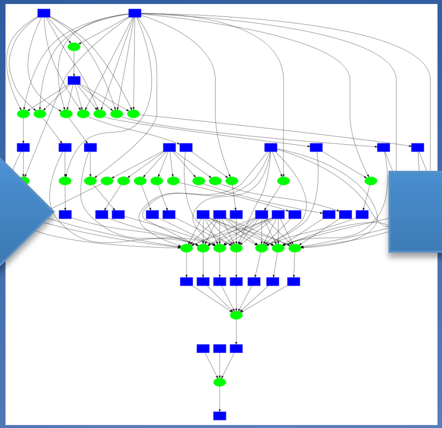


Challenges in Delivering and Deploying Software at Scale in Large Clusters

Douglas Thain and Kyle Sweeney
University of Notre Dame

{dthain|ksweene3}@nd.edu

Software Deployment on HPC

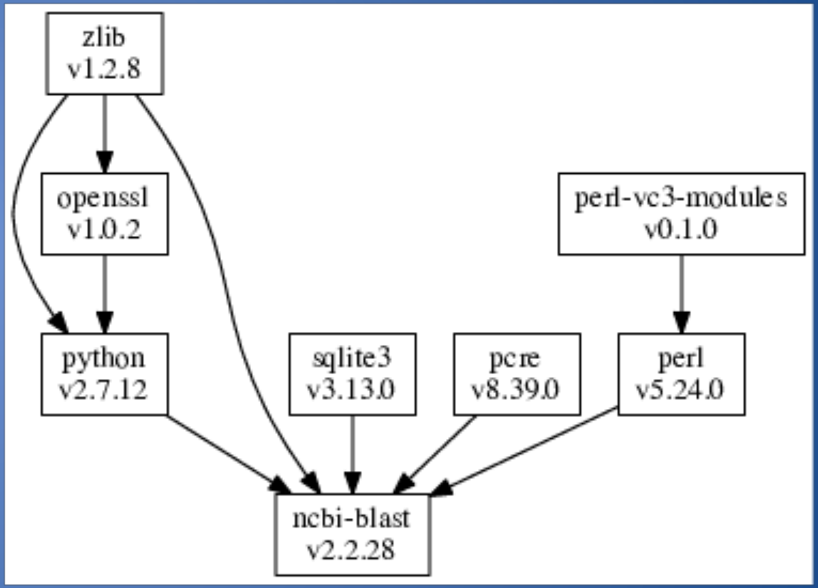
- Classic Approach
  - Single process MPI app created by end user.
  - Sysadmin installs, tests, proves the application.
  - Adjust to exploit local libraries / capabilities.
  - Application satisfied with a single site.
- Evolving Approach
  - Complex stacks of commodity software.
  - Developer is not the user!
  - Installed by end user just in time.
  - Users migrate quickly between sites.

Problem: Software Deployment

- Getting software installed on a new site is a big pain! The user (probably) knows the top level package, but doesn't know:
  - How they set up the package (sometime last year)
  - Dependencies of the top-level package.
  - Which packages are system default vs optional
  - How to import the package into their environment via PATH, LD_LIBRARY_PATH, etc.
- Many scientific codes are not distributed via rpm, yum, pkg, etc. (and user isn't root)

Typical User Dialog Installing BLAST

"I just need BLAST."
"Oh wait, I need Python!"
"Sorry, Python 2.7.12"
"Python requires SSL?"
"What on earth is pcre?"
"I give up!"



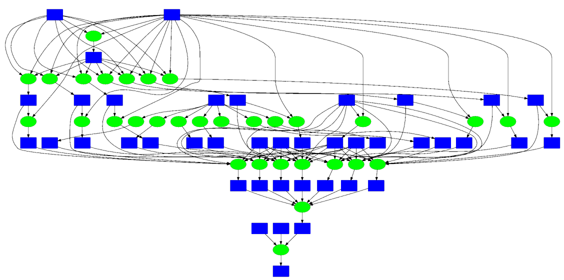
VC3: Virtual Clusters for Community Computation

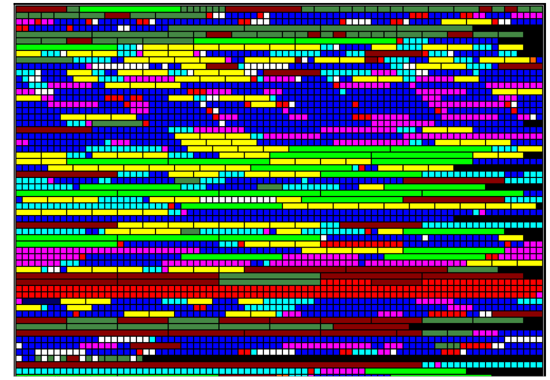
Douglas Thain, University of Notre Dame
Rob Gardner, University of Chicago
John Hover, Brookhaven National Lab

http://virtualclusters.org

Lincoln Bryant, Jeremy Van, Benedikt Riedel, Robert Gardner, Jose Caballero, John Hover, Ben Tovar, and Douglas Thain, **VC3: A Virtual Cluster Service for Community Computation,** *PEARC* 2018. DOI: 10.1145/3219104.3219125

You have developed a large scale workload which runs successfully at a University cluster.





Now, you want to migrate and expand that application to national-scale infrastructure.
(And allow others to easily access and run similar workloads.)

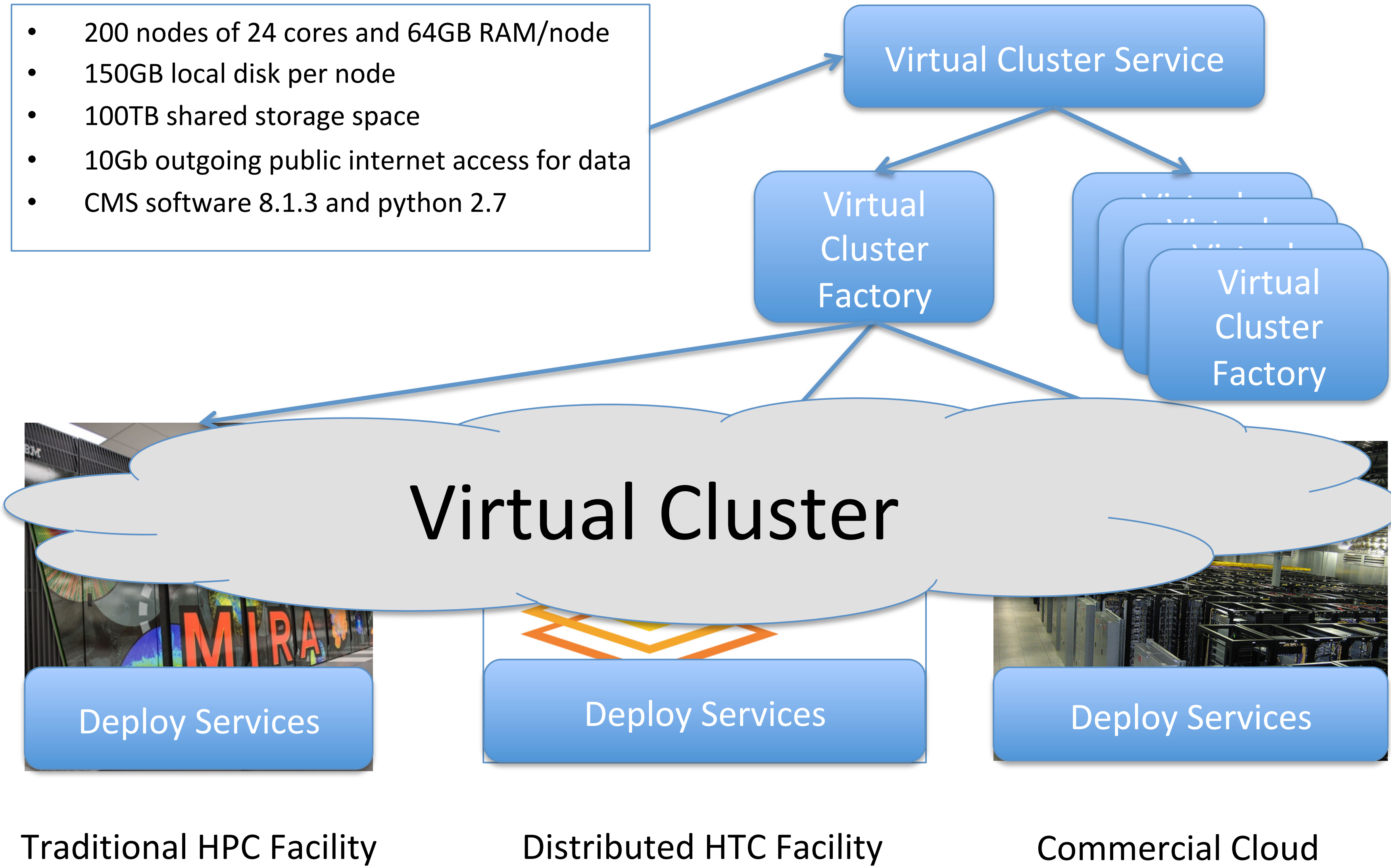
Traditional HPC Facility

Open Science Grid

Distributed HTC Facility

Commercial Cloud

Concept: Virtual Cluster



How do we get complex software delivered and deployed to diverse computing resources?

(without bothering sysadmins)

Delivery vs Deployment

- Delivery: Articulating and installing all of the necessary components at one site.

- Deployment: Moving all of the necessary components to each individual cluster node in an efficient manner.

