



Managing Computational Experiments

Jared O'Neal (he/him) & Anshu Dubey (she/her)
Argonne National Laboratory

Software Productivity and Sustainability track @ Argonne Training
Program on Extreme-Scale Computing summer school

Contributors: Jared O'Neal (ANL), Anshu Dubey (ANL)




See slide 2 for
license details



License, Citation and Acknowledgements

License and Citation

- This work is licensed under a [Creative Commons Attribution 4.0 International License](#) (CC BY 4.0). 
- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Todd Gamblin, Jared O'Neal, and Boyana R. Norris, Software Productivity and Sustainability track, in Argonne Training Program on Extreme-Scale Computing, St. Charles, Illinois, 2022. DOI: [10.6084/m9.figshare.20416215](#).**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial, ISC, 2022 ...

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at the Los Alamos National Laboratory, which is managed by Triad National Security, LLC for the U.S. Department of Energy under Contract No. 89233218CNA000001
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
- This work was performed in part at University of Oregon through a subcontract with Argonne National Laboratory.

My high-level experience

- I try to adapt tips, tools, & techniques to computational science world
- Doing computational science work is hard
- What is supposed to make the work easier can often make it harder
- I miss simple, physical laboratories
 - Work in one space and on the same machines with all tools at hand
 - Evolution of machines and tools is slow
 - Your collaborators are usually co-located, ready to share, and ready to talk
 - Paper lab notebooks are “good enough”
- I experienced something like an apprenticeship

We never discussed the W in DIKUW!

The accumulated wisdom of a community

In life and in labs we learn and repeat short phrases

General

- The devil's in the details
- If it's worth doing, it's worth doing well
- Perfect is the enemy of good
- It's better to be correct, than fast

Hardware

- Know your tools
- Use the right tool for the job
- Measure twice, cut once
- Clean your workspace

Software

- Code is meant to be written once and read many times
- Premature optimization is the root of all evil – Donald Knuth
- Beware of code smells
 - Refactor or reassess design

Know your tools / Right tool for the job

Experimental World

- What is the tool's intended use?
- How does the tool work?
- Will this damage the tool?
- Will this damage the work piece?
- Will I get a better result if I use a different tool?
- Will a different tool be more efficient?

Computational World

- If a function has 10 optional variables, you should know what each does and set each one.
- If you design a library, make sure that users can learn how to use your tool.
- Is it OK to use high-level productivity tools blindly?

Know your tools: Example

- Explicit software stack construction – don't get it right by accident or coincidence
- `ldd` prints external dependencies
 - Include `ldd` output in build & job logs

Python needed by build system only

SW Environment at Build

```
Currently Loaded Modules:
 1) gcc/8.2.0-g7hpkz   2) intel-mkl/2018.4.274-2amycpi
 3) intel-mpi/2018.4.274-ozfo327  4) hdf5/1.10.5-vozfsah  5) anaconda3/5.2.0

[[0]] (publi) joneal@beboplogin2:bebop > ldd bin/my_bin_gcc_omp.x
linux-vdso.so.1 => (0x00007ffcd592000)
libmkl_gf_lp64.so => /soft/anaconda3/5.2.0/lib/libmkl_gf_lp64.so (0x00002b60ca32a000)
libmkl_gnu_thread.so => /soft/anaconda3/5.2.0/lib/libmkl_gnu_thread.so (0x00002b60cae0f000)
libmkl_core.so => /soft/anaconda3/5.2.0/lib/libmkl_core.so (0x00002b60cc522000)
libgomp.so.1 => /soft/anaconda3/5.2.0/lib/libgomp.so.1 (0x00002b60d052b000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00002b60d074e000)
libgfortran.so.5 => .../gcc-8.2.0-g7hpkz/lib64/libgfortran.so.5 (0x00002b60d096a000)
libm.so.6 => /lib64/libm.so.6 (0x00002b60d0ddb000)
libdl.so.2 => /lib64/libdl.so.2 (0x00002b60d10dd000)
libz.so.1 => /soft/anaconda3/5.2.0/lib/libz.so.1 (0x00002b60d12e1000)
libmpifort.so.12 => .../gcc-8.2.0/intel-mpi-2018.4.274-ozfo327/.../libmpifort.so.12 ...
libmpi.so.12 => .../gcc-8.2.0/intel-mpi-2018.4.274-ozfo327/.../libmpi.so.12 ...
librt.so.1 => /lib64/librt.so.1 (0x00002b60d2530000)
libgcc_s.so.1 => /soft/anaconda3/5.2.0/lib/libgcc_s.so.1 (0x00002b60d2738000)
libquadmath.so.0 => /soft/anaconda3/5.2.0/lib/libquadmath.so.0 (0x00002b60d294a000)
libc.so.6 => /lib64/libc.so.6 (0x00002b60d2b7b000)
/lib64/ld-linux-x86-64.so.2 (0x00002b60ca106000)
```

