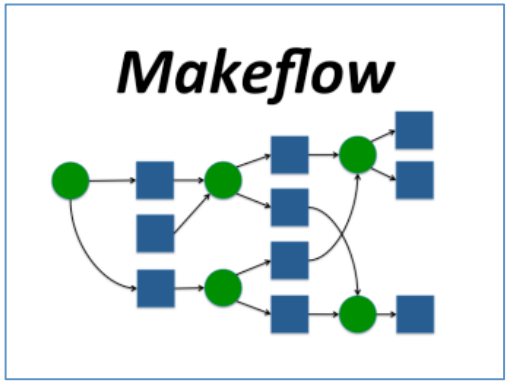
Combining Containers and Workflow Systems for Reproducible Execution

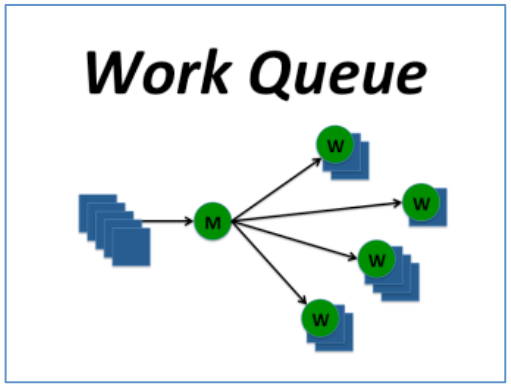
Douglas Thain, Alexander Vyushkov,
Haiyan Meng, Peter Ivie, and Charles Zheng

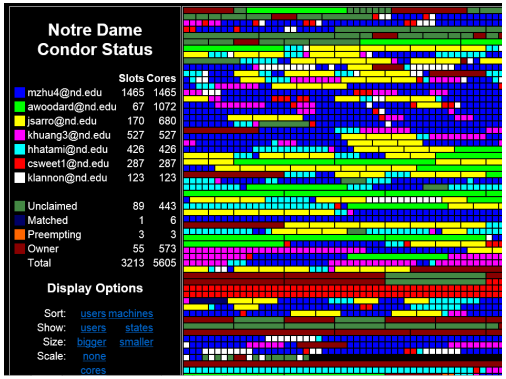
University of Notre Dame

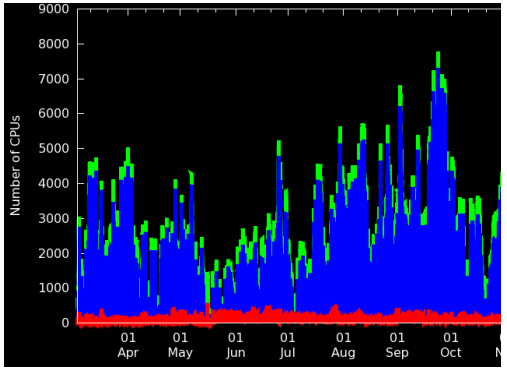
UNIVERSITY OF NOTRE DAME

The Cooperative Computing Lab









http://ccl.cse.nd.edu

The Cooperative Computing Lab

- We *collaborate with people* who have large scale computing problems in science, engineering, and other fields.
- We *operate computer systems* on the O(10,000) cores: clusters, clouds, grids.
- We *conduct computer science* research in the context of real people and problems.
- We *release open source software* for large scale distributed computing.

