Lab Notebooks and Managing Computational Experiments

Presented by

COLABS: Collaboration for Better Software for Science

Anshu Dubey (she/her)
Argonne National Laboratory

Software Practices for Reproducible Science tutorial @ ACM-REP 2024

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

In collaboration with

With prior support from

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is:** Anshu Dubey and Gregory R. Watson, Software Practices for Reproducible Science tutorial, in 2024 ACM Conference on Reproducibility and Replicability (ACM-REP), Rennes, France and online, 2024. DOI: 10.6084/m9.figshare.26019469**.**
- Individual modules may be cited as *Speaker, Module Title*, in *Tutorial Title*, …

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 supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Next-Generation Scientific Software Technologies (NGSST) program.
- 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.

A minimal definition of a lab notebook

A goal of keeping a lab notebook is "...to write with enough detail and clarity that another scientist could pick up the notebook at some time in the future, repeat the work based on the written descriptions, and make the same observations that were originally recorded. If this guideline is followed, even the original author will be able to understand the notes when looking back on them after considerable time has passed!”

- Howard Kanare, *Writing the Laboratory Notebook*

DIKUW

Data, Information, Knowledge, Understanding, Wisdom

A classification scheme that overloads everyday words so that we can use the same language and understand each other.

We will build on this to understand & appreciate documentation & lab notebooks.

DIKW pyramid (unaltered) – Wikipedia/Longlivetheux

Data & Information

- Data
  - Collection of numbers, symbols, text, *etc.*
  - It has value only because it was recorded and exists.
  - **Example**: Timeseries representation of temperature, relative humidity, and precipitation.
- Information
  - Facts gleaned from data.
  - Answers questions such as who, what, when, how much, how long, *etc*.
  - **Example**: Starting at 2pm the temperature dropped by 5ºF over 15 minutes. At 2:05 pm it started to rain and 0.25” of rain was accumulated over the next 30 minutes.

Knowledge & Understanding

- Knowledge
  - Derived from information, experience, and **understanding**.
  - **Example**: When relative humidity levels are high and the temperature drops substantially, there is an increased probability of precipitation.
- Understanding
  - A deep theoretical background in and practical experience with the system whose data was acquired and studied?
  - The ability to explain why?
  - “Understanding is a kind of ecstasy” – Carl Sagan
  - **Example**: A meteorologist could explain at different levels of detail how the atmosphere works to substantiate the knowledge.

Knowledge Management

- DIKUW is related to knowledge management
- We want to
  - **Recognize** when we generate knowledge
  - **Capture** and **preserve** that knowledge
  - **Communicate** that knowledge
- Scientific work can benefit from knowledge management

Is knowledge communication only about communicating to others?