SW Environment at Execution

```
Currently Loaded Modules:
 1) gcc/8.2.0-g7hpkz   2) intel-mkl/2018.4.274-2amycpi
 3) intel-mpi/2018.4.274-ozfo327  4) hdf5/1.10.5-vozfsah

[[0]] (publi) joneal@beboplogin2:bebop > ldd bin/my_bin_gcc_omp.x
linux-vdso.so.1 => (0x00007ffefdf00000)
libmkl_gf_lp64.so => .../gcc-8.2.0/intel-mkl-2018.4.274-2amycpi/.../libmkl_gf_lp64.so ...
libmkl_gnu_thread.so => .../gcc-8.2.0/intel-mkl-2018.4.274-2amycpi/.../libmkl_gnu_thread.so ...
libmkl_core.so => .../gcc-8.2.0/intel-mkl-2018.4.274-2amycpi/.../libmkl_core.so
libgomp.so.1 => .../gcc-8.2.0-g7hpkz/lib64/libgomp.so.1 ...
libpthread.so.0 => /lib64/libpthread.so.0 (0x00002b70fa325000)
libgfortran.so.5 => .../gcc-8.2.0-g7hpkz/lib64/libgfortran.so.5 (0x00002b70fa541000)
libm.so.6 => /lib64/libm.so.6 (0x00002b70fa9b2000)
libdl.so.2 => /lib64/libdl.so.2 (0x00002b70facb4000)
libz.so.1 => /lib64/libz.so.1 (0x00002b70faeb8000)
libmpifort.so.12 => .../gcc-8.2.0/intel-mpi-2018.4.274-ozfo327/.../libmpifort.so.12 ...
libmpi.so.12 => .../gcc-8.2.0/intel-mpi-2018.4.274-ozfo327/.../libmpi.so.12 ...
librt.so.1 => /lib64/librt.so.1 (0x00002b70fc106000)
libgcc_s.so.1 => .../gcc-8.2.0-g7hpkz/lib64/libgcc_s.so.1 (0x00002b70fc30e000)
libquadmath.so.0 => .../gcc-8.2.0-g7hpkz/lib64/libquadmath.so.0 (0x00002b70fc526000)
libc.so.6 => /lib64/libc.so.6 (0x00002b70fc766000)
/lib64/ld-linux-x86-64.so.2 (0x00002b70f3bed000)
```

Yellow ellipses in library paths indicate snipped content

Measure twice, cut once

Experimental World

- Some actions are not reversible
- Slow down!
- Think before you act
- Don't waste materials
- Don't waste time & effort

Computational World

- Do appropriate level of planning
- Don't waste energy
- Don't waste core hours
- Don't produce bad data
- Long-term efficiency
- Maximize scientific impact

Clean your workspace

Last 15 minutes of workday is for cleaning

Experimental World

- Leave a safe and clean area
- Don't lose tools or equipment
- Prevent damage to tools & equipment by others
- Respect your tools & clean them
- Leave equipment in obvious state

Computational World

- Write good commit messages
- Maintain issues, PRs, & docs up-to-date as you work
- Don't leave clones in undocumented intermediate state
- Communicate bugs & errors explicitly and obviously

Experimental laboratory environment

- In an experimental lab,
 - have all my tools at hand, clean, and ready for use,
 - know how to use my tools,
 - fully characterized and configured instrument,
 - understand broken or underperforming instrument performance, and
 - I want to take comprehensive lab notes quickly and easily.

The Goal

Concentrate on acquiring data, analyzing the data, drawing conclusions, designing next steps, and recording the work for reproducibility.