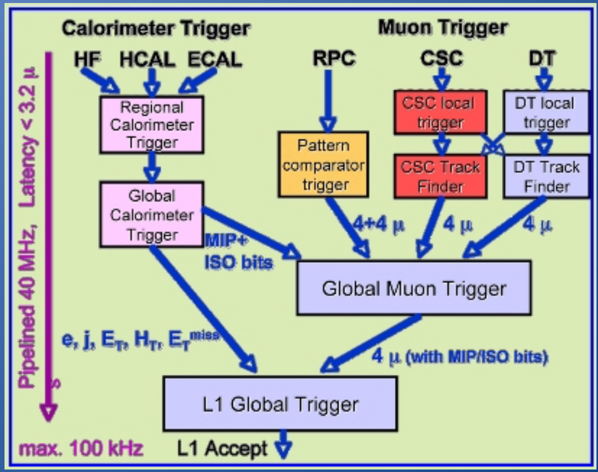
Example: CMS Analysis Software

Large Hadron Collider

Compact Muon Solenoid

100 GB/s

Online Trigger



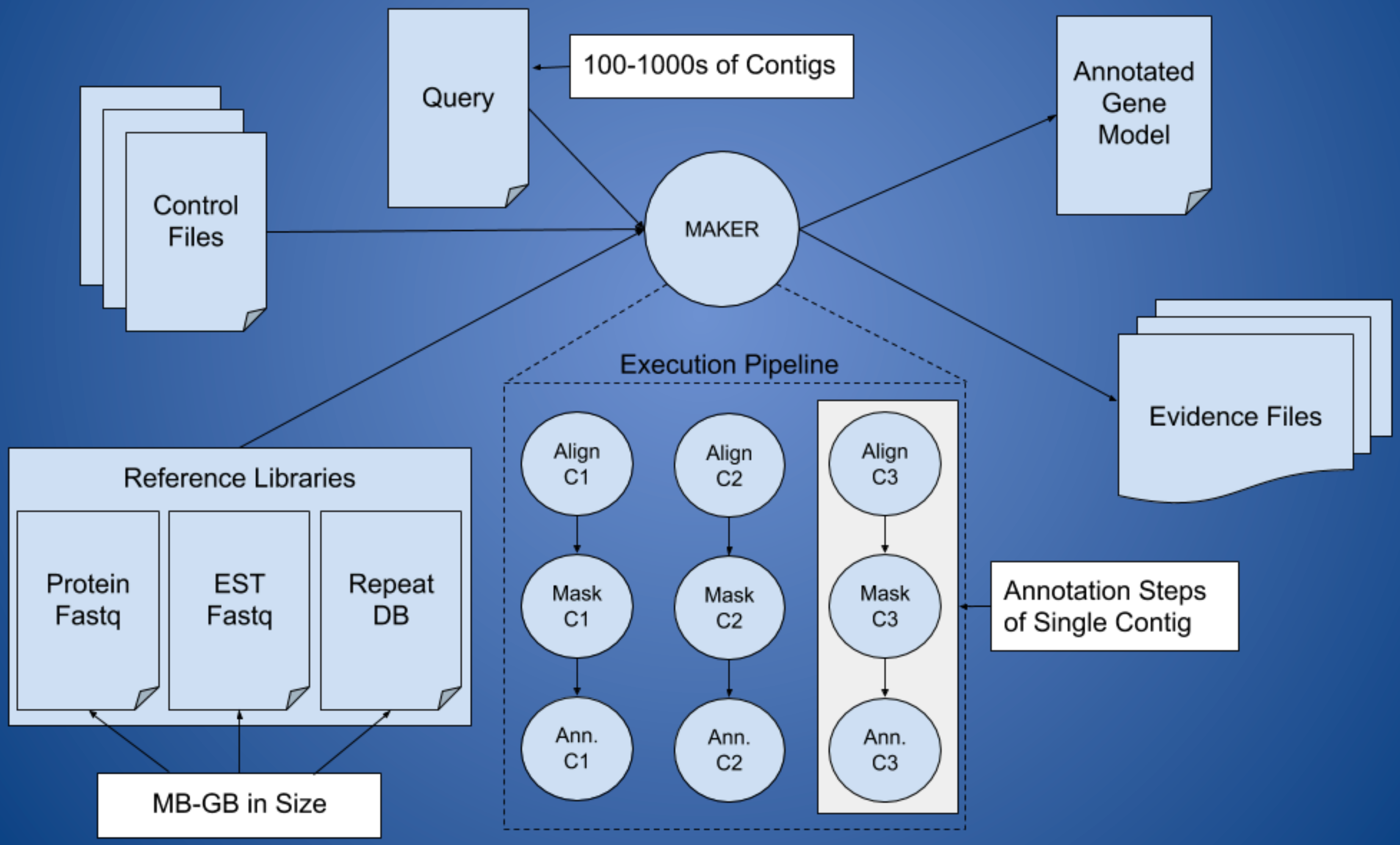
Many PB
Per year

Worldwide LHC Computing Grid

Example: CMS Analysis Software

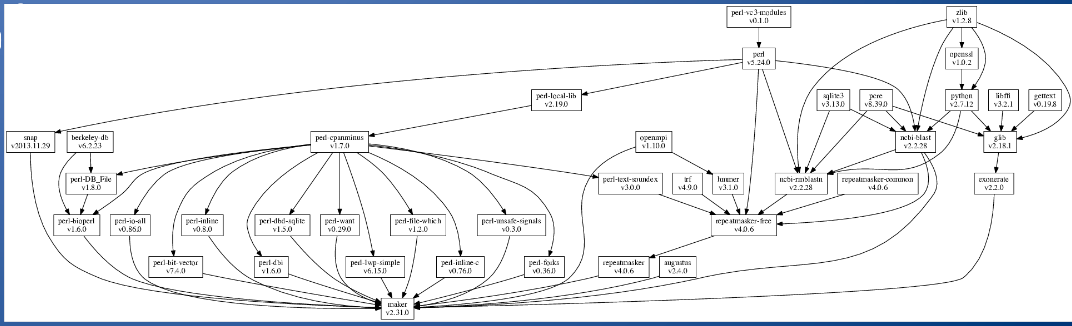
- Developed over the course of decades by 1000s of contributors with different expertise.
- Core codes in F77/F90/C99/C++18 + shell scripts, perl and python, scripts, shared libraries, config files, DSLs…
- Centrally curated by experts at CERN for consistency, reproducibility, etc.
- One release: 975GB, 31.4M files, 3570 dirs.
- Releases are very frequent!

Example: MAKER Genome Pipeline



Example: MAKER Genome Pipeline

- Large number of software dependencies (OpenMPI, Perl 5, Python 2.7, RepeatMasker, BLAST, several Perl modules)
- Composed of many sub-programs written in different languages (Perl, Python, C/C++)
- 21,918 files in 1,757 directories
- Typical installation model:
  Ask author for help!



Software Deployment/Delivery

- **Filesystem Methods**
  - Big Bucket of Software!
  - MetaFS: Metadata Acceleration
  - CVMFS: A Global Filesystem
- Packaging Methods
  - VC3-Builder: Automated Package Installation
  - Builder + Workflows
- Container Methods
  - Container Technologies
  - Containers + Workflows

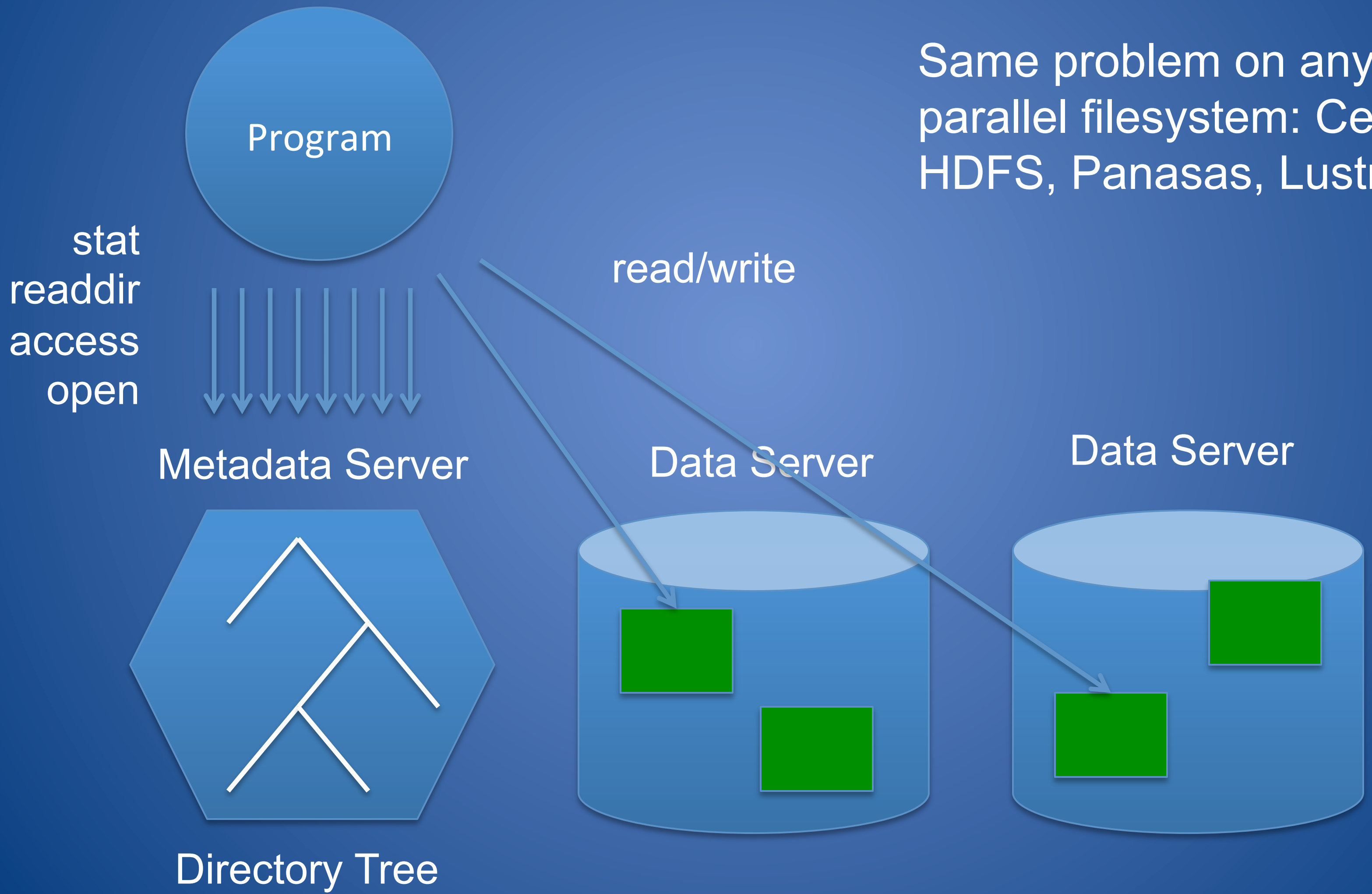
Big Bucket of Software!

- Collect everything – binaries, interpreters, libraries – into one big tarball.
- Delivery is easy: copy, unpack, setenv.
  - (Not all software can be relocated to a new path)
- User-compatible approach – no sysadmin support needed, occupies user storage, etc.
- Just set up batch jobs to refer to the deployed location, set PATH, and go.

But: Metadata Storms!

