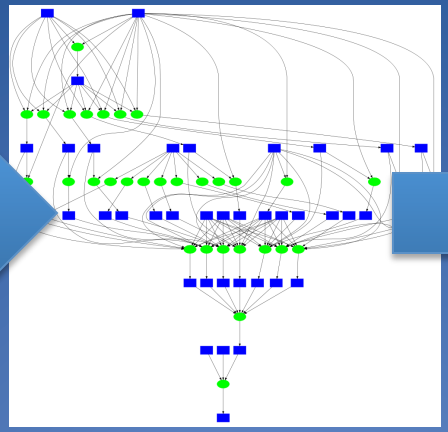




Seamless Scientific Computing from Laptops to Clouds

Prof. Douglas Thain, University of Notre Dame

http://www.nd.edu/~dthain
dthain@nd.edu
@ProfThain

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 *develop open source software* for large scale distributed computing.

http://ccl.cse.nd.edu

The Cooperative Computing Lab

Software | Download | Manuals | Papers

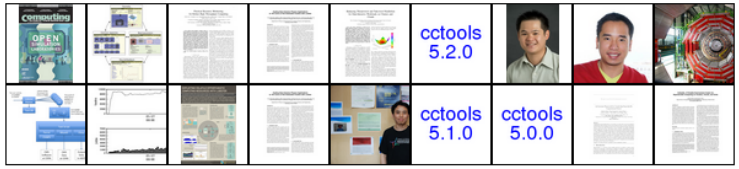
Take the ACIC 2015 Tutorial on Makeflow and Work Queue

About the CCL

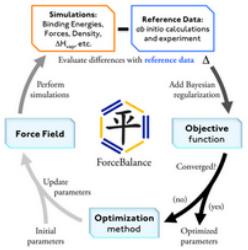
We design software that enables our collaborators to easily harness large scale distributed systems such as clusters, clouds, and grids. We perform fundamental computer science research in that enables new discoveries through computing in fields such as physics, chemistry, bioinformatics, biometrics, and data mining.

CCL News and Blog

- Virtual Wind Tunnel in IEEE CiSE *(09 Sep 2015)*
- Three Papers at IEEE Cluster in Chicago *(07 Sep 2015)*
- CCTools 5.2.0 released *(19 Aug 2015)*
- Recent CCL Grads Take Faculty Positions *(18 Aug 2015)*
- CMS Analysis on 10K Cores Using Lobster *(14 Aug 2015)*
- Haipeng Cai Defends Ph.D. *(16 Jul 2015)*
- CCTools 5.1.0 released *(16 Jul 2015)*
- CCTools 5.0.0 released *(07 Jul 2015)*
- Preservation Framework for Computational Reproducibility at ICCS 2015 *(01 Jul 2015)*
- (more news)



Community Highlight



ForceBalance is an open source software tool for creating accurate force fields for molecular mechanics simulation using flexible combinations of reference data from experimental measurements and theoretical calculations. These force fields are used to simulate the dynamics and physical properties of molecules in chemistry and biochemistry.

The Work Queue framework gives ForceBalance the ability to distribute computationally intensive components of a force field optimization calculation in a highly flexible way. For example, each optimization cycle launched by ForceBalance may require running 50 molecular dynamics simulations, each of which may take 10-20 hours on a high end NVIDIA GPU. While GPU computing resources are available, it is rare to find 50 available GPU nodes on any single supercomputer or HPC cluster. With Work Queue, it is possible to distribute the simulations across several HPC clusters, including the Certainty HPC cluster at Stanford, the Keeneland GPU cluster managed by Georgia Tech and Oak Ridge National Laboratories, and the Stampede supercomputer managed by the University of Texas. This makes it possible to run many simulations in parallel and complete the high level optimization in weeks instead of years.

- Lee-Ping Wang, Stanford University

Research

- Papers
- Projects
- People
- Jobs
- REU

Software

- Download
- Manuals
- Makeflow
- Work Queue
- Parrot
- Chirp
- SAND
- AWE

Community

- Forum
- Getting Help
- Highlights
- Annual Meeting
- Workshops
- For Developers