Computational laboratory environments

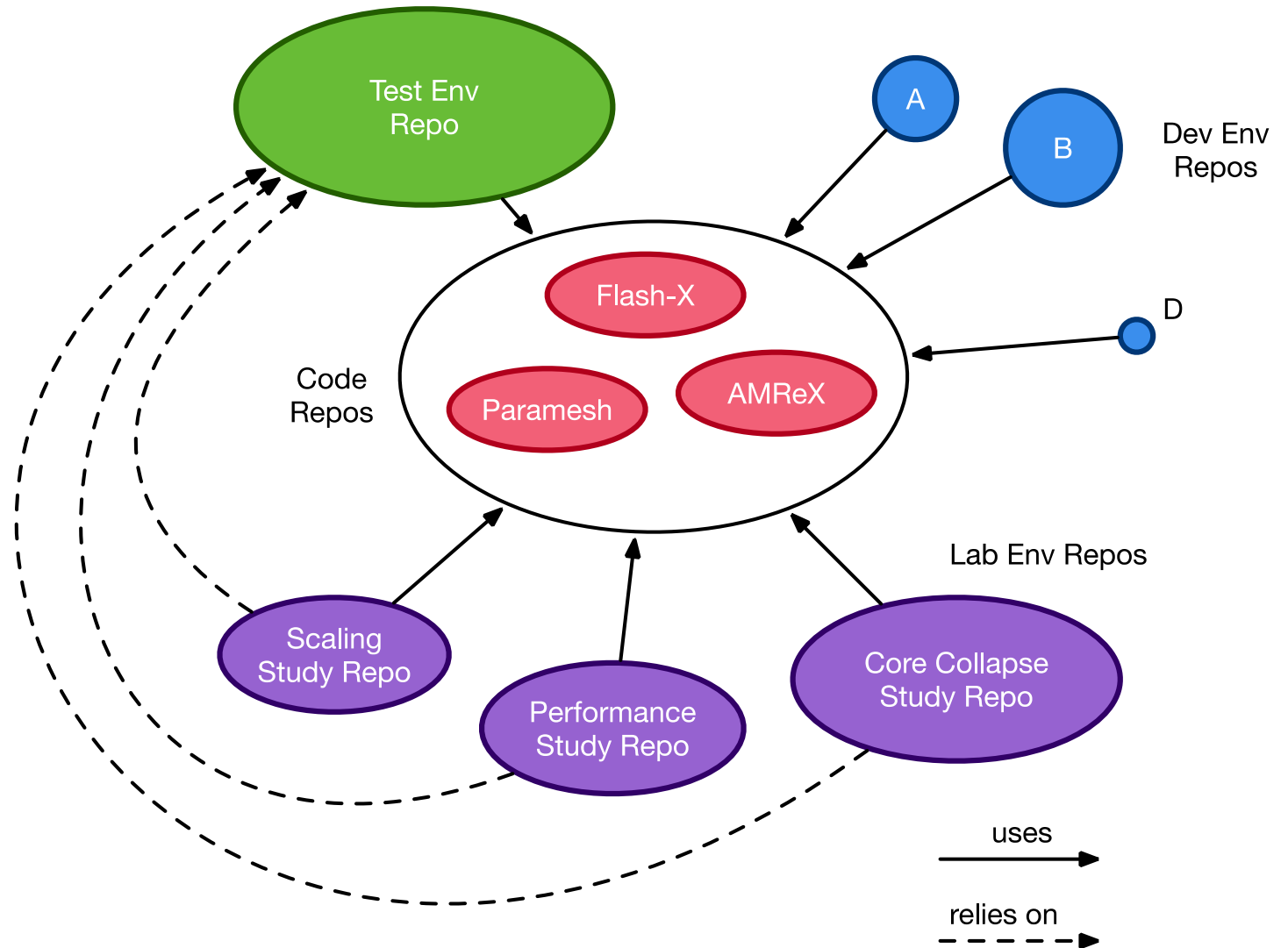
Use many simple, minimal lab environments

- Tailor formality, complexity, and automation to each use case
- Store context & metadata next to data
- Each code repository has dedicated test environment
- One environment per scientific study
- Each developer can have dedicated development environment (optional)
- Environments are encapsulated
 - Dedicated documentation
 - Dedicated software environments
 - Can be updated and managed independently

A system of repositories

- Each environment is encapsulated in an individual repository
- Some complexity transferred to interconnecting repositories
- Systems of repos exist & seem to be increasingly common

Oval size indicates environment size/complexity



Constructing computational lab environment

Start from the bottom

- File storage, protection, maintenance, & sharing
 - Try to avoid deep folder structures & long names
- Construct software environments
 - File management tools
 - Compilation & libraries (e.g., Spack env)
 - Data analysis tools (e.g., python virtual env)
- Platform-specific build systems
- Platform-specific job scripts
- Testing, verification, & validation
- Analysis tools
- Documentation infrastructure, & workflow
- Article infrastructure

Build, test, & officially verify.
Don't alter or update afterward.