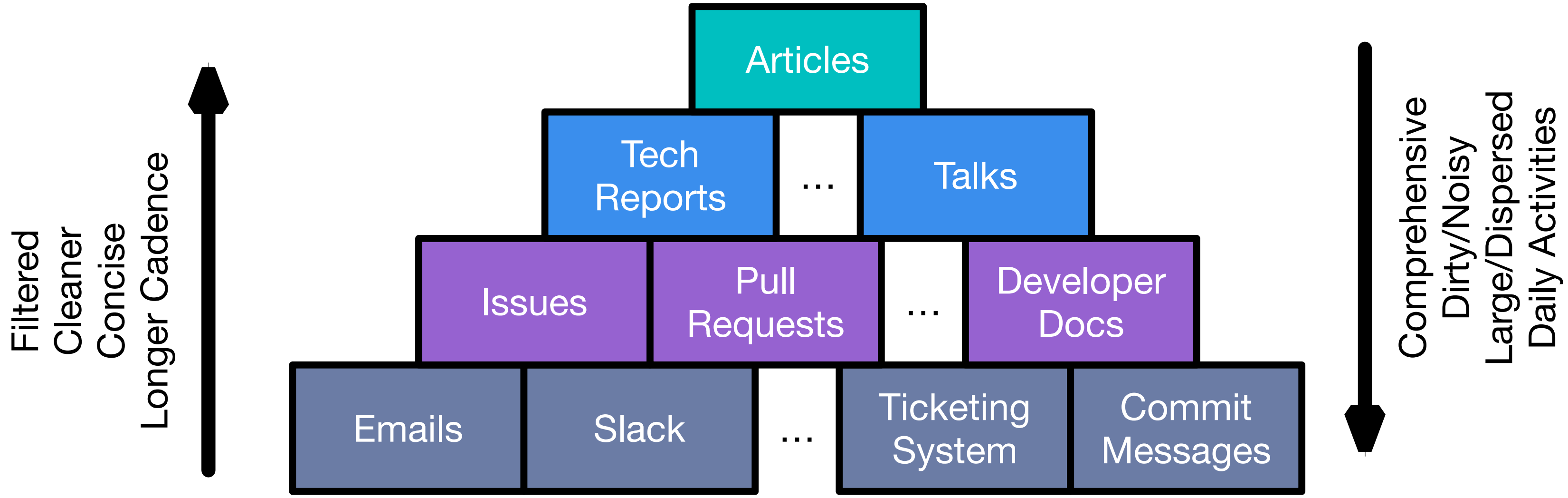
Example: Lessons learned

- After we live through an experience, we want to derive lessons learned
  - Experience is valuable, more so if we reflect and are thoughtful
  - Generate/capture knowledge to grow, improve, and avoid difficulties/mistakes
  - Hope to improve understanding
- We get more if we derive lessons learned together
  - Create more or higher quality knowledge
  - Communicate the knowledge implicitly

Should we communicate lessons learned to others?

Which leads us to documentation

Documentation is knowledge communication & can build understanding



- Data at the bottom. Knowledge as we move up?
- Some documents frozen at creation. Others are living.
- Does this capture how hard it is to do documentation in a distributed, digital world?

And finally, we reach our destination

Lab notebooks are

- A fundamental part of communication as well as rigorous, reproducible science in a lab,
- A common-place or required part of an experimental laboratory,
- Populating a scientific “lab notebook” was an “automated” process at the observatory,
- A tool for preventing scientific fraud, and
- Record of invention and defending against allegations of fraud.

Lab notebooks

- Should be used regularly,
- Should be comprehensive and never filtered,
- Don’t need perfect grammar and full sentences,
- Content is frozen at creation, and
- Hopefully contains more than just data (*e.g.*, motivation, reasoning, conclusions).

They aren’t good at communicating knowledge, **but**
people interact, evolve, and grow by collaborating through the notebook.

Example notebook entries

A bad example

<u>**Monday July 25, 2022**</u>
9:05 am - Do study ABC
8:47 pm - Lot’s of interesting data!
- Results are in GCE

A better example

<u>**Monday July 25, 2022**</u> (Jared)
9:05 am - Continuing work for study ABC. (See July 7)
- I presently believe that if A happens, then B must also happen.
- To verify this, I intend to
  - …
  - ***

9:30 am - Started executing this experiment on Bebop.
- Built debug version of binary with Intel 20.4
- Based on <u>clean</u> commit 5a43b21c
- Build log saved to `my_test_2022.log`
- No errors or warnings emitted
- Used job script `run_my_test` with configuration 24 (Job ID 123456)
- Stdout/err & results saved in folder ABC

10:07 am - Analysis run with script `analyze_my_test.py` and results saved in same folder.
- Since no peak seen around 1.5 MeV, I was wrong. But based on this, I now *believe* that if A happens, then C must also happen.

Conversations with Carlo

Carlo Graziani is a Computational Scientist at ANL
BSSw blog article *HPC and the Lab Manager*

- As researchers' careers progress
  - The problems become more complex and larger
  - Previous informal techniques for executing a study start to fail
  - The researchers' sense that something is missing
  - They invent processes and tools to compensate

This happened to Carlo and at some point, he realized that
"I had re-invented the lab notebook!"