http://ccl.cse.nd.edu

Condor Matrix Display

condor.cse.nd.edu/condor_matrix.cgi

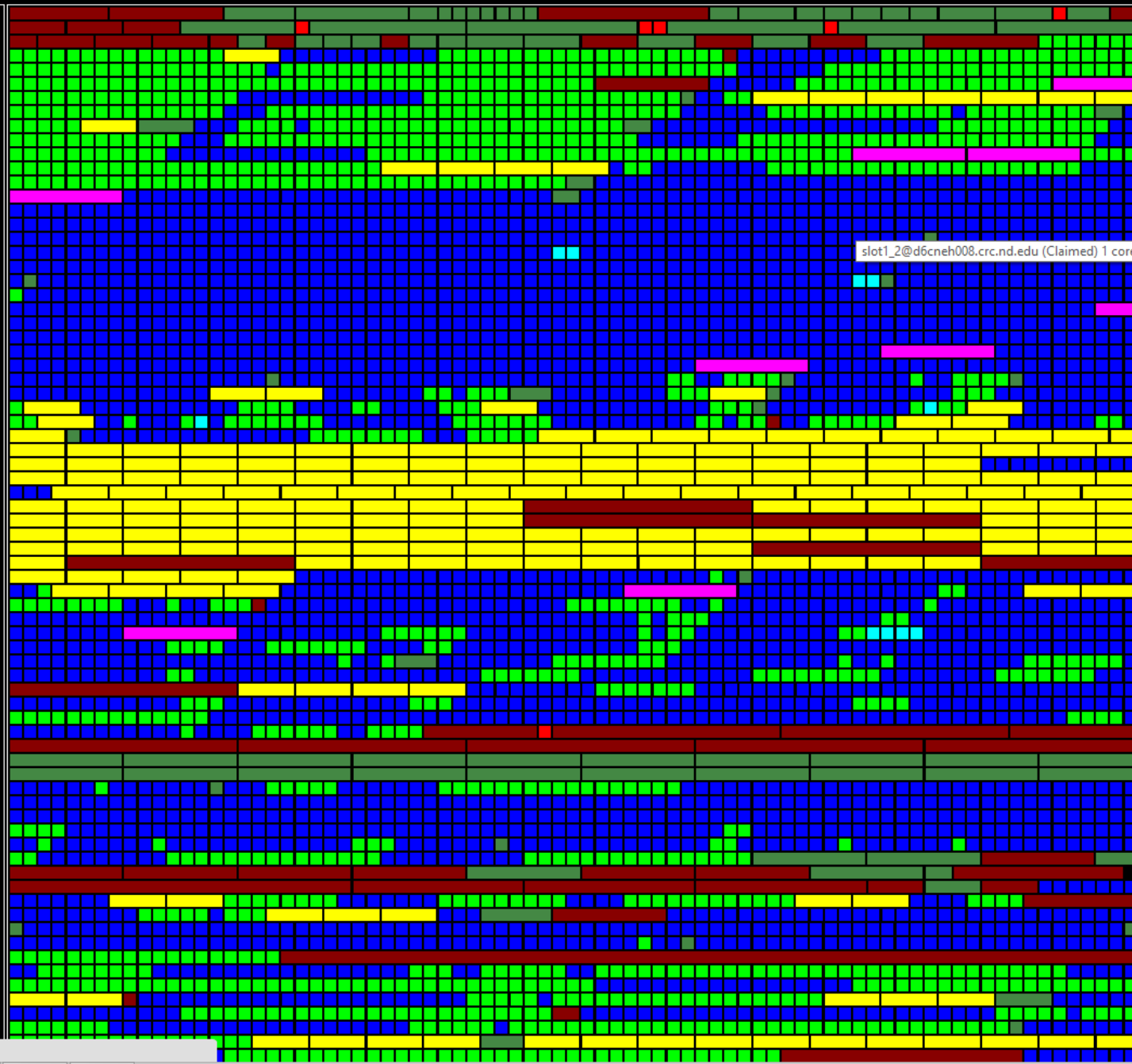
Notre Dame Condor Status

	Slots	Cores
csweet1@nd.edu	4153	4153
mzhu4@nd.edu	2878	2878
jsarro@nd.edu	325	1300
ksmith37@nd.edu	18	144
apaul2@nd.edu	12	12
pivie@nd.edu	12	12
Unclaimed	114	504
Matched		
Preempting		
Owner	76	765
Total	7588	9768