Operations

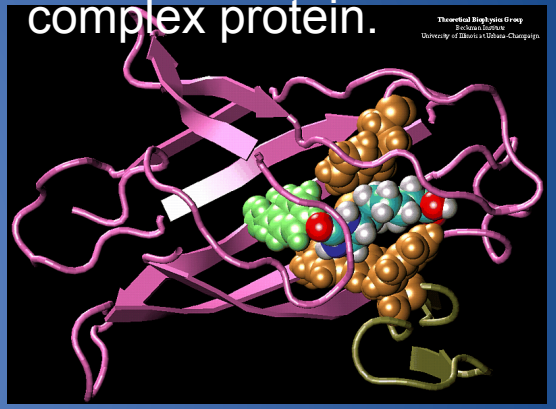
- Condor Display
- Condor Pool
- Hadoop Cluster
- Biocompute
- BXGrid
- Condor Log Analyzer
- Internal

http://ccl.cse.nd.edu

Some of Our Collaborators

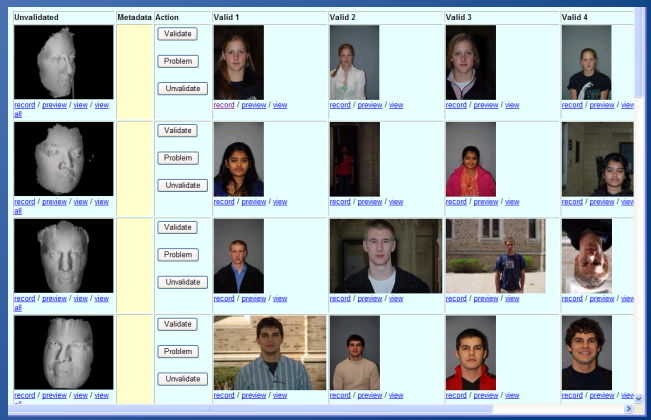
K. Lannon: Analyze 2PB of data produced by the LHC experiment at CERN

J. Izaguirre: Simulate 10M different configurations of a complex protein.



S. Emrich: Analyze DNA in thousands of genomes for similar sub-sequences.

P. Flynn: Computational experiments on millions of acquired face videos.



From the scientist's perspective...

It took ~~a while~~ (most of a year) but now I have my code written, installed, debugged, calibrated, and verified on my laptop.

Now I want to run at a scale 1000x larger by using a cluster, cloud, grid, or whatever you computer people are calling it today.

There is no way you are going to convince me to re-write this valuable program in order to run on your crazy cluster / OS / framework!

(Important science codes outlive OS/HW.)