Not all lab notebooks are alike

- Lab notebooks to record work done on instrument
- Lab notebooks to record acquisition of data
- Pull Request as **filtered** lab notebook – higher up the hierarchy
  - PR allows for additional content that's distinct from the individual commits
  - Flash-X PR #247 is example
  - Record process to verify correctness of changes
  - 2-2.5 days effort carried out over a week
  - Copy/pasted from previous PR and adapted first (designed process)
  - Improved as I carried out process – converging on a quasi-procedure
  - Filtered so that reviewers aren't overwhelmed
  - Helped me organize effort & design good tests
  - Senior reviewers provide feedback & suggest improvements
  - Junior reviewers exposed to work habits of other people

Example:
Flash-X
PR #247

jared321 commented on Aug 18 • edited

Author

Working on GCE/compute-012 with

```
Currently Loaded Modules:
  1) intel/20.4   2) mpich/3.4.2-intel   3) hdf5/1.12.1-mpich-3.4.2-parallel-fortran
```

- All tests will be built in debug mode using the Intel/20.4 + MPICH + HDF5 stack loaded via GCE modules.
- The same pseudo-UG-only testing was carried out identically for the 2D/Sedov/simpleUnsplit and 3D/Sedov/Unsplit simulations with Paramesh, AMReX, and Milhoja. These two simulations were chosen as they will hopefully become official Milhoja Comparison tests in the GCE test suite.
- All 2D tests ran in 304 steps with 6 checkpoint files including the initial conditions; 3D, 61 steps and 2 checkpoint files.
- Ran the 2D tests with 4 MPI processes; the 3D, with 8 MPI processes. These are the number of processes presently used for Sedov tests in the GCE test suite.
- Confirmed with `sfocu` that the Paramesh and AMReX final checkpoints are identical and that the Paramesh and Milhoja final checkpoints are identical.
- Confirmed that the integrated quantities were conserved in each case up to a reasonable level and that the initial values were consistent across all associated runs to at least 12 decimal digits.
  - standard deviation of 2D conserved quantities less than 2.5e-14
  - standard deviation of 3D conserved quantities less than 3.25e-14
  - Confirmed that the initial values of the [xyz]-momenta were all exactly zero and that the z-momenta was exactly zero for each all time steps in the 2D simulations.
- Confirmed in Milhoja 2D case that we get, as expected, the identical final checkpoints if we run with 4 and 8 MPI processes.
- Where possible, the `milhoja.log` files were reviewed to confirm correct configuration and execution.
- Viewed the Milhoja final checkpoints in Visit to qualitatively confirm reasonable results for all variables and correct mesh overlays. 3D data was viewed with a slice at 50%.

No one likes writing lab notes…

We love to consume documentation; write it, not so much.

Optimistic

- Lack of experience
- Lack of training
- Lack of appreciation
- Lack of incentives

Cynical

- We want and appreciate when others share knowledge with us.
- We don’t want to take the time to capture, preserve, and communicate knowledge we generate.

One aspect of productivity

One person decreases their short-term efficiency so that many (and the team) achieve long-term efficiency and quality.

Nothing beats good ol' pen and paper

"Since at least the 1990s, articles on technology have predicted the imminent, widespread adoption of electronic laboratory notebooks (ELNs) by researchers. It has yet to happen — but more and more scientists are taking the plunge."

- Roberta Kwok, *How to pick an electronic laboratory notebook*, Nature

Pen & Paper Pros

- Most can use paper and pen in any situation
- Open format can allow for creativity and easier annotation
- Concentrate on the work rather than tooling
- Good if notetaking slows down progress
- Notebook is stored **publicly** next to where it is used

Electronic Woes

- Tied to technology that could fail
- Overwhelming variety of possible solutions with different pros and cons
- Uncertainty about future of tool, increased costs, inability to export
- Does funding restrict where and how digital notes can be stored?

Criteria for lab notebooks for computing?

- **Paper won't work.** We work anywhere and sometimes in distributed way.
- Should notebooks be public and how to do that?
- How many different types of notebooks do we need?
- Do we use a single ELN or distribute notes across a suite of tools?
- How can we use automation appropriately to overcome difficulties and increase productivity?