Display Options

Sort: users machines
Show: users states
Size: bigger smaller
Scale: none
cores
memory

http://condor.cse.nd.edu



condor.cse.nd.edu/condor_matrix.cgi#

www.daspos.org

DASPOS Data and Software Preservation for Open Science

ABOUT | PEOPLE | WORKSHOPS | RESEARCH | REPORTS

The massive data sets accumulated by High Energy Physics (HEP) experiments represent the most direct result of the often decades-long process of construction, commissioning and data aquisition that characterize this science. Many of these data are unique and represent an irreplaceable resource for potential future studies. Forward-thinking efforts for preservation are necessary now in order to achieve the relevant parameters, analysis paths and software to preserve the usefulness of these rich and varied data sets.

"Ten or 20 years ago we might have been able to repeat an experiment. They were simpler, cheaper and on a smaller scale. Today that is not the case. So if we need to re-evaluate the data we collect to test a new theory, or adjust it to a new development, we are going to have to be able to resuse it. That means we are going to need to save it as open data..."

Rolf-Dieter Heur 2008
Director General, CERN

First Workshop Scheduled

The first DASPOS Workshop has been scheduled for Thursday - Friday, March 21-22, 2013, at CERN. More information

Data and Software Preservation for Open Science, DASPOS, represents an initial exploration of the key technical problems that must be solved to provide appropriate data, software and algorithmic preservation for HEP, including the contexts necessary to understand, trust and reuse the data. While the archiving of HEP data may require some HEP-specific technical solutions, DASPOS will create a template for preservation that will be useful across many different disciplines, leading to a broad, coordinated effort.

Discovery and Coordination

Series of highly-structured public workshops to define, discuss and document the details of data and software preservation

Prototyping and Experimentation

Key areas of reserach: data and query models and software sustainability models

The DASPOS Team

Computer science experts, experienced digital librarians, and experts in data-intensive fields, such as physics, astrophysics and bioinformatics

Workshop 1

2012-12-17 19:11:04

WORKSHOP 1 Establishment of Use Cases for Archived Data and Software in HEP Date: Thursday-Friday...

Workshop 2

2012-12-17 19:11:04

WORKSHOP 2 Survey of Commonality with other Disciplines Attendees: Broad participation from many...

Reproducibility is the cornerstone of the scientific method.

Can we really claim to be conducting science?

Reproducibility in e-Science is absolutely terrible today!

- Can I re-run a result from a colleague from five years ago successful, and obtain the same result? How about a student in my lab?
- Today, are we preparing for our current results to be re-used by others five years from now?
- Multiple reasons why not:
  - Rapid technological change.
  - No archival of artifacts.
  - Many implicit dependencies.
  - Lack of backwards compatibility.
  - Lack of social incentives.

Many different Rs…

- **Repeat** precisely what someone else did on the same resources, with the same techniques.
- **Reproduce** an equivalent computation on different resources, with similar techniques.
- **Repurpose** an experiment by running it again with a slight change to the data, software, or environment.
- **Reuse** the same artifact across many different experiments, for a longitudinal comparison.
- **Rely** on one party to set up an environment and make it usable for multiple parties. (Think sysadmins.)
- Other Rs?

Typical Computing Experiment

- PI gives student some general directions. Student writes some code, does some experiments, saves the outputs, writes the paper.
- Source code is often carefully curated. But what about the operating system, the software dependencies, the experimental configuration, the input data, etc…
- **Preservation** is necessary but insufficient. We must also be able to **reconstitute** the result from the preserved components.
- If we did manage to re-run everything, can we verify equivalence?

Preserve the Mess or Encourage Cleanliness?

- Preserve the Mess:
  - Let the end user do **whatever they want**, and then preserve the artifacts actually used.
  - Least user burden, but ingredients, once mixed, are hard to separate.
- Encourage Cleanliness:
  - Require the user to **preserve items in advance**, and then combine them in precise ways.
  - Higher user burden, but better captures intent and distinguishes between components.