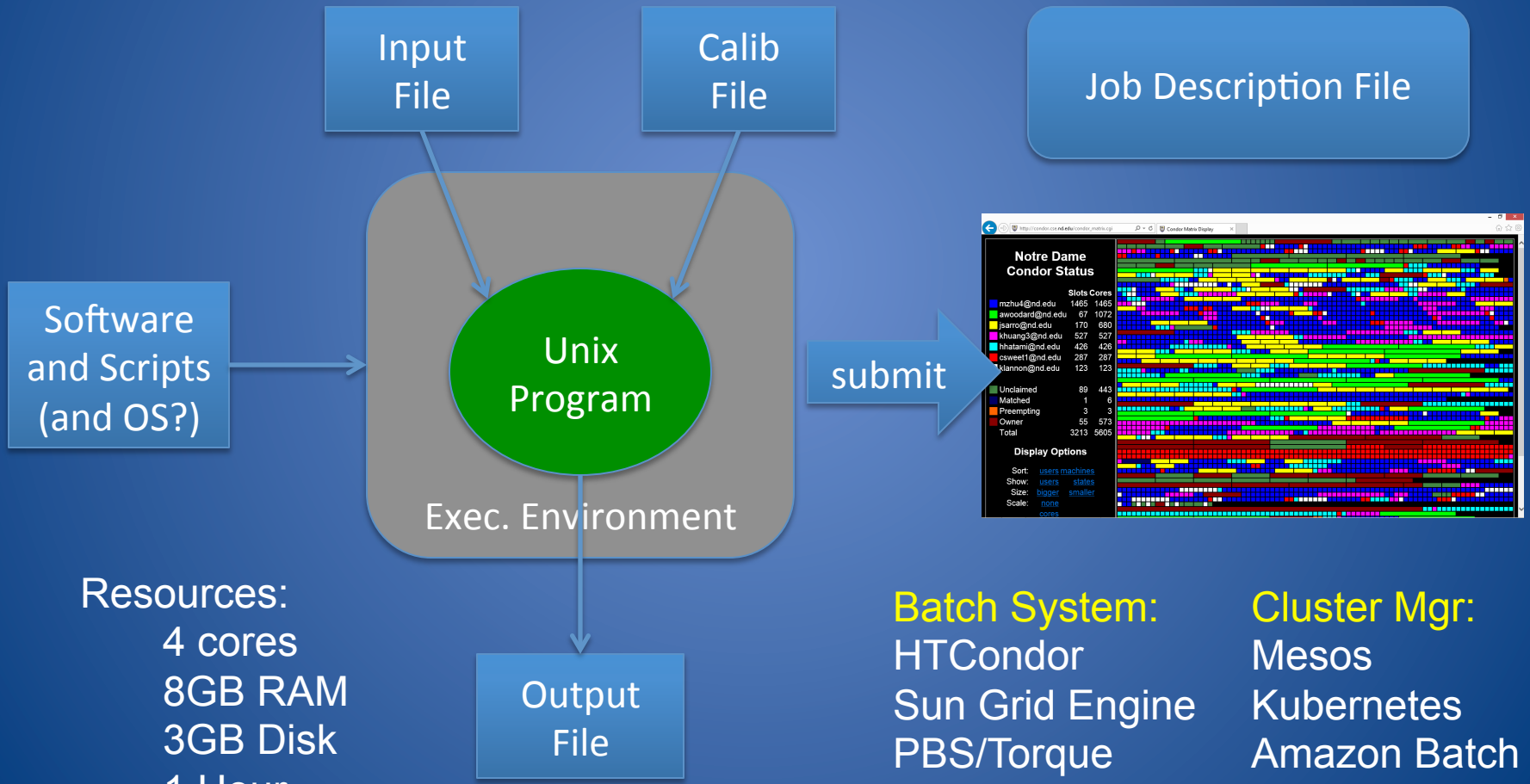
On my laptop…

"sim.exe –p 50 in.dat -o output.dat"

Output!

But to run on the cloud…

"sim.exe –p 50 in.dat -o output.dat"



Resources:
- 4 cores
- 8GB RAM
- 3GB Disk
- 1 Hour

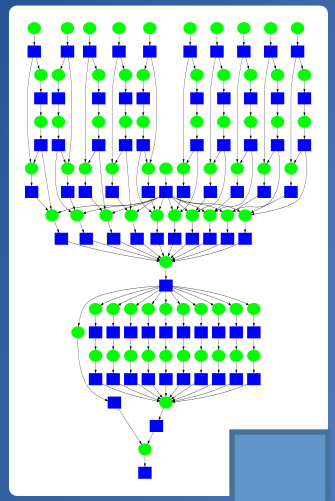
Batch System:
- HTCondor
- Sun Grid Engine
- PBS/Torque

Cluster Mgr:
- Mesos
- Kubernetes
- Amazon Batch

Outline

- The Laptop Perspective
- Scaling Up with Makeflow and Work Queue
- VC3: Virtual Clusters
- Problem: Software Deployment
- Problem: Resource Sizing
- Lessons Learned

Makeflow = Make + Workflow



- Provides portability across batch systems.
- Enables parallelism (but not too much!)
- Fault tolerance at multiple scales.
- Data and resource management.
- Transactional semantics for job execution.