We likely need many streams of lab notes

Different streams of lab notes

- Lab notebook for changes to scientific 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
  - We don’t want a single 10,000-line README

Git lab notes stream

Keep lab notes for your software as close to the “instrument” as possible

```
Date:   Mon Jun 27 10:42:22 2022 -0500

    (Issue #215) Added in Milhoja Init unit test.  I have tried to structure this in
    accord with the unitTest architecture in the User Manual.  One main consequence
    of this is that it uses the generic Grid unitTest evolveAll routine, which
    writes a unitTest output file.  Since this test also uses the ut_testDriverMod
    framework, I had to simplify that so that it doesn't attempt to write the same
    file.  This change will likely affect the AMReX unit tests.

    I have run this successfully in 1D, 2D, and 3D on GCE/compute-12 with Intel.
    The test correctly creates a unitTest_XXXX output file and adds in a success
    line only if the test was 100% successful.  I temporarily dumped the ICs and
    final solution to AMReX-format files and manually confirmed correct content.
```

*This message is missing a title as the first line.

Details not obvious from commit diff:
Motivation, reasoning, consequences

Testing notes

Use Pull Request to capture final verification streams

README lab notes streams

- High-level road maps with motivation, documented decisions, and conclusions
- Concise living docs that function as executive summaries
  - Higher up in documentation hierarchy
- Low-level notes such as managing software stack
  - Traceability of SW environment & therefore verification
  - Can be turned into procedures

High-level README

My Study Executive Summary

This repository encapsulates the computational laboratory environment constructed and used for our 2022 My Study scientific research. Not only does it contain all tools needed to setup and use the environment, but it also contains the metadata and context needed to understand how data was acquired and how to use it for analysis and drawing conclusions.

Study Goals

Our motivation for this study is ...

We believe that ...

We intend to use our software to ...

To accomplish our goals we require that ...

We understand that our software is limited and therefore that our study is limited in the sense that ...

Some optional goals are ...

Ideally, we would like to publish our results in the Journal ZYX with a submission target date of ...

Organization

ABC will be the PI and will ...

XYZ is responsible for ...

Low-level README

AMReX Installation History

5 March 2020 - FlashFluxRegister b1dd083

- Using changes from FlashFluxRegister branch by Weiqun.
- Need for improved flux handling across all branches.
- Built 1D/2D/3D libraries for both gfortran and Intel at this commit.
- These were tested with the Staged/Staged-Intel/Master compute00x test suites with no other changes made. All tests passed.

6 November 2019 – master – 93fb085d

- Austin encountered an AMRex based bug and reported it in Issue #643. He reports that AMRex fixed that bug at commit - 23f943f.
- New install location is /nfs/proj-flash5/ a shared group at MCS, we are no longer going to use the /sandbox/flash/ versions on compute00[123]
- Built 1D/2D/3D libraries for both gfortran and Intel at this commit.
- These were tested with the Staged/Staged-Intel/Master compute00x test suites with no other changes made. All tests passed.

29 March 2019 – development – 06c6c0e2

- Rebuilt the libraries from a commit that contains the multifab changes implemented manually in the previous libraries. These changes were merged in to AMReX at commit 66392f8d on Tue Mar 26. No code in AMReX needed modifying for this build.
- These libraries were also built with Particles and linear solvers enabled for Saurabh's tests. He showed me that I had to update the automagically updated makefile to get the Particles Fortran interface into the libraries.
- Built 1D/2D/3D libraries for both gfortran and Intel at this commit.
- These were tested with the Staged/Staged-Intel/Master compute00x test suites with no other changes made. All tests passed.

20 March 2019 – master – 884d3194 (modified)

- Updated the files AMReX_multifab_fi.cpp and AMReX_multifab_mod.F90 at commit to incorporate the local tile index in the Fortran interface. These changes had already been tested during development in local repositories.
- Built 1D/2D/3D libraries with both gfortran and intel on compute001. These were then copied over to compute002 and compute003.

Capturing data context & metadata

- Automate as much as possible
- Build dates, user, system name, git hashes, configuration data in file headers
  - Self-documenting files
- Build & job logs
  - Software environment info (e.g., modules, `ldd` output)
  - git diffs
  - Environment variables

Jupyter notebooks

The exception to the rule?

- 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
- Where do notebooks fit into the documentation hierarchy?
- Repetitive use of notebook?
  - Limit amount of code in notebook

How to organize your “virtual” (multi-stream) lab notebook?

- It should be
  - Easy to create and maintain lab notes
  - Easy to concentrate more on executing work & less on documenting it
  - Easy to find what you need
- Each stream should
  - Have a clear identity for what it records
  - Not contain notes contained in other streams
  - Be recorded by using the right tool for the job

This implies the need for a documentation scheme.