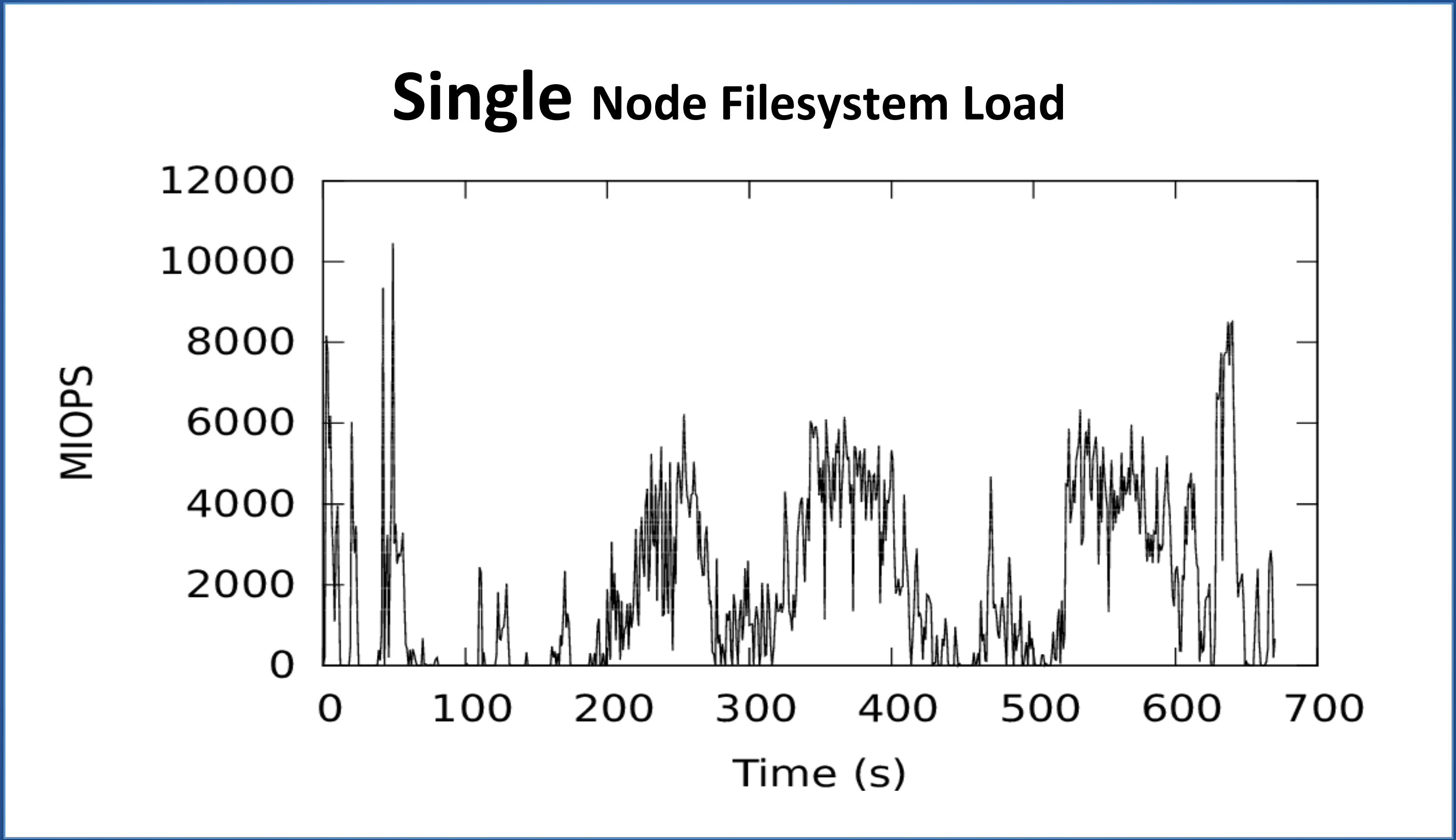
- Common behavior: long burst of metadata access at the beginning of an application:
  - Search through PATH for executables.
  - Search through LD_LIBRARY_PATH for libraries.
  - Load Java classes from CLASSPATH.
  - Load extensions from file system.
  - Bash script? Repeat for every single line!
- Complex program startup can result in millions of metadata transactions!

Metadata Storm



Same problem on any parallel filesystem: Ceph, HDFS, Panasas, Lustre, …

MAKER Metadata Storm

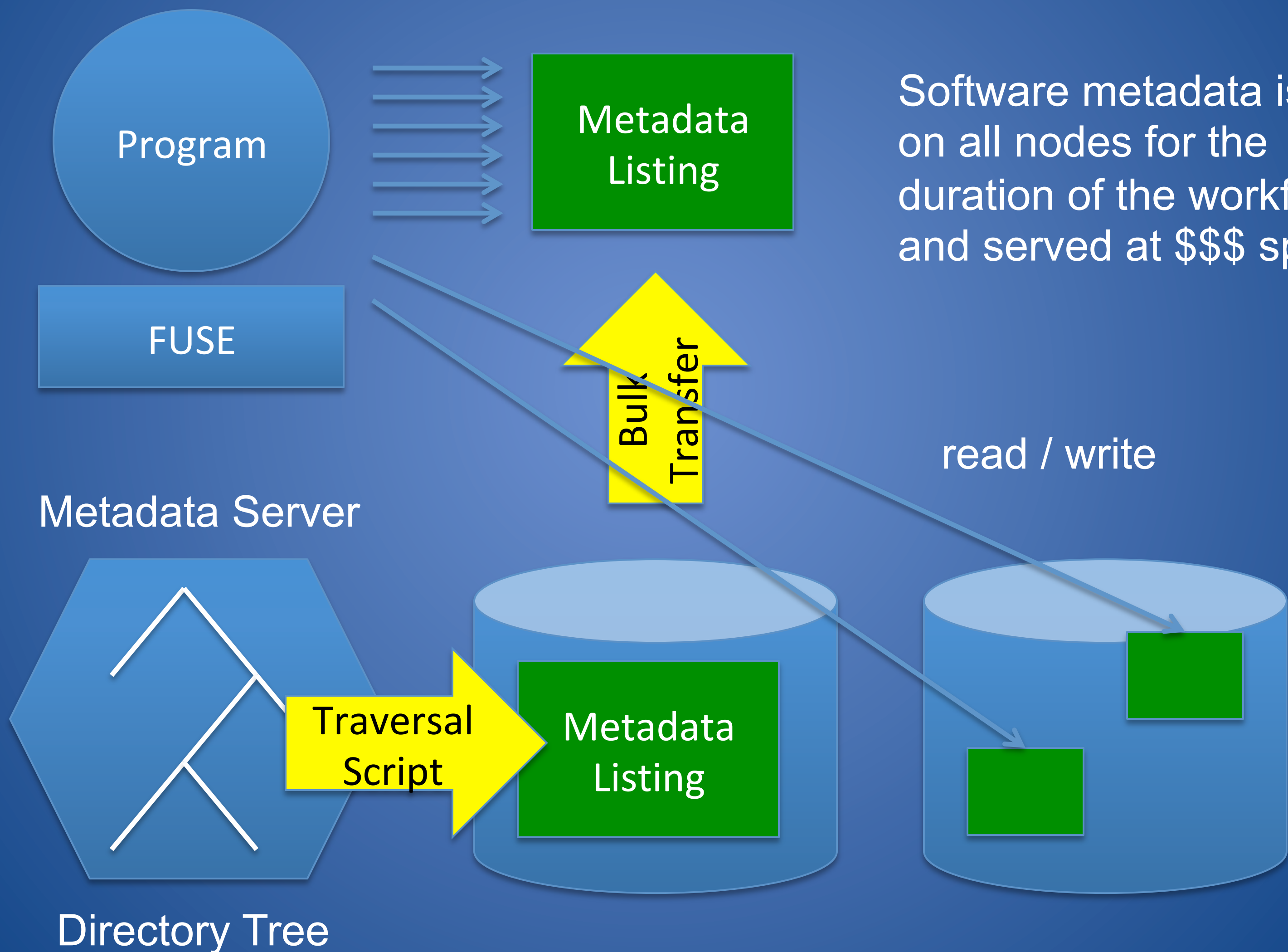


Tim Shaffer and Douglas Thain, Taming Metadata Storms in Parallel Filesystems with MetaFS, PDSW Workshop, 2017. http://dx.doi.org/10.1145/3149393.3149401

Idea: Bulk Metadata Distribution

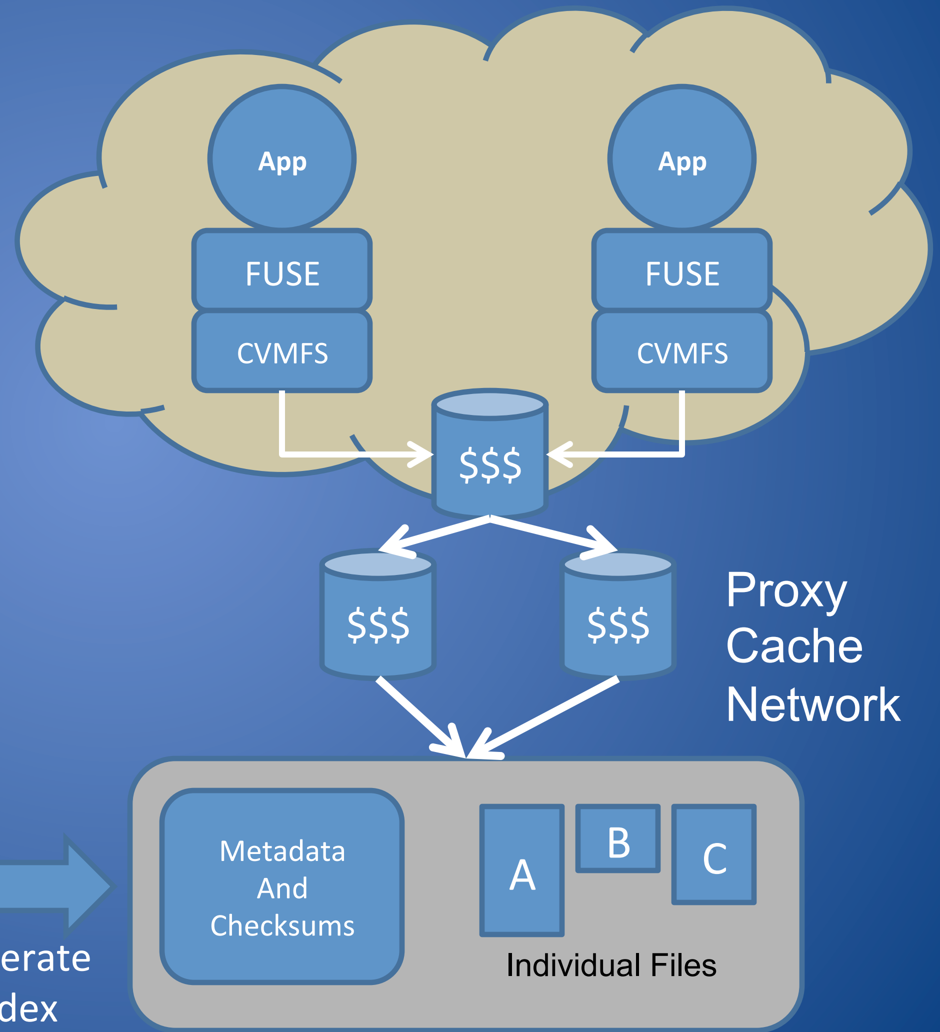
- We know some things in advance:
  - Which nodes need to load the software.
  - Which software is needed.
  - Software won't change during the run.
- Idea:
  - Build up all the metadata needed in advance.
  - Deliver it in bulk to each node.
  - Cache it for as long as the workflow runs.