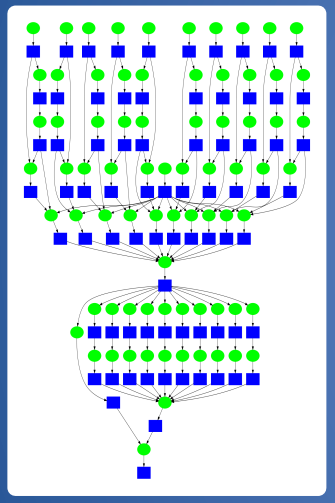
Makeflow

Local | HTCondor | Torque | Work Queue | Amazon

http://ccl.cse.nd.edu/software/makeflow

Makeflow Shapes a Workflow

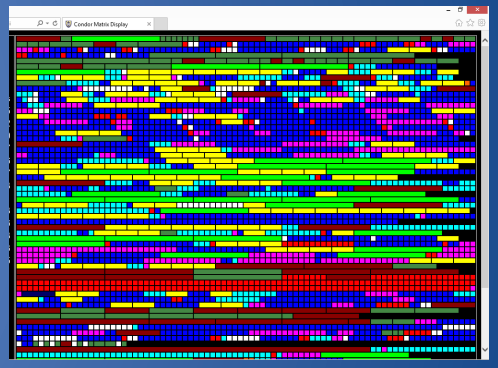
Millions of Tasks



Concurrency
and Policy Control

Make
Flow

Cluster or Cloud

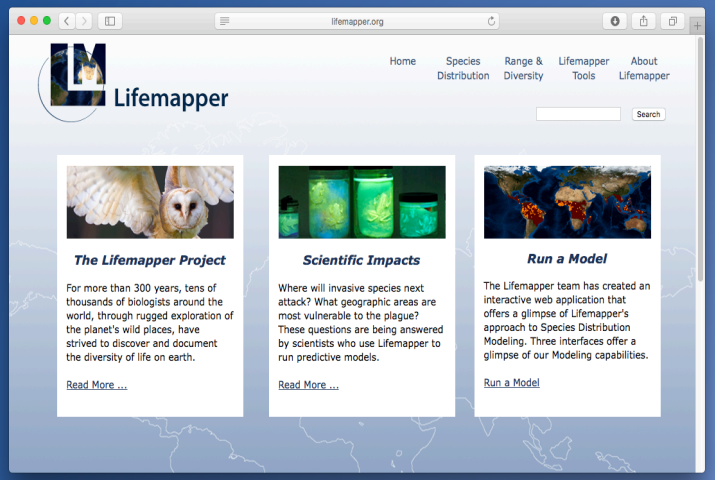


Precise
Cleanup

Transaction
Log

Performance
Monitoring

Example: Species Distribution Modeling



Full Workflow:

