

Exascale Computing Project Annual meeting summary

Fri@CEESD - April 23, 2021

Luke Olson, Matthias Diener, Matt Smith, and Mike Campbell

Software development, environment, and testing

- ▶ E4S: Extreme-Scale Scientific Software Stack
 - Pantheon - workflow management
 - DOE Code
 - ECP CI
- ▶ IDEAS Team: Better Scientific Software through better practices

E4S: Extreme-Scale Scientific Software Stack

Community and infrastructure for developing and deploying software in HPC environments.

- ▶ Heavily spack-based
- ▶ Supercontainers (e.g. Docker, Singularity, Shifter)
- ▶ Turnkey environments
- ▶ Standardization of HPC software deployment
- ▶ Collection of build caches
- ▶ ECP stacks, tested, deployed on HPC platforms (ECP CI)
- ▶ Extreme-Scale Scientific Software Development Kits xSDK
 - Math Libraries
 - Data and Visualization
 - Scientific Workflows (Pantheon)
 - Software Ecosystem (DOE Code, ECP CI)



<https://e4s-project.github.io>

E4S for CEESD

► Use E4S:

- *MIRGE* in an E4S turnkey environment
- Simplify deployment on emerging platforms
- Accelerate builds and runtime (E4S Build Cache, RAM cache)

► Join E4S:

- Requirements (<https://e4s-project.github.io/policies.html>)
- Leverage E4S validation test suite (automated deployment!)
- Resources (a throat to choke)
- Increased visibility/exposure



<https://www.exascaleproject.org/research-project/e4s-and-sdk-efforts/>

Pantheon: Reproducible workflows for HPC

► What is it?

- Open standard for workflow specification (<https://pantheonscience.org/standards>)
- YAML-based (install, run, postprocess, validate)
- What's under-the-covers is your business (e.g. Parsl, conda)
- Unification of job design for

all supported platforms
(includes all lab platforms)

► How can we use it at CEESD?

- Publish *MIRGE* production workflows
- Manage/re-imagine multi-platform *MIRGE* performance monitoring/benchmarking



PANTHEON

<https://pantheonscience.org>

<https://github.com/cinemascienceworkflows>

DOE Code

DOE CODE

U.S. Department of Energy
Office of Scientific and Technical Information


(<https://osti.gov/doecode>)

- ▶ DOE establishes DOI for the code
- ▶ Maintain/propagate legacy of the DOE-funded code
- ▶ Reduces the big messy pile of DOE-funded code
- ▶ Helps resolve ongoing licensing debacle

ECP Continuous Integration

- ▶ CI servers at most major supercomputing sites
 - Operates from site-hosted gitlab instances
 - All currently operating lab-based machines (emerging supported quickly)
 - Regularly exercises ECP codes w/ E4S stacks
- ▶ Two ways CEESD could leverage ECP CI:
 - Independent DOE-funded code mirror
 - E4S stack inclusion

General ECP-CI Info, documentation: (<https://ecp-ci.gitlab.io>)
LLNL/LC (including Lassen): (<https://lc.llnl.gov/gitlab>)

- 
- 1 Customize and curate methodologies**
 - Target scientific software productivity and sustainability
 - Use workflow for best practices content development
 - 2 Incrementally and iteratively improve software practices**
 - Determine high-priority topics for improvement and track progress
 - *Productivity and Sustainability Improvement Planning (PSIP)*

- 3 Establish software communities**
 - Determine community policies to improve software quality and compatibility
 - Create Software Development Kits (SDKs) to facilitate the combined use of complementary libraries and tools
- 4 Engage in community outreach**
 - Broad community partnerships
 - Collaboration with computing facilities
 - Webinars, tutorials, events
 - *WhatIs* and *HowTo* docs
 - Better Scientific Software site (<https://bssw.io>)

For more about our work see this report:
<https://doi.org/10.2172/1606662>

IDEAS
productivity

ECP
EXASCALE
COMPUTING
PROJECT

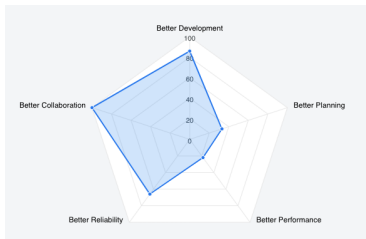
(<https://ideas-productivity.org/ideas-ecp>)

Software process and practice



- ▶ Better Scientific Software (<https://bssw.io>)
 - Best practices - development and project management
 - Training materials and resources
- ▶ Some observations questions, and recommendations:
 - Your software will live longer than you expect, so plan for it now (software lifecycle)
 - Consider stand-alone, separate testing suites
 - Restart testing is highly valuable.
 - Resource adaptation is highly valuable.