I design my lab notebooks into a larger execution environment.

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
  - “Store on a shelf” for later

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?

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.

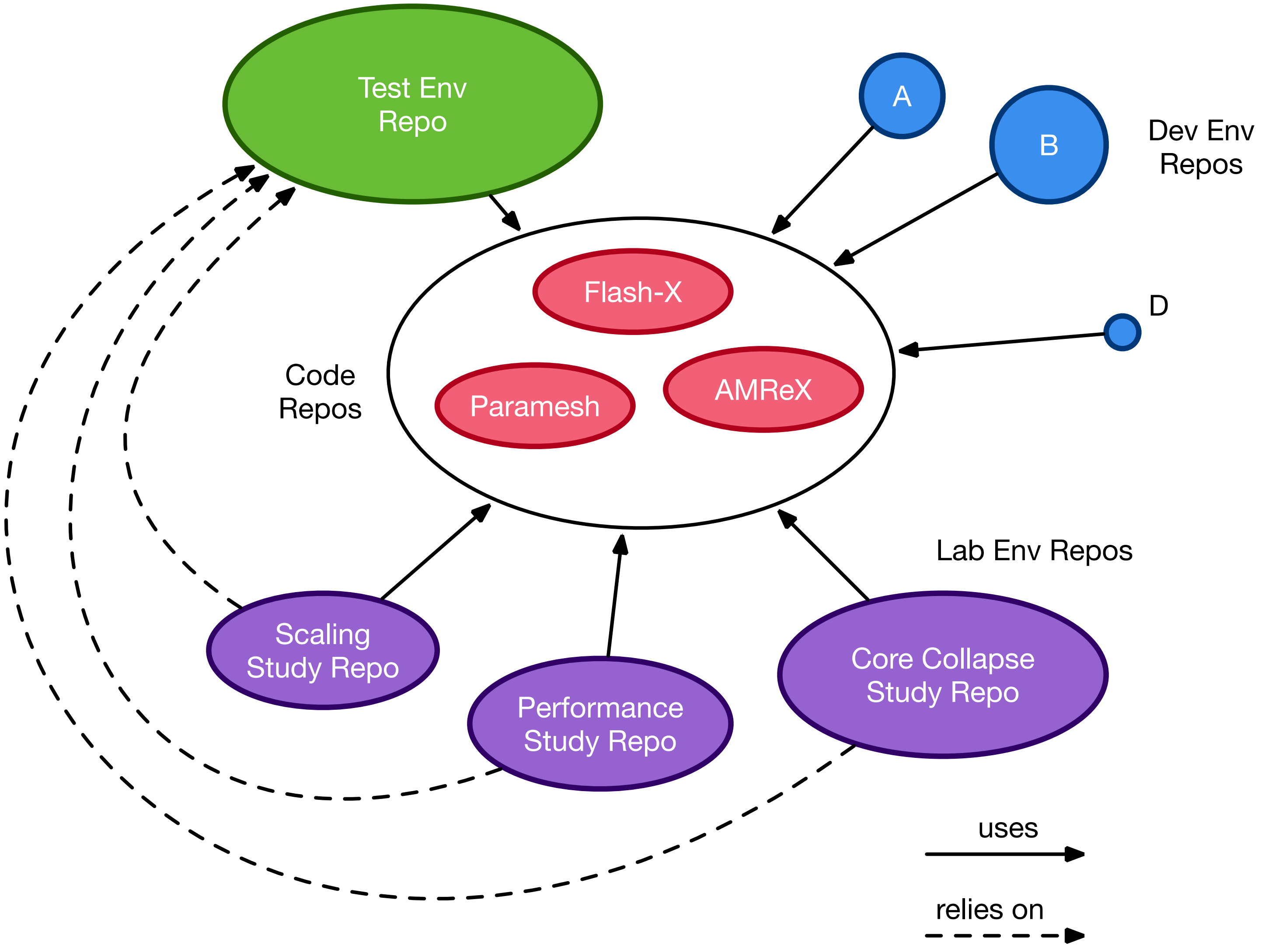
Computational laboratory environments

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 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

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