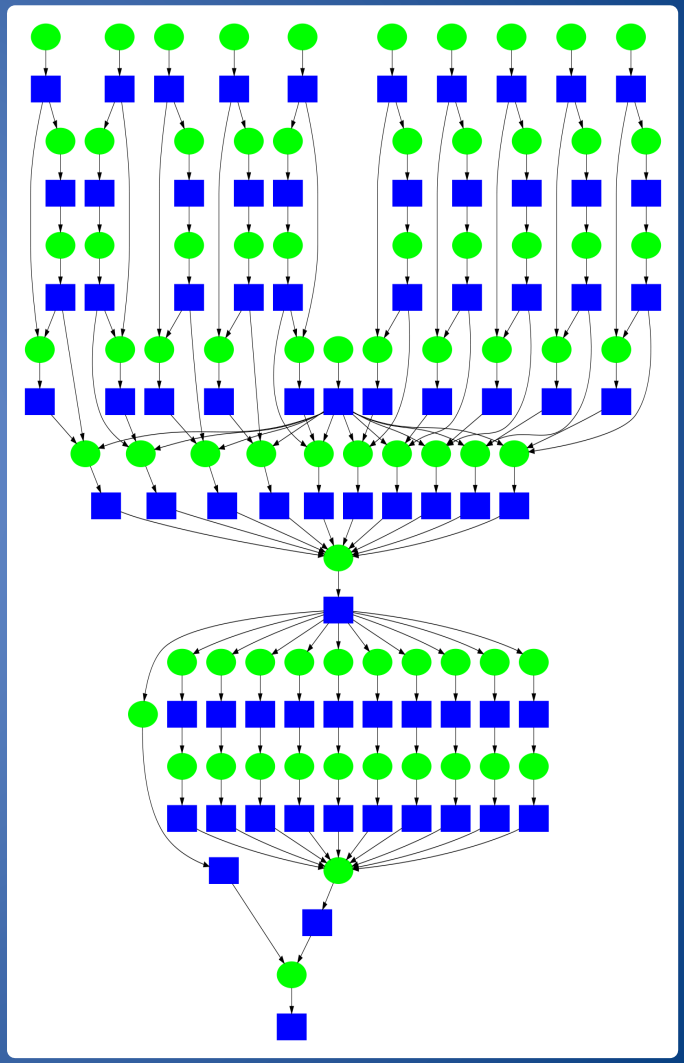
12,500 species

x 15 climate scenarios

x 6 experiments

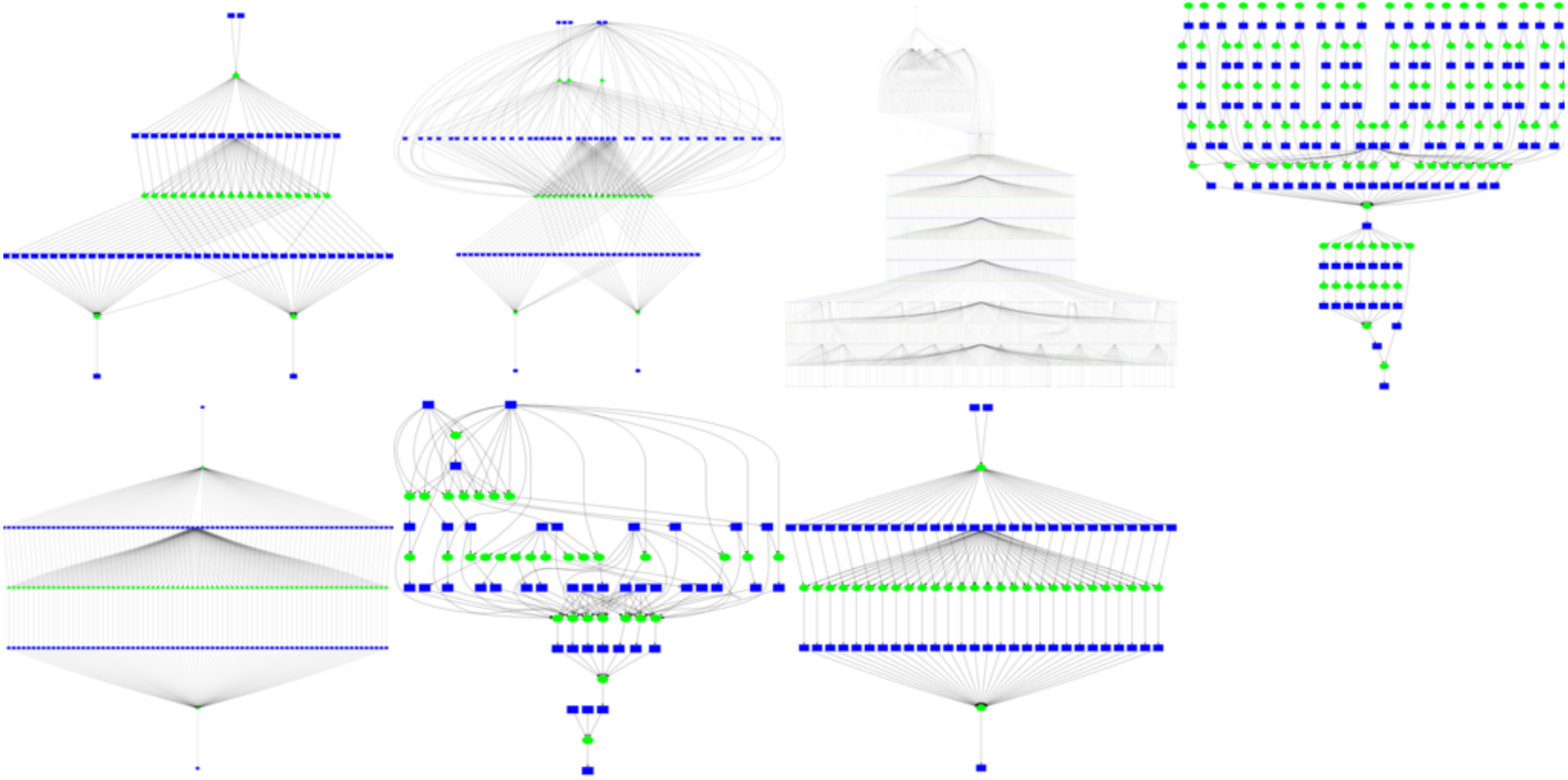
x 500 MB per projection

= 1.1M jobs, 72TB of output