Two Examples of Encouraging Cleanliness:

Umbrella and Prune

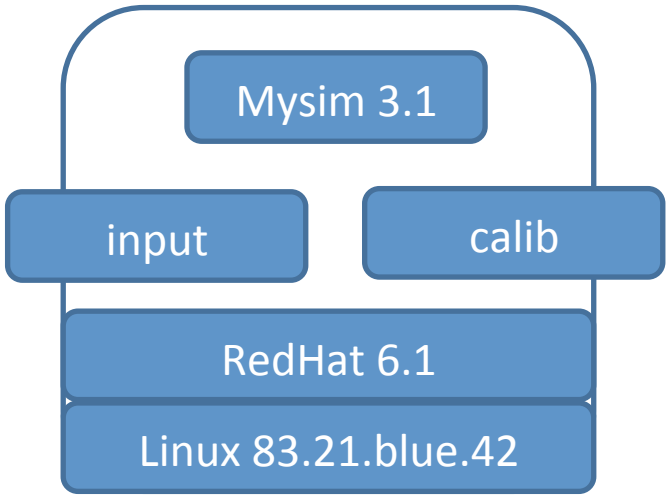
(research prototypes)

myenv1.json

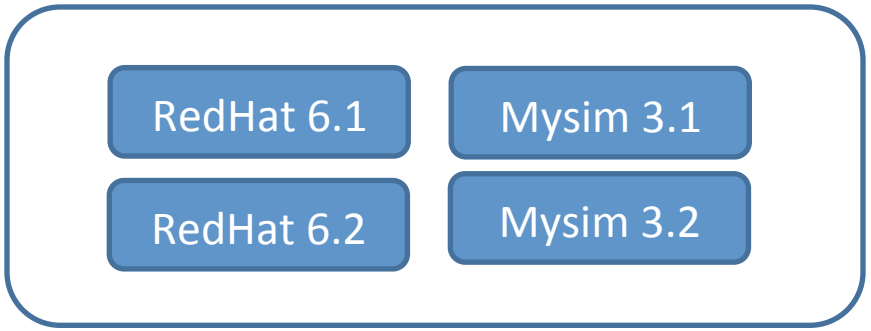
```
hardware = {
        arch = “i386”
        memory = 16GB; }
kernel = {
        name = “Linux”;
        version = “83.21.blue.42” }
opsys = {
        name = “RedHat”;
        version = “6.1” }
software = {
        simulator = {
                mount = “/soft/sim”;
                name = “mysim-3.1”; }
data = {
        input = {
                mount = “/data/input”;
                url = “http://some.url”; }
        calib =  {
                mount = “/data/calib”;
                url = “http://other.url”;
                checksum = “xyz”;  }
}
```