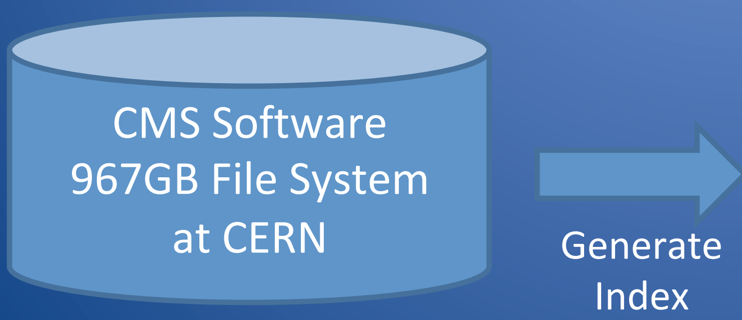
Bulk Metadata Load



Software metadata is cached on all nodes for the duration of the workflow and served at $$$ speed.

CVMFS Filesystem on >100K Cores Around the World





CVMFS: Cern-VM Filesystem

Some Quick Numbers

TABLE 1
Cache Effectiveness

			FUSE Syscalls			CernVM-FS Client Ops		
		Time	stats (x1000)	opens (x1000)	reads (MB)	HTTP Requests	Downloaded Data (MB)	Downloaded Metadata (MB)
CMS Software	cold cache	12m05s	2429	11	840	4536	895	147
	warm cache	8m14s	2429	11	772	1	0	0
Firefox	cold cache	16s	17	1	186	268	71	1.5
	warm cache	2s	17	1	186	1	0	0
LaTeX	cold cache	23s	150	2	85	351	19	12
	warm cache	17s	150	2	85	1	0	0

Nearly 2.5M metadata ops to start application

Reduced to a load of a single 147MB metadata file.

Jakob Blomer, Predrag Buncic, Rene Meusel, Gerardo Ganis, Igor Sfiligoi and Douglas Thain, **The Evolution of Global Scale Filesystems for Scientific Software Distribution,** *IEEE/AIP Computing in Science and Engineering*, **17**(6), pages 61-71, December, 2015. DOI: 10.1109/MCSE.2015.111

However CVMFS on HPC is tricky!

- Mounting filesystem on user nodes
  - FUSE -> requires some degree of privilege
  - Parrot -> requires precise ptrace behavior
- Live network access can be a problem.
  - Cache software in advance locally.
  - But which parts are needed for job X?
- CVMFS itself can be metadata intensive!
  - One site: Admins limited number of in-memory inodes allocatable by a given user, couldn't run!

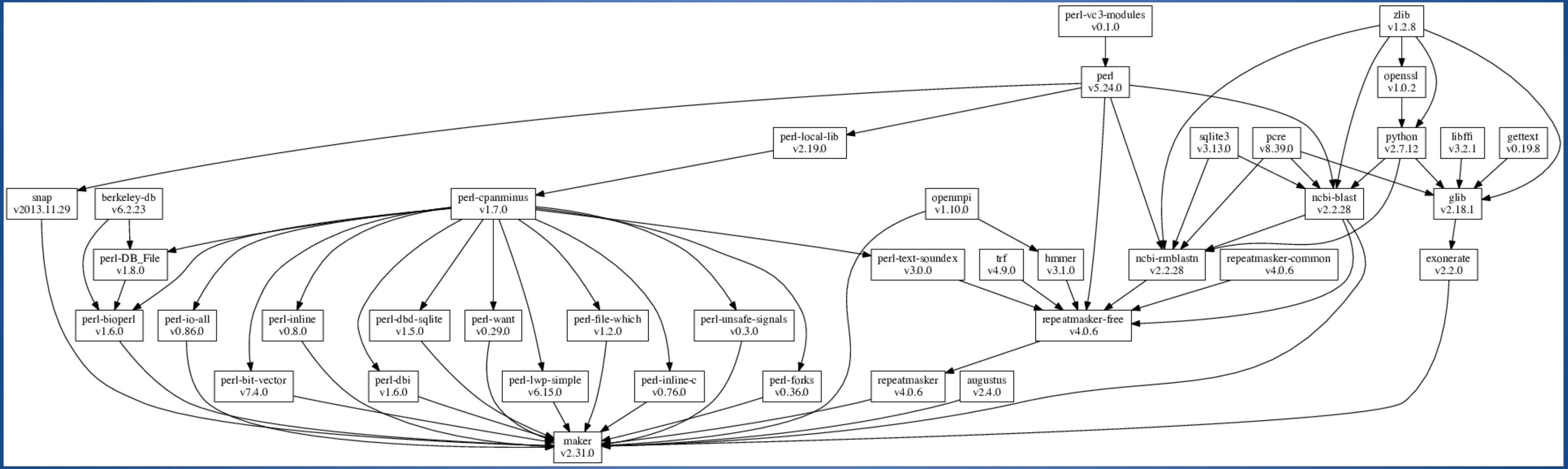
Software Deployment/Delivery

- **Filesystem Methods**
  - Big Bucket of Software!
  - MetaFS: Metadata Acceleration
  - CVMFS: A Global Filesystem
- **Packaging Methods**
  - **VC3-Builder: Automated Package Installation**
  - **Builder + Workflows**
- Container Methods
  - Wharf: Docker on Shared Filesystems
  - Containers + Workflows

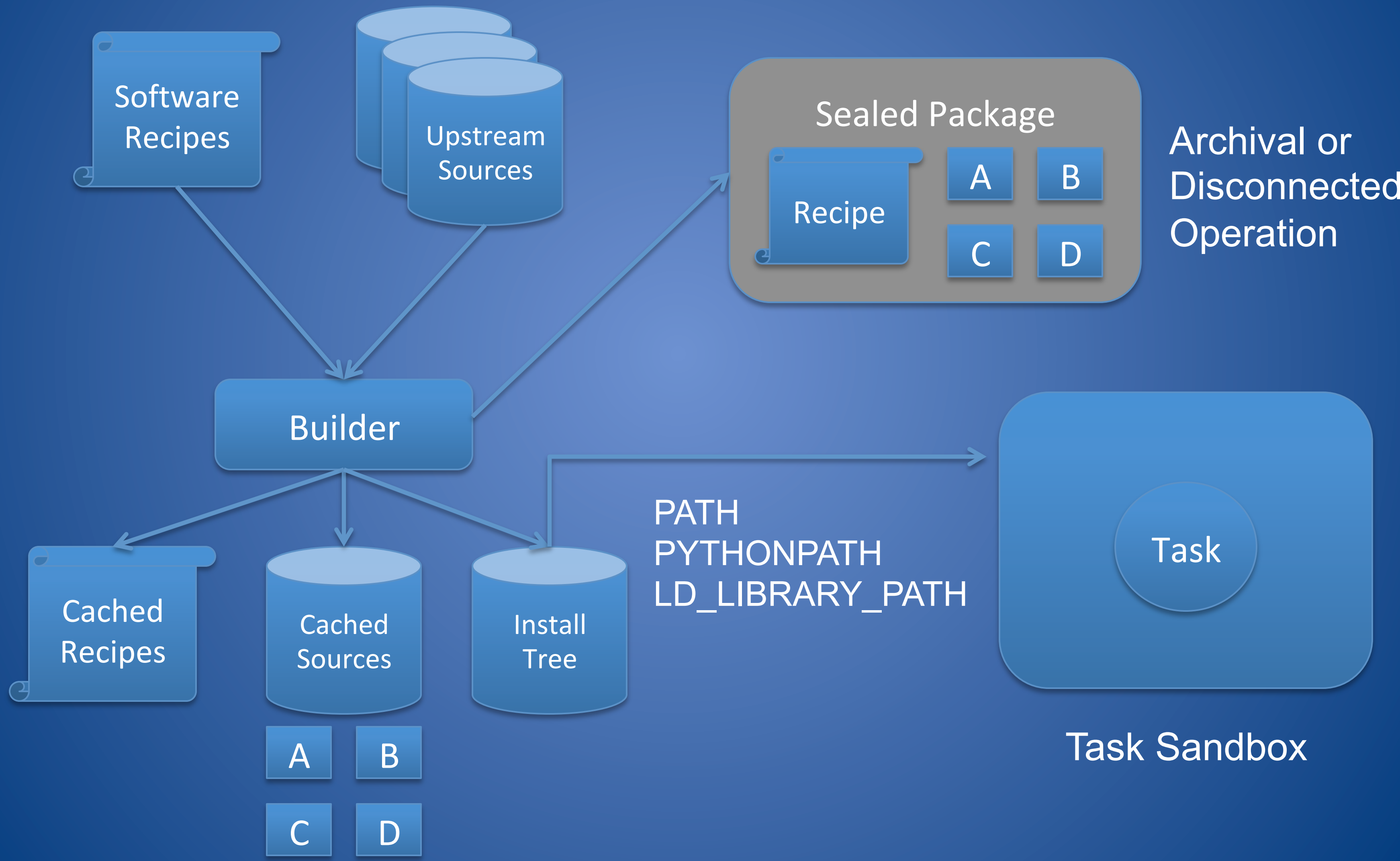
User-Level Package Managers

- Idea: Provide build recipes for many packages.
- Build software automatically in user space, each package in its own directory.
- Only activate software needed for a particular run. (PATH, LD_LIBRARY_PATH,…)
- Examples:
  - Nix – Build from ground up for reproducibility.
  - Spack – Build for integration with HPC modules.
  - VC3-Builder – Build via distributed resources.