Small Example: 10 species x 10 expts

More Examples

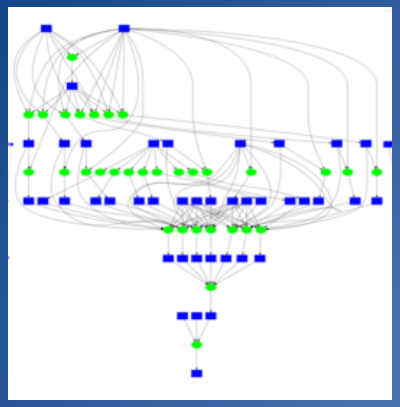


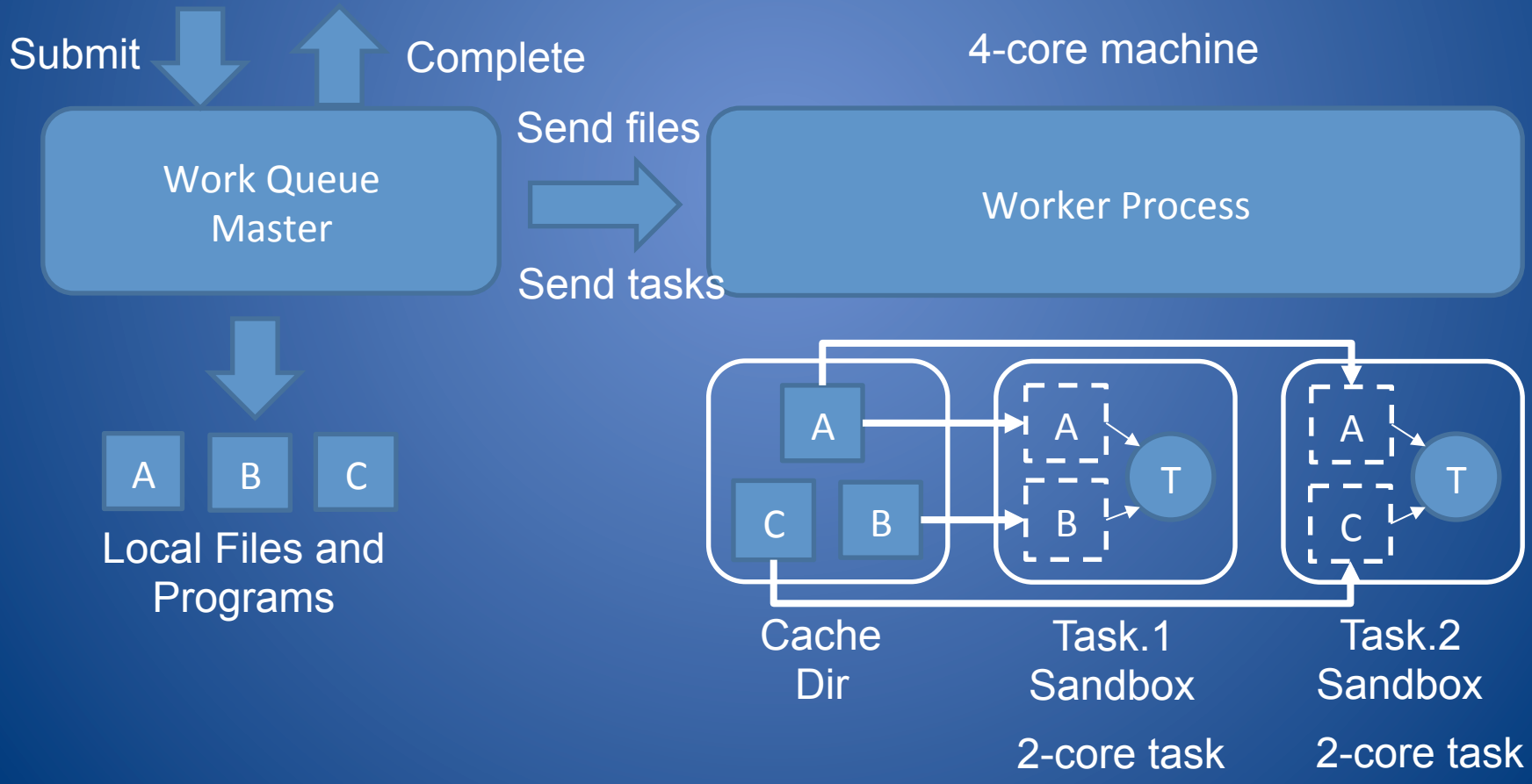
http://github.com/cooperative-computing-lab/makeflow-examples