- There is a deep interest in performance tracking over time (Kitware supported)
- Testing tax is mostly imagined. The cost of defects is extremely high.
- Onboarding and offboarding procedures are highly valuable.



Better Scientific Software

- ▶ Improving productivity for users with computing workflows
- ▶ Best practices - for developing and in distributed teams and environments
- ▶ Software practices: Code will live longer than you think (plan now)
 - Testing suites as stand-alone software suites
 - Test-driven development (TDD)
 - Actually test fault detection
 - Focus on exercising your code
- ▶ Community guidelines for testing (in-general) needs to be developed:
 - No testing is not enough
 - What is the risk of bugs creeping in?
 - How much time do you spend on defects?
 - How much compute time do you waste computing a spurious result?
- ▶ With HPC risk level for defect is very high due to cost of runs
- ▶ Restart tests are highly valuable
- ▶ Code review is highly valuable (think pair coding similar in function, but cost 2x as high)
- ▶ Testing as a part of the development is extremely important
- ▶ Don't forget to test the tests, inject errors on purpose to see if the test catches

bof1

- ▶ build caches of common stacks
- ▶ OS updates underneath can be very costly
- ▶ dependencies work to update is very high
- ▶ Spack based (heavily)
- ▶ Container based (heavily)
- ▶ E4S is the preferred path to contributing to spack
- ▶ ECP projects
 - Vis and analysis
 - tools and building
 - Scientific Workflows
- ▶ e4s-project.github.io/policies.html
- ▶ Supercontainers - containers on the compute platforms
 - Shifter(Cori), singularity(lassen), charliecloud
 - Looks like spack-based build/install can use CI on compute platforms
 - 1. Spock and Articus
 - 2. Make a spack recipe
 - 3. Create smoke tests
 - Support: a throat to choke

bof2

How to set goals for software improvement?

- ▶ How to measure performance - what metrics
- ▶ performance/dollar
- ▶ performance/watt
- ▶ performance: run faster, don't care how much it costs
- ▶ threshold performance: what performance is required to get done in time
- ▶ relative performance: performance b / performance a
- ▶ Important to consider amount of science work done / walltime (accuracy of result included) ECP uses this
- ▶ Identify Figure of Merit (FOM) - that characterizes your performance relative to your end/capstone goal
- ▶ FOM1: $\# \text{ DOF} \times \# \text{ CG iters} / \text{time}$
- ▶ FOM2: $\text{fac1} * \# \text{ gp} \times \text{fac2} * \# \text{ part} \times \text{nsteps} / \text{time}$ (facs function of grid push and particle push)
- ▶ Observing high overheads for kernel launch times
- ▶ Parting thoughts:
 - No such thing as *just* porting - refactoring and reprogramming for performance on modern systems touches your code intimately



Better software

- ▶ What are your software pain points? (legacy code with no tests) - Team turnover - Resources to spend on better software (what if you had more funding, what would you spend it on?) - Prioritizing science *with the code* over science *into the code* over attention to software process and improvements - short time science vs. long-term science : spending on software process and improvements invests in long-term science - Reproducibility is a big deal - Incorporate these things into your software process, or your code is quickly irrelevant - quickly changing machines, etc
- ▶ What resources do you use to improve your software process? - BSSW site - Software resource management
- ▶ What would help with your SW development going forward?
- ▶ How could better SW practices reduce risk in your project? - Transient developers (team changing in and out) remain useful
- ▶ How do you convince people to improve SW process? - How do you convert dug-in skeptics? If you are the skeptic, what would convince you?



Questions?

This material is based in part upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number DE-NA0003963.