Documentation

- We don't want a single 10,000 line README
- Lab notebook for changes to sci instrument
 - Changes in code repo necessary for study
 - Changes to SW environments
 - Changes to build/job files and build systems
- Lab notebook for data analysis tools
- Lab notebook to detail how experiment was designed and executed
- Right tool for the job
- Need flexibility to structure data and documentation in repo

Documentation: READMEs

- High-level road maps with motivation, documented decisions, and conclusions
- Concise living docs that function as executive summaries
- READMEs distributed through folder structure
 - Each sub-experiment has its own README?

Documentation: Version control tools

Locate lab notes associated with code “next” to the instrument

- Commit messages
- Issues to capture design discussions & requirements?
- Pull requests to document code review, verification/validation, *etc.*?

Documentation: Data context & metadata

- Automate as much as possible
- Build dates, user, system name, git hashes, configuration data in file headers
 - Self-documenting files
- A lot of this comes from build & job logs
 - Software environment info
 - git diffs
 - Environment variables

Documentation: Jupyter notebooks

- Jupyter notebook can put context & metadata next to data
 - High-level design & motivation up top
 - Low-level lab notes for acquiring data
 - Load & use data
 - Generate visualizations in place
 - Comment on results
- Figuring how to structure work into notebooks helps structure lab env
- Where do notebooks fit into the documentation hierarchy?

Is this working?

- Work in progress (and always will be)
- Used for different types of studies across different projects
- I'm not the only one
 - 2018 BSSW Fellow [Ivo Jimenez](#)
 - [Popper](#) tool for implementing scientific exploration pipelines that yield reproducible results
 - Aaron Lentner – George Washington University
 - [FlashKit](#) a high-level interface for helping users structure and manage research with Flash-X

For your exploration: [Computational Lab Environment Example](#)

So far we have talked about setting up the infrastructure that is needed to collect and analyze results.

We still haven't talked about how to plan running the study/experiments

This section talks about preparing for a successful simulation campaign

So far we have talked about setting up the infrastructure that is needed to collect and analyze results.

We still haven't talked about how to plan running the study/experiments

This section talks about preparing for a successful simulation campaign

Running simulations at large scales for science discovery is more of a craft and less of science. More than any other aspect of computational science it relies on experience and acquired wisdom that helps one develop a nose for fruitful possibilities.

Why do you need to plan?

- Machines are expensive and rare resources
 - Operating them is also very expensive
 - Many people are competing for these resources
 - You are likely charting new scientific territory

Why do you need to plan?

- Machines are expensive and rare resources
 - Operating them is also very expensive
 - If you made a mistake in your input parameters and got garbage results on a large scale run, you just wasted hundreds of thousands of dollars
 - Many people are competing for these resources
 - Your wasted run is likely to be either your or someone else's opportunity lost
 - You are likely charting new scientific territory
 - Some aspect of using your code may not have been important before, but may become critical in the new study
 - Some solver may run up against the limits of its validity
 - Inflight correction may be needed to parameters to continue with the study
- Aim for no surprises, but be prepared for them

Why do you need to plan?

- Machines are expensive and rare resources
 - Operating them is also very expensive
 - If you made a mistake in your input parameters and got garbage results on a large scale run, you just wasted hundreds of thousands of dollars
 - Many people are competing for these resources
 - Your wasted run is likely to be either your or someone else's opportunity lost
 - You are likely charting new scientific territory
 - Some aspect of using your code may not have been important before, but may become critical in the new study
 - Some solver may run up against the limits of its validity
 - Inflight correction may be needed to parameters to continue with the study
- Aim for no surprises, but be prepared for them

In the 2005 simulation mentioned earlier, out of 5 teams, ours was the only team that had success in getting a good science outcome