Limitations of Direct Submission to Batch Systems

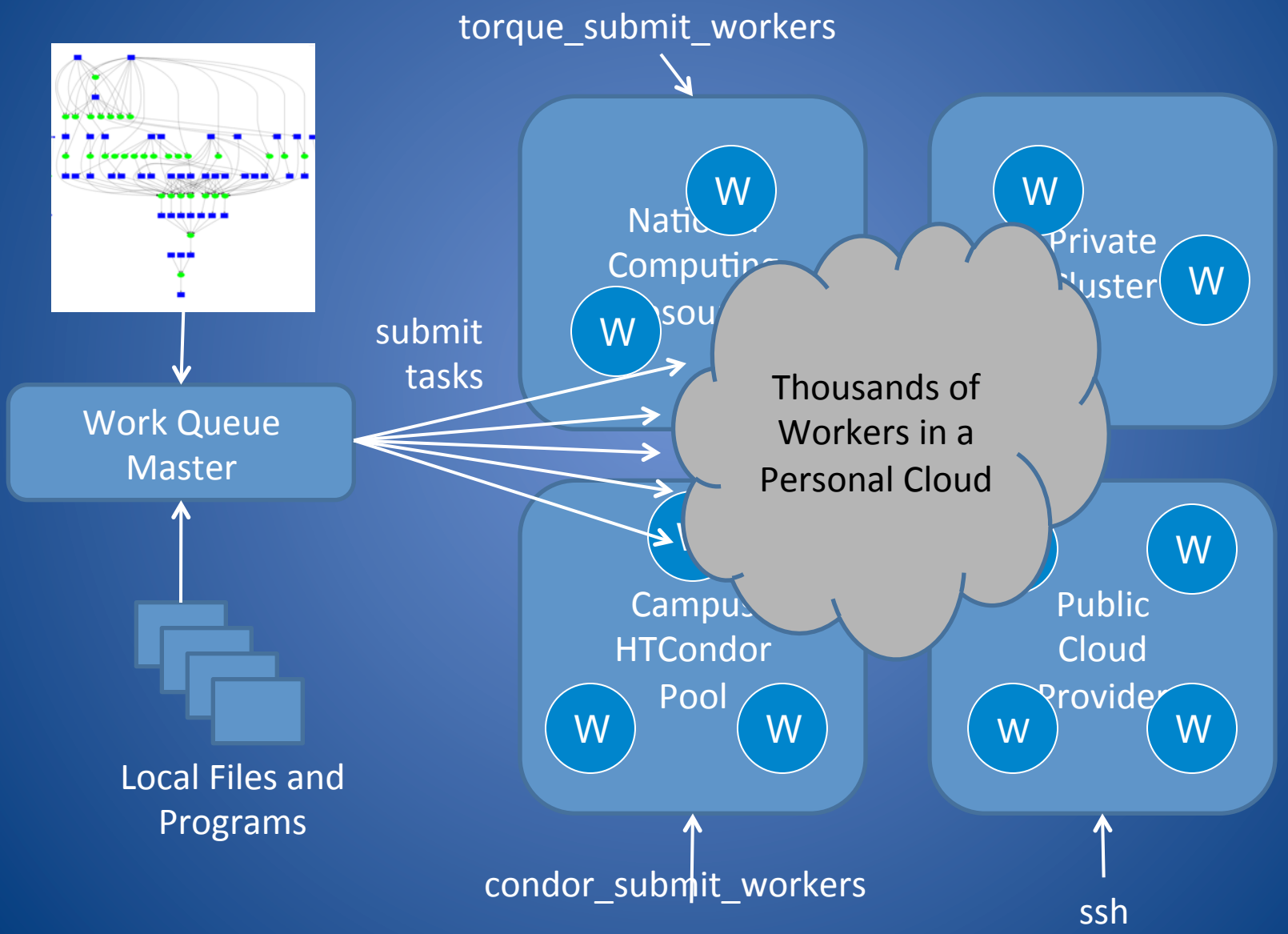
- Dispatch Latency
  - Rule of Thumb: 30 seconds to provision a resource and start a job. (UGE, HTCondor, Amazon, …)
- No Data Locality
  - Same input files get sent again and again to the same jobs. (sometimes even on the same host.)
- Solution: Deploy Work Queue Overlay
  - Accelerate subsequent job starts.
  - Share data between concurrent tasks on a node.