MAKER Bioinformatics Pipeline



VC3-Builder Architecture



"vc3-builder –require ncbi-blast"

..Plan: ncbi-blast => [,]
..Try: ncbi-blast => v2.2.28
....Plan: pe
....Try: pe
....could not
....Try: pe
....could not
....Try: pe
......Plan: p
......Try: p
......Success
....Success:
....Plan: py
....Try: py
....could not
....Try: py
......Plan: c
……………

(New Shell with Desired Environment)

```
bash$  which blastx
/tmp/test/vc3-root/x86_64/redhat6/ncbi-blast/v2.2.28/
bin/blastx

bash$ blastx –help
USAGE
  blastx [-h] [-help] [-import_search_strategy filename]
   . . .

bash$ exit
```

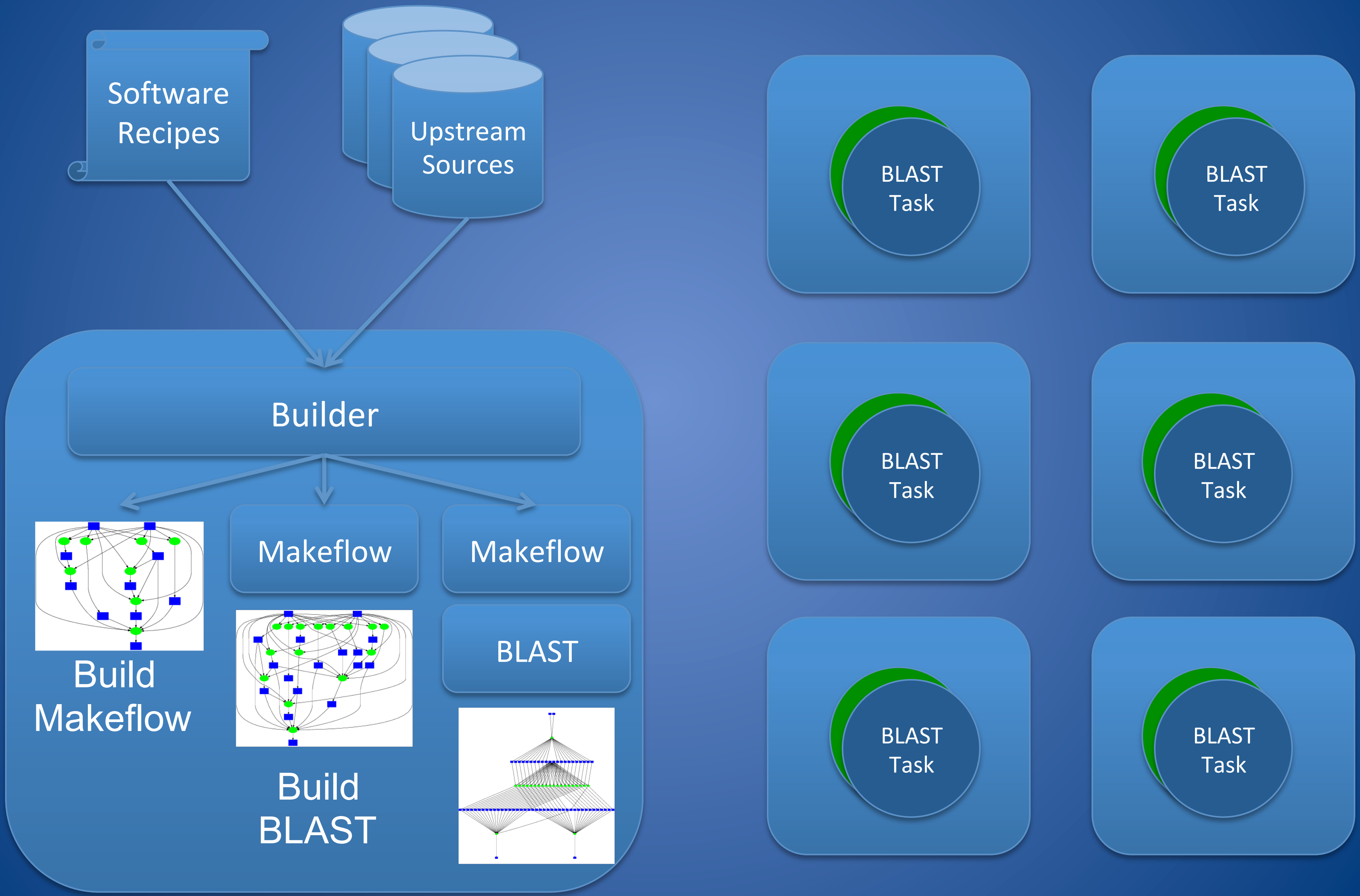
Downloading
details: /tmp/test/vc3-root/x86_64/redhat6/python/v2.7.12/python-build-log
processing for ncbi-blast-v2.2.28
preparing 'ncbi-blast' for x86_64/redhat6
Downloading 'ncbi-blast-2.2.28+-x64-linux.tar.gz' from http://download.virtualclusters.org…
details: /tmp/test/vc3-root/x86_64/redhat6/ncbi-blast/v2.2.28/ncbi-blast-build-log

Problem: Long Build on Head Node

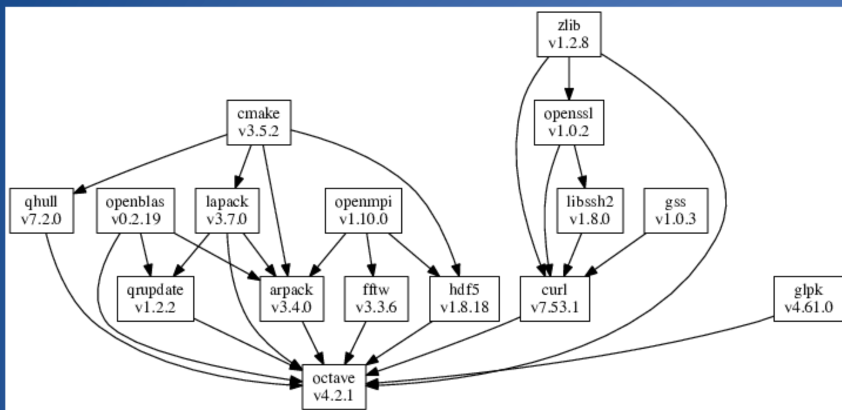
- Many computing sites limit the amount of work that can be done on the head node, so as to maintain quality of service for everyone.
- Solution: Move the build jobs out to the cluster nodes.   (Which may not have network connections.)
- Idea: Reduce the problem to something we already know how to do: Workflow!
- But how do we bootstrap the workflow software? With the builder!

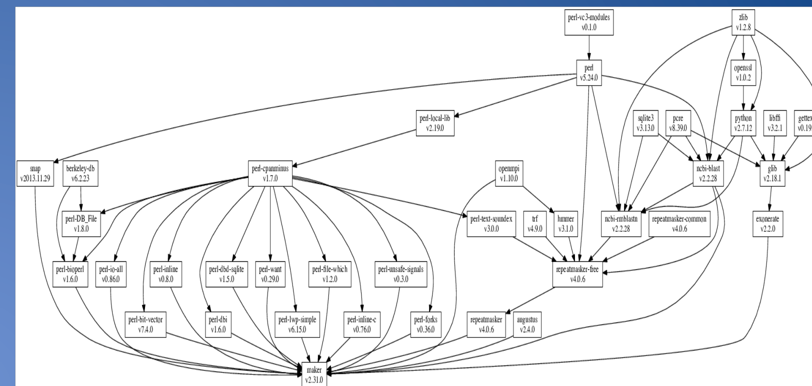
```
vc3-builder
    --require makeflow
    --require ncbi-blast
    --
    makeflow –T condor blast.mf
```

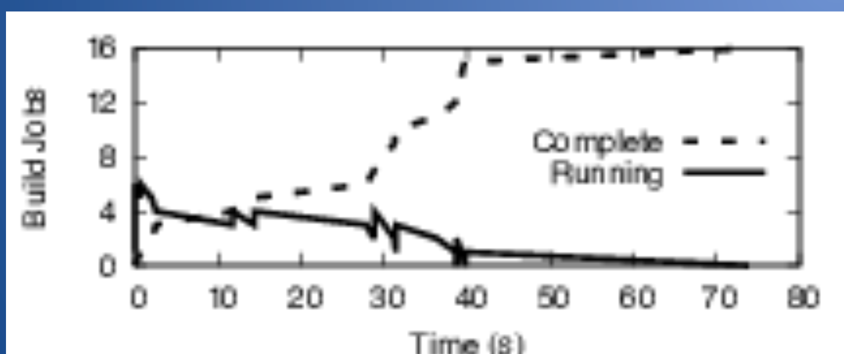
Bootstrapping a Workflow

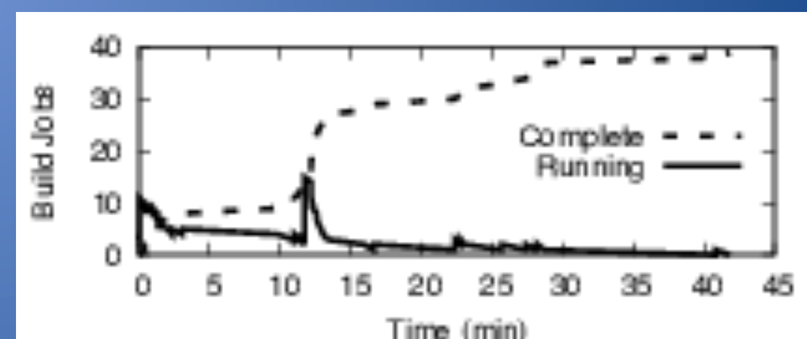


Example Applications









Octave

MAKER

Benjamin Tovar, Nicholas Hazekamp, Nathaniel Kremer-Herman, and Douglas Thain, Automatic Dependency Management for Scientific Applications on Clusters, *IEEE International Conference on Cloud Engineering (IC2E)* , April, 2018.