How do you plan

- Focused verification of the target simulation on the target platform
 - Over and above regular testing
 - Emphasis on understanding solver validity regime
- Pathfinder runs to get a good estimate of needed resources
 - Cost benefit analysis of fidelity vs reaching science goals in allocated resources

How do you plan

- Develop helpful diagnostics
 - Low overhead ways of confirming the health of the run
 - Are conserved quantities conserved?
 - Has any quantity become unphysical?
- Develop hierarchy of analysis
 - Full analysis of runs is not feasible in flight
 - Intermediate level analysis can give further insight into health of the simulation

Story of one simulation campaign

- Theory of Type Ia supernova explosion – 2006/2007
 - Evidence from observations:
 - Light curve powered by Ni56 decay
 - Evidence of medium weight elements, but in much smaller quantities
 - Implied transition from deflagration to detonation
- A 2D exploratory run had given a tantalizing answer to how?
 - To confirm a full 3D run was needed at good enough resolution
 - It would be the largest run of its kind at the time – totally uncharted territory
 - Until then 3D runs had been octants relying on symmetry
 - The 2D run had shown that symmetry had to be avoided

Preparation Steps

- Step 1 – develop a test that represents the most complex physics interactions
- Challenges:
 - Features take a long time to develop
 - Want to ensure that at least one refinement step occurs during the test
 - IO too slow to restart from a large checkpoint at late stage of the run
 - Also test would need a large chunk of the machine
- Use physics understanding to create initial conditions that would quickly develop comparable complexity

Preparation Steps

- Step 2 – Use the new test to characterize the performance behavior of the target platform
- Motivation:
 - Standard performance studies could not give crucial information
 - AMR refinement patterns make each application different
 - Interoperability and trade-off opportunities needed to be explored in a closely resembling simulation behavior
- Full fidelity 2D runs, and a set of runs of the new test provided enough information to extrapolate and estimate needed CPU hours

Preparation Steps

- Step 3 – Look for trade-offs and optimization opportunities
- Motivation:
 - Initial CPU estimates too high to complete the runs within allocations
 - Exploration of any parameter space needs to minimize individual run times
- Many opportunities were found, documented in reference below.
 - Identify redundant refinement and get rid of it
 - Coarsen computations for some physics
 - Move some computations to post-processing
- **All optimizations were based on scientific and numerical intuitions**

Dubey A, Calder AC, Daley C, et al. Pragmatic optimizations for better scientific utilization of large supercomputers. The International Journal of High Performance Computing Applications. 2013;27(3):360-373. doi:[10.1177/1094342012464404](https://doi.org/10.1177/1094342012464404)



Preparation Steps

- Step 4 – Prepare diagnostics and quick analysis mechanism
- Examples-- diagnostics
 - Conservation of mass, momentum energy
 - Changes in dt recorded in the logfiles
 - Spikes in variable values
- Examples – quick analysis
 - Quick visualization of random 2D slices
 - Inspection of critical quantities in 1D

Preparation Steps

- Step 4 – Prepare diagnostics and quick analysis mechanism
- Examples-- diagnostics
 - Conservation of mass, momentum energy
 - Changes in dt recorded in the logfiles
 - Spikes in variable values
- Examples – quick analysis
 - Quick visualization of random 2D slices
 - Inspection of critical quantities in 1D

Lab notebook artifacts

- every run registers all configurations and runtime parameters in a logfile.
- logfiles are cumulative
- dedicate space for storing all results in a preconfigured directory structure
- scripts to move output from scratch to the dedicated space

Outcome

- A successful campaign
 - But not without hitches
- Optimization related runs were not given the same level of care
 - Data was considered disposable
 - Code changes were documented

Outcome

- A successful campaign
 - But not without hitches
- Optimization related runs were not given the same level of care
 - Data was considered disposable
 - Code changes were documented
- For the paper the referee asked for details from optimizations
 - We did not have them
 - Fortunately the referee was satisfied with reasoning and other supporting evidence we produced

Summary and Takeaways

- Good science with computation is a craft -- training is needed in how to do it
- Machines are expensive to build and expensive to run
 - They provide opportunity for great work
 - Care is needed to ensure that the outcome meets expectations
- Reproducible results are a necessity, not a luxury
 - There is no credible science without provenance

"a parameter combination that induces erroneous results is easily selected"
- <https://doi.org/10.1063/1.476021>