Work Queue Architecture



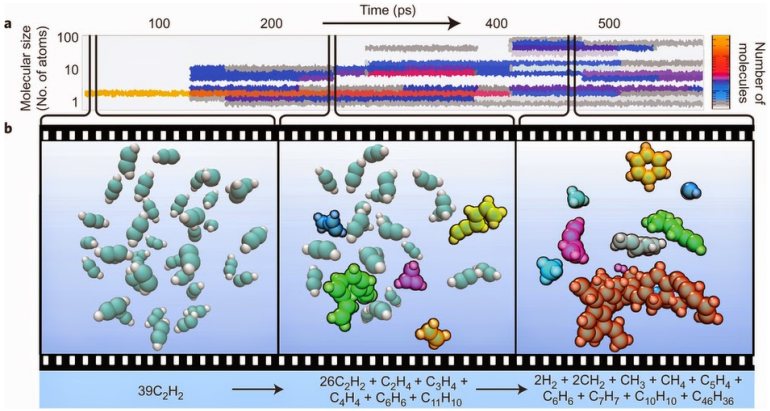


Harness Multiple Resources

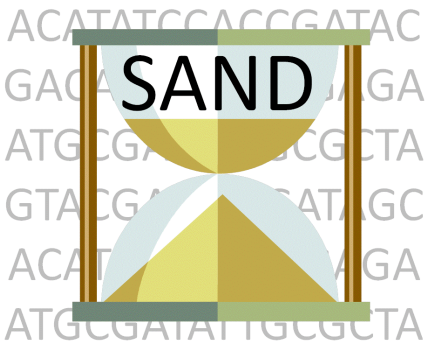


Work Queue Applications

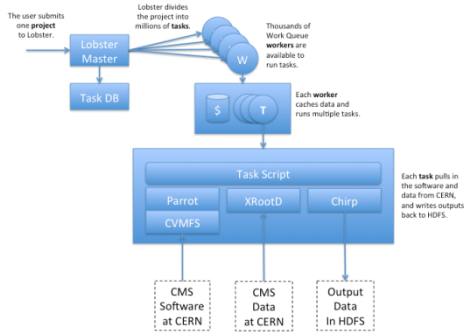
Nanoreactor MD Simulations



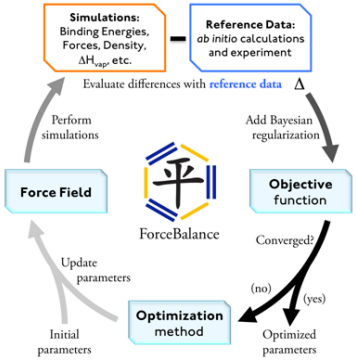
Scalable Assembler at Notre Dame



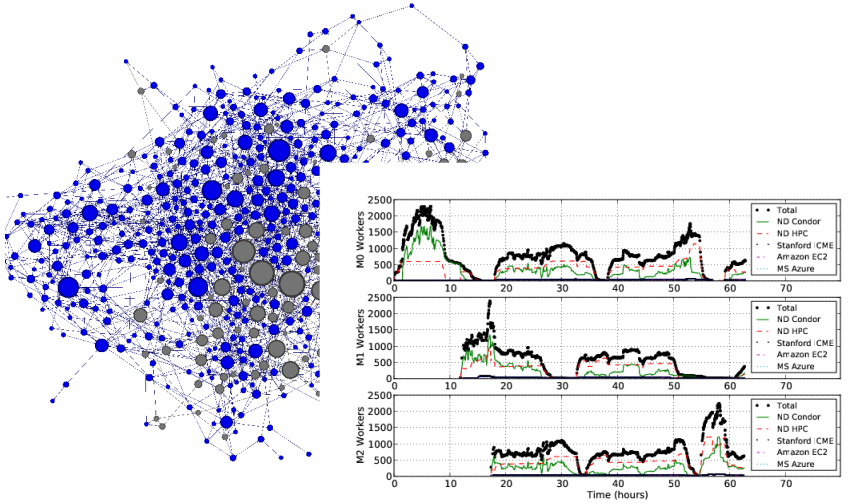
Lobster HEP



ForceBalance



Adaptive Weighted Ensemble



Scaling Up Problems

- OS and Software Environment
  - User has a long and vague history of installing software packages on demand to solve problems.
- Exact Specification of Data Dependencies
  - "Oh, I forgot about that 1TB calibration file."
- Resource Selection
  - "My laptop has 16GB RAM!"
  - "But how much does the application need?"
- Time/Cost of Moving Data
  - Does it pay to move 1TB of data in order to run one task for one hour? What about 1M tasks?

Outline

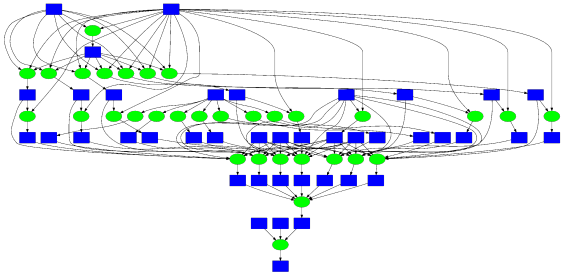
- The Laptop Perspective
- Scaling Up with Makeflow and Work Queue
- VC3: Virtual Clusters
- Problem: Software Deployment
- Problem: Resource Sizing
- Lessons Learned

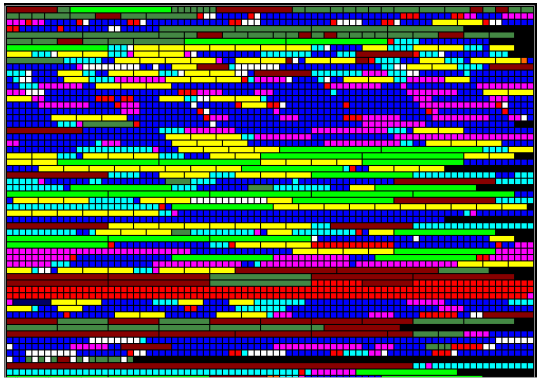
VