins service or at least allow cron jobs
- At the Office of Science level
  - Make testing a welcome consumer of cycles at centers.
  - Require explanation of testing process in computer time proposals.
  - Allow testing core hours to be significant (1/2) portion of total request.
  - Allow testing-only proposals.
  - Provide more cycles to fit expanded testing role.

Thank you!

(Disclaimer: all opinions are my own and not necessarily those of E3SM)