Umbrella

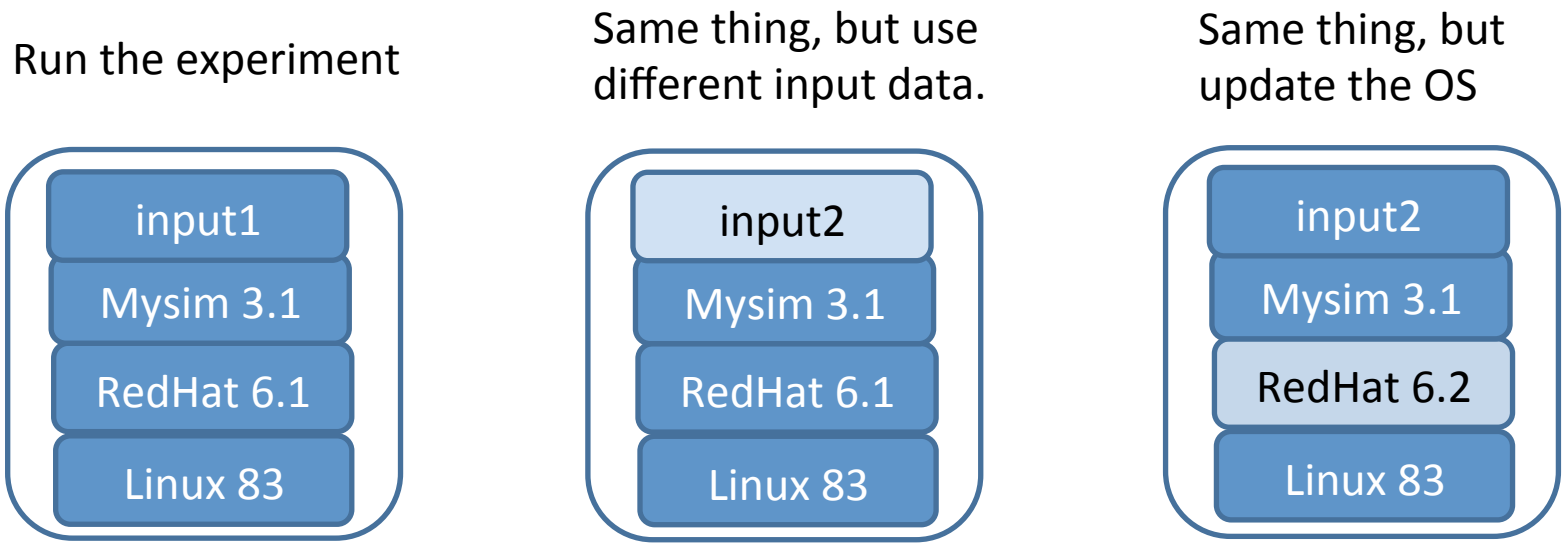
umbrella run myenv1.json



Online Data Archives



Umbrella specifies a reproducible environment while avoiding duplication and enabling precise adjustments.



Online Data Archive

RedHat 6.1	input1	input2	Linux 83	Mysim 3.1
RedHat 6.2	calib1	calib2	Linux 84	Mysim 3.2

Specification is More Important Than Mechanism

- Current version of Umbrella can work with:
  - Docker – create container, mount volumes.
  - Parrot – Download tarballs, mount at runtime.
  - Amazon – allocate VM, copy and unpack tarballs.
  - Condor – Request compatible machine.
- More ways will be possible in the future as technologies come and go.
- Key requirement: **Efficient runtime composition**, rather than procedural construction.