Delivering a Global Filesystem with VC3-Builder

"vc3-builder –require cvmfs"

..Plan: cvmfs => [,]
..Try: cvmfs => v2.0.0
....Plan: parrot => [v6.0.16,]
....Try: pa
......Plan: c
......Try: c
........Plan:
........Try:
........Succes
......Fail-prer
........Plan:
........Try:
..........Plan:
..........Try:
..........Succe
........could n
........Try:
..........Plan:
..........Try:
..........Success: perl-vc3-modules v0.1.0 => [v0.1.0,]
........could not add any source for: perl v5.016 => [v5.10.0, v5.10001.0]
........Try: perl => v5.24.0
..........Plan: perl-vc3-modules => [v0.001.000,]
..........Try: perl-vc3-modules => v0.1.0
..........Success: perl-vc3-modules v0.1.0 => [v0.1.0,]
........Success: perl v5.24.0 => [v5.10.0, v5.10001.0]

(New Shell with Desired Environment)

bash$ **ls /cvmfs/oasis.opensciencegrid.org**

atlas csiu geant4 ilc nanohub osg-software
auger enmr glow ligo nova sbgrid
cmssoft fermilab gluex mis osg
snoplussnolabca
. . .

bash$ exit

Software Deployment/Delivery

- **Filesystem Methods**
  - Big Bucket of Software!
  - MetaFS: Metadata Acceleration
  - CVMFS: A Global Filesystem
- Packaging Methods
  - VC3-Builder: Automated Package Installation
  - Builder + Workflows
- **Container Methods**
  - **Container Technologies**
  - **Containers + Workflows**

Many Possible Container Techs

docker

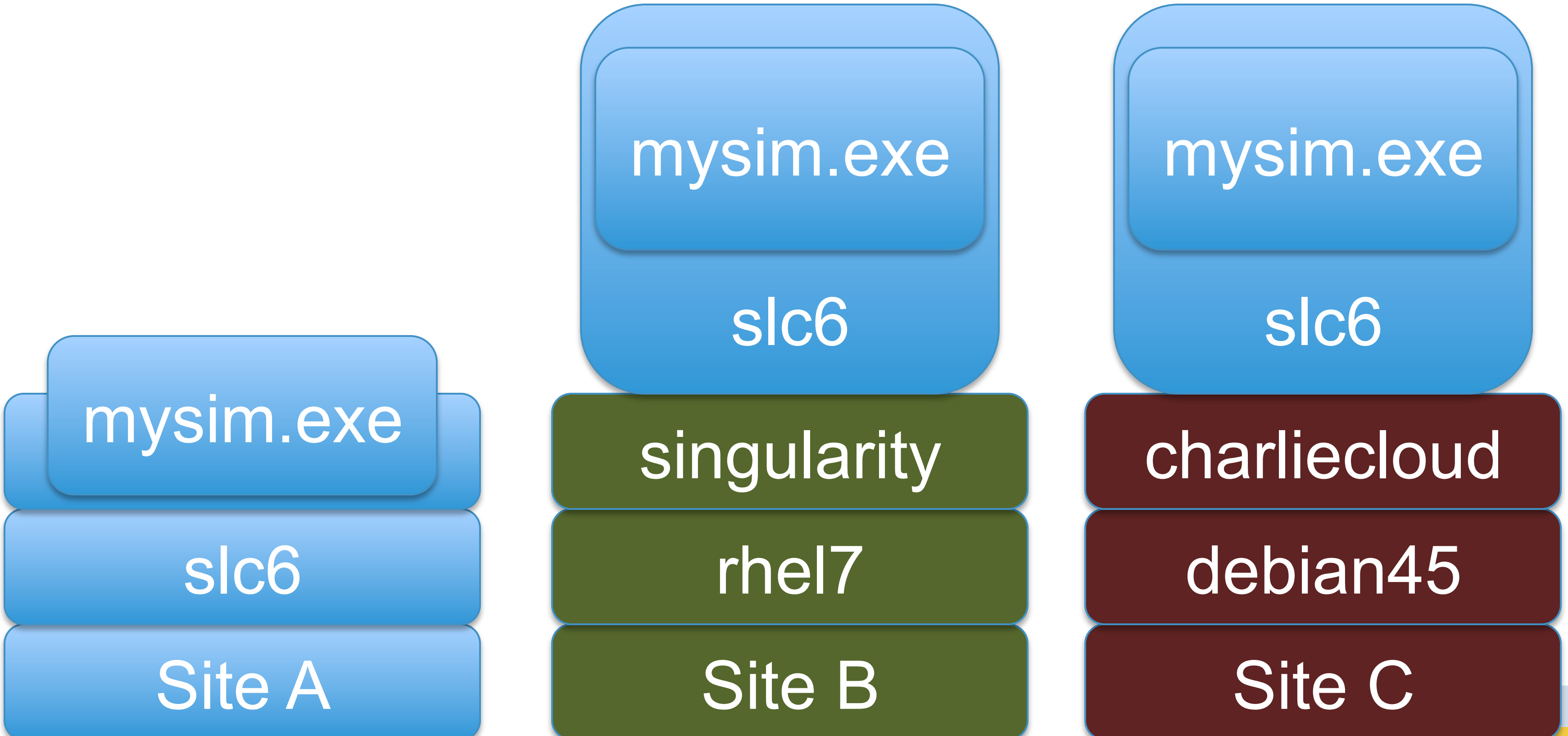
- ✔ Widely used
- ✔ Convenient global repo
- ❌ Builds up images locally
- ❌ Root Daemon

Charliecloud

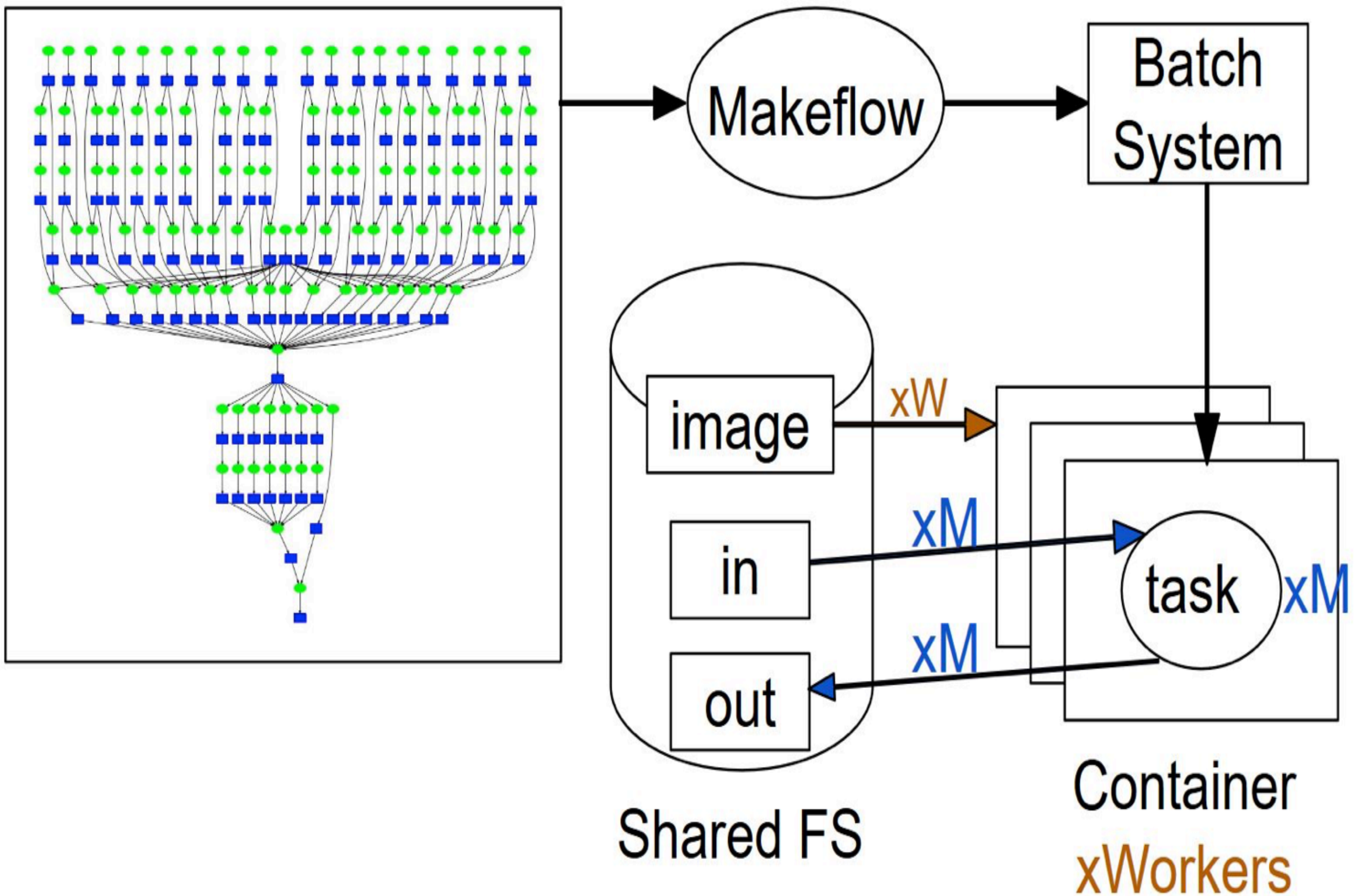
- ✔ Built on Docker Images
- ✔ No Root Daemon
- ❌ Requires Very Modern Kernel

- ✔ No Root Daemon
- ✔ Only one file
- ✔ Works with many image types
- ❌ Loop Devices

"runos slc6 – mysim.exe"



Desired Architecture

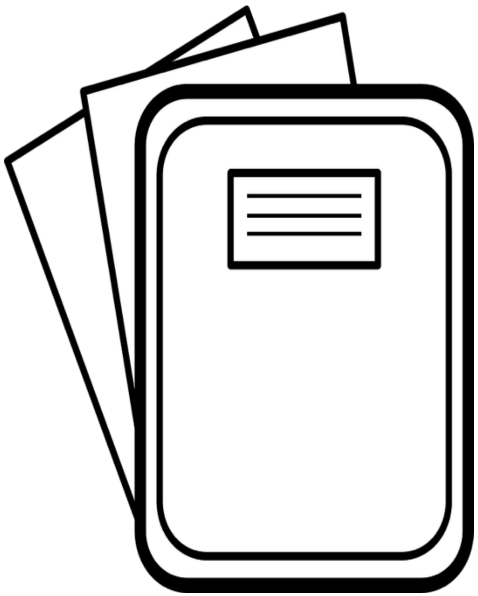


Types of Data

OS

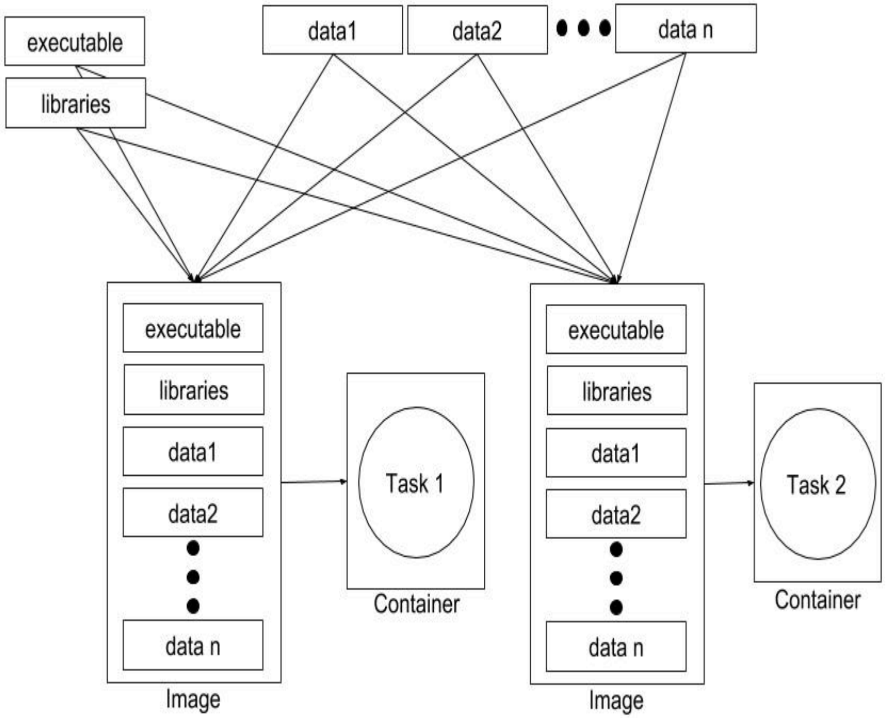
Read-Only

Workdata

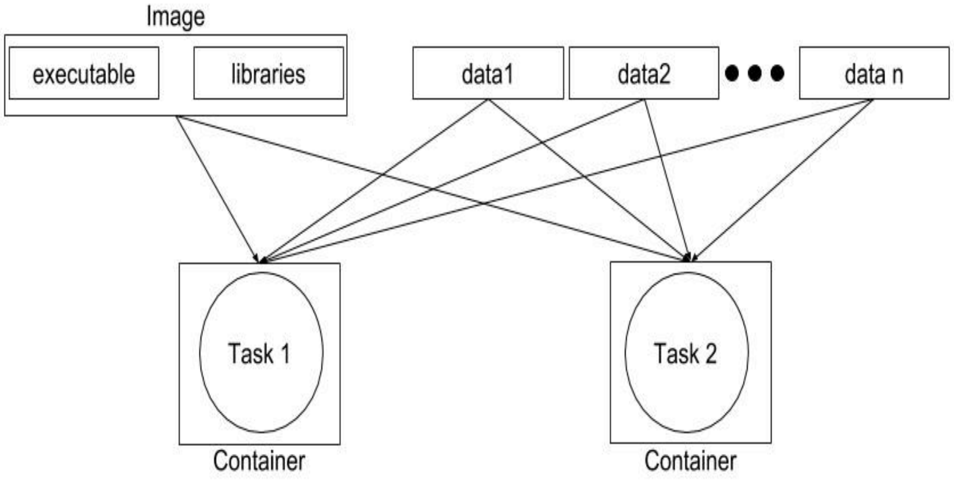


Container Composition

Static Composition

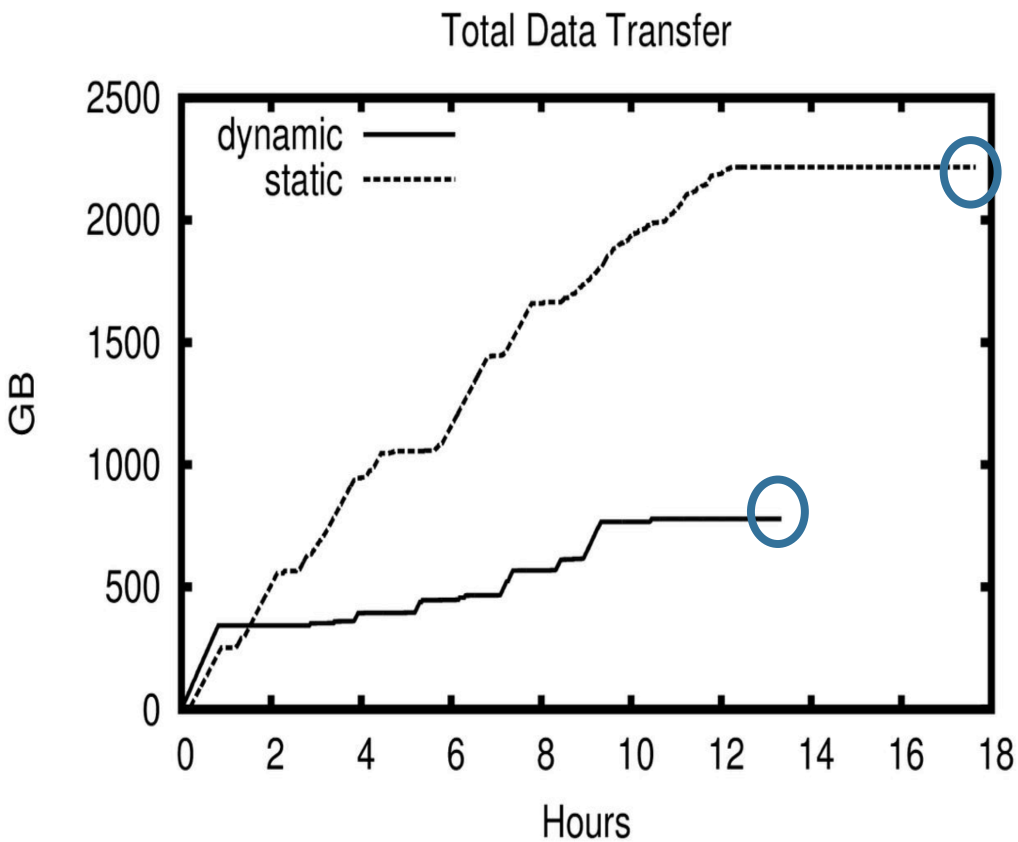


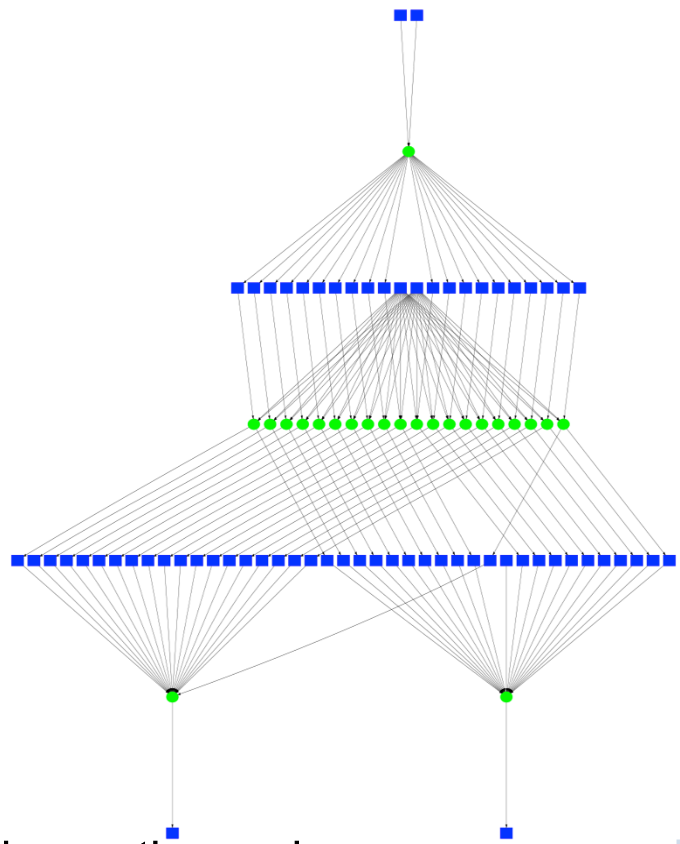
Dynamic Composition



Kyle Sweeney and Douglas Thain,
Efficient Integration of Containers into Scientific Workflows,
Science Cloud Workshop at HPDC, June, 2018. DOI: 10.1145/3217880.3217887

Experiment: BLAST

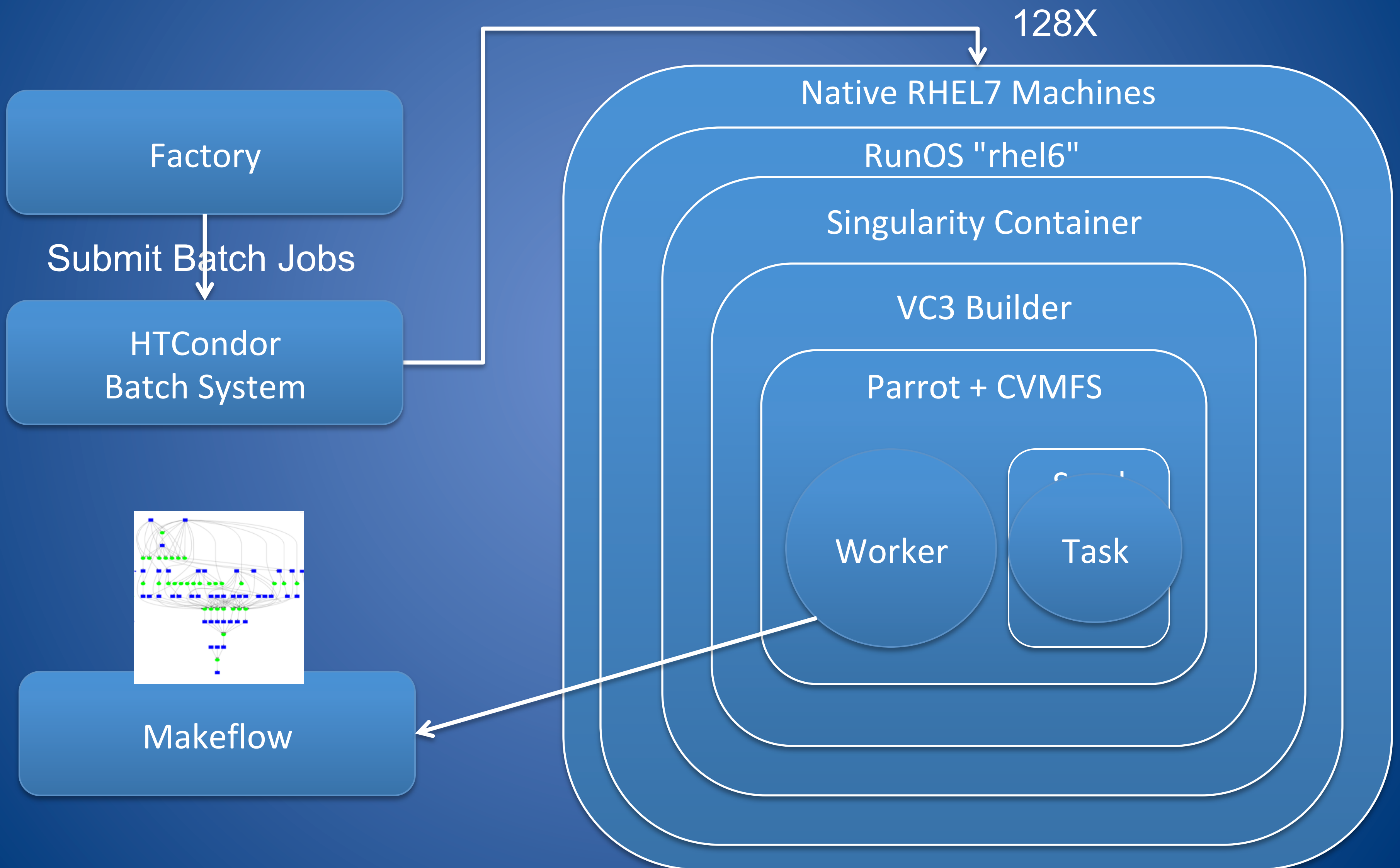




We delivered 1/3rd less data, and finished in ~3/4ths the runtime using dynamic composition