Example Umbrella Apps

- Povray ray-tracing application

http://dx.doi.org/doi:10.7274/R0BZ63ZT

- OpenMalaria simulation

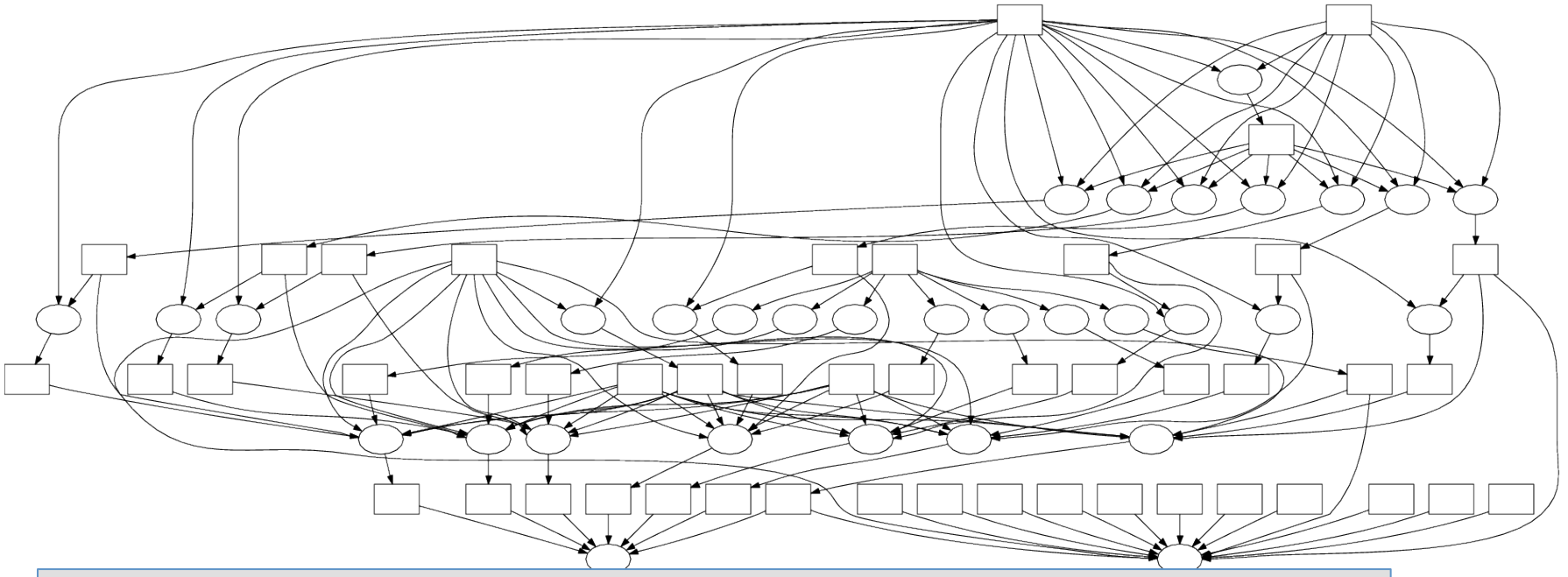
http://dx.doi.org/doi:10.7274/R03F4MH3

- CMS high energy physics simulation

http://dx.doi.org/doi:10.7274/R0765C7T

P.S. DOIs are almost (but not quite) the right solution for citing an executable object.

But how do we apply this to complex scientific workflows?



Is every single task a container? **No!**
Each task must be **placed** into a container so that we can use a common image for 1000s of tasks.

PRUNE – Preservation Run Environment

- Observation: The shell user interface does not accurately describe the environment or dependencies needed by a given task:

  mysim.exe -i input.txt –o output.dat

- Idea: Replace the traditional command line with an interface more like function invocation:

  output = mysim(input, calib) ENV myenv.json

- Build on ideas from GridDB, VDL, Swift, Taverna, Galaxy, but here focus is on precise reproduction and sharing with others.