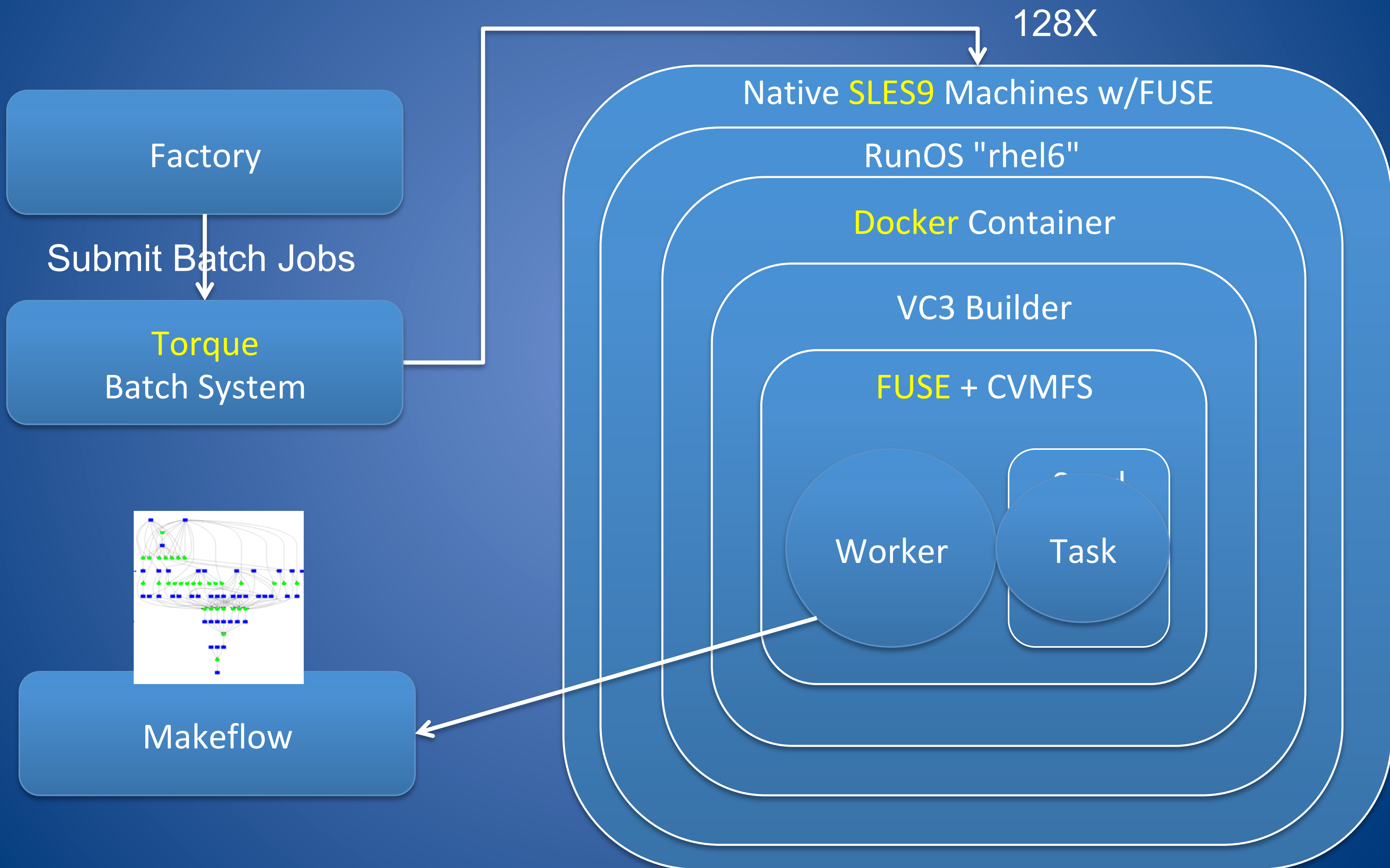
Putting it All Together

Request 128 nodes of16 cores, 4G RAM, 16G disk
with RHEL6 operating system, CVMFS and Maker software installed:



Same Thing, Different Site:

Request 128 nodes of16 cores, 4G RAM, 16G disk
with RHEL6 operating system, CVMFS and Maker software installed:



- Big Bucket of Software
  + Maximum portability, compatibility, archivability.
  - Horrible metadata performance.
  + / - Correct with metadata oriented filesystems.
- User-Level Package Managers
  + Explicit statement of dependences. (repro!)
  + Deliver only needed components. (sharing!)
  - Long build/deploy processes. (use cluster)
- Container Technologies
  + Leverage commodity software tools.
  + Naturally metadata efficient.
  - Requires privileges, kernel tech, specialized tools.
  - Create new storage management problems.

Thoughts on Dependencies:

- Make software dependencies more explicit.
  - Proposed: Nothing should be available by default, all software should require an "import" step.
- Need better, portable, ways of expressing:
  - What software environment the user wants.
  - What software components are actually ***used***.
  - What environment the site provides.
- The ability to nest environments is critical!
  - Sysadmin provisions machine via VM/container.
  - Batch system provisions slot with container.
  - User provisions software with container.

Thoughts on Filesystems

- Open/read/write/close has worked well for a long time, but seems to be too small a granularity for large scale systems/software.
- Can we have flexible transaction to balance between small changes and wide distribution?
- Do we need new filesystem ops?
  - fd = Opentree("/home/dthain",O_RDONLY);
  - Results = Search("$PATH","sim.exe");
  - Something like SQL for metadata?

Acknowledgements

People in the Cooperative Computing Lab


Douglas Thain
Director


Benjamin Tovar
Research Soft. Engineer


Peter Ivie


Nicholas Hazekamp


Charles Zheng


Nathaniel Kremer-Herman


Tim Shaffer


Kyle Sweeney

Notre Dame CMS:
Kevin Lannon
Mike Hildreth
Kenyi Hurtado

Univ. Chicago:
Rob Gardner
Lincoln Bryant
Suchandra Thapa
Benedikt Riedel

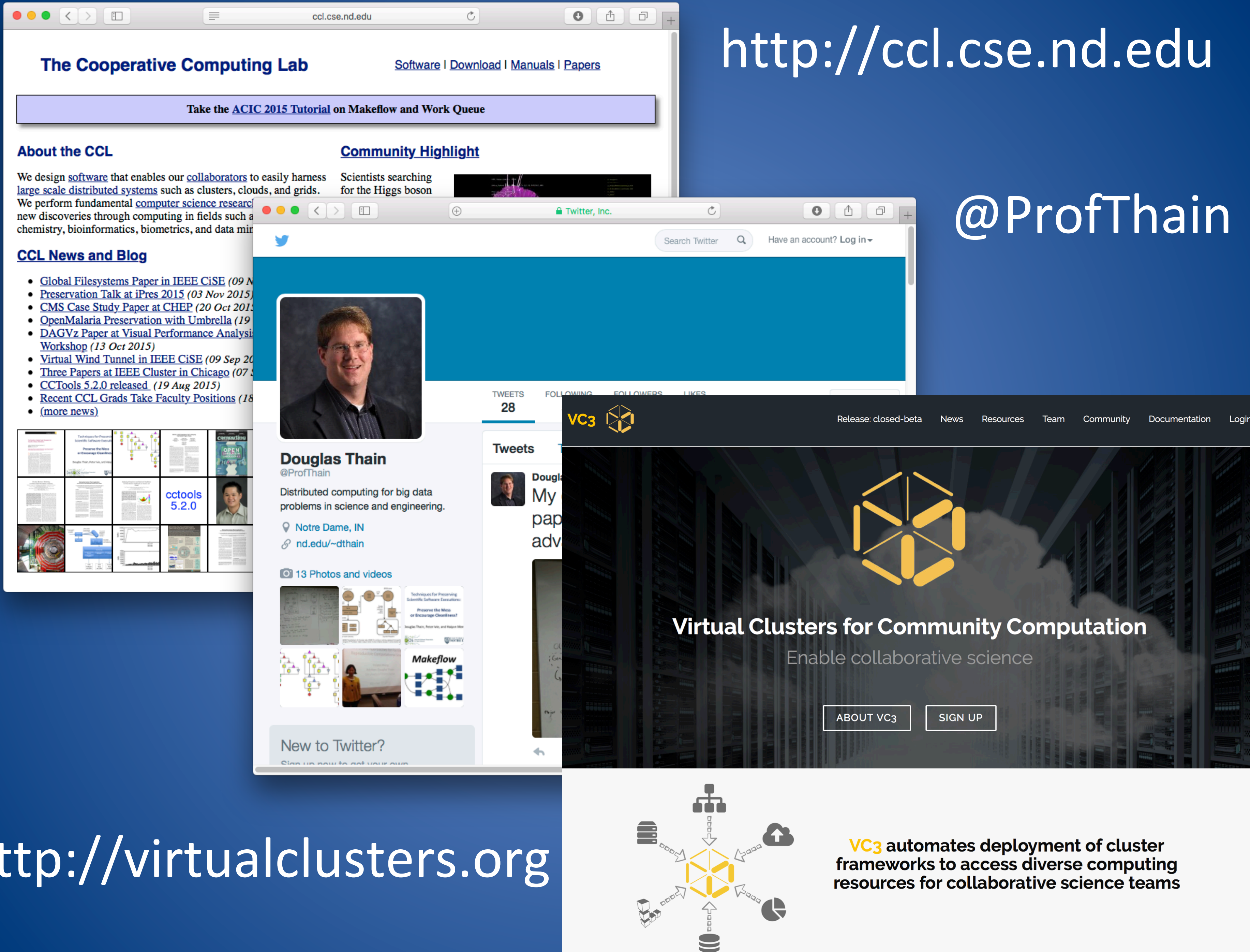
Brookhaven Lab:
John Hover
Jose Caballero

DE-SC0015711
VC3: Virtual Clusters for Community Computation

ACI-1642409
SI2-SSE: Scaling up Science on Cyberinfrastructure with the Cooperative Computing Tools

http://ccl.cse.nd.edu

@ProfThain



http://virtualclusters.org