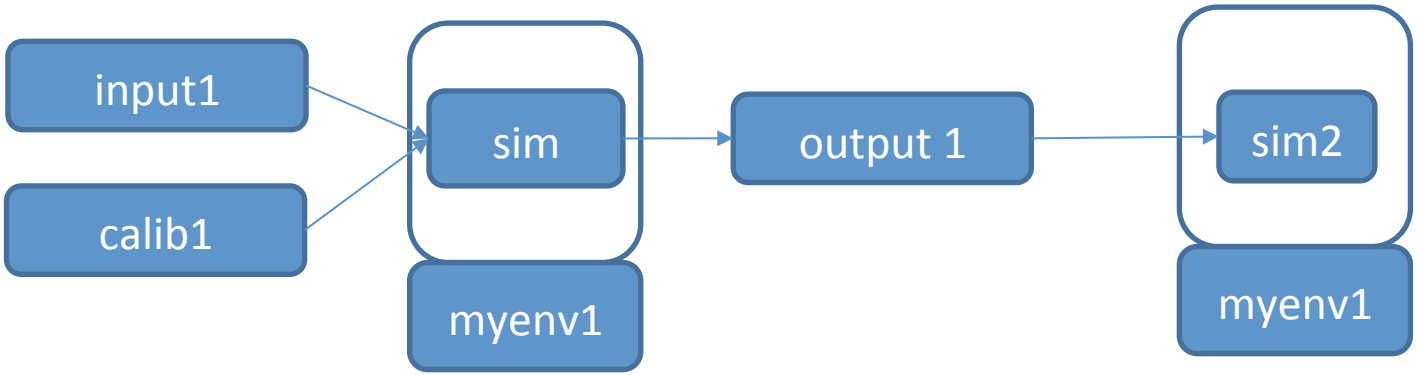
PRUNE – Preservation Run Environment

PUT “/tmp/input1.dat” AS “input1” [gets id 3ba8c2]
PUT “/tmp/input2.dat” AS “input2” [gets id dab209]
PUT “/tmp/calib.dat” AS “calib” [gets id 64c2fa]
PUT “sim.function” AS “sim” [gets id fffda7]

out1 = sim(input1, calib) IN ENV myenv1.json
[out1 is bab598]
out2 = sim(input1, calib) IN ENV myenv2.json
[out2 is 392caf]
out3 = sim(input2, calib) IN ENV myenv2.json
[out3 is 232768]

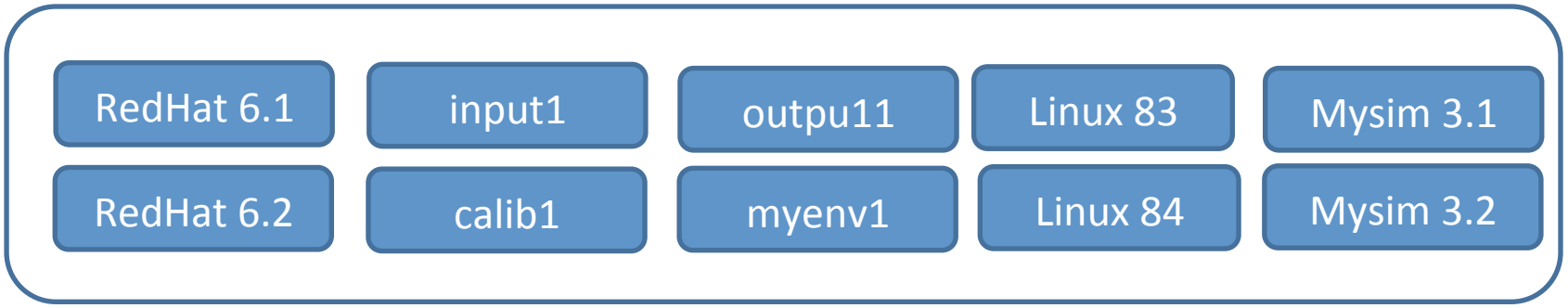
PRUNE connects together precisely reproducible executions and gives each item a unique identifier

output1 = sim(input1, calib1) IN ENV myenv1.json



Bab598 = fffda7 (3ba8c2, 64c2fa) IN ENV c8c832

Online Data Archive



All Sorts of Open Problems

- Naming: Tension between usability and durability. At least two levels of naming.
- What is the intersection of version control (store deltas) and provenance (store ops) ?
- Usability: Can we accommodate existing work patterns, or do we force new habits?
- Repositories: Who will run them, how many should we have, what will they cost…?
- Compatibility: Can we work in existing workflow technologies without starting over?
- Composition: MPI, BoT, Workflows, Map-Reduce, …

Ruminations

- Important to distinguish between the **environment** that is expected and the technology used to deliver it.
- Scientific users are accustomed to an **implicit** environment (laptop, hpc center) and we need to train them to be explicit about needs.
- Best practice: Start with empty environment and only include what is **explicitly** imported. (Golang corollary: Do not import what is not used.)
- Portability and preservation are two sides of the same coin: specification needed to run at scale is also the spec needed to preserve for the long term!

Acknowledgements

Haiyan Meng
hmeng@nd.edu
(Umbrella)

Alex Vyushkov
avyushko@nd.edu

Peter Ivie
pivie@nd.edu
(PRUNE)

Umbrella Technology Preview:
http://ccl.cse.nd.edu/software/umbrella

This work was supported by NSF grant PHY-1247316:
Data and Software Preservation for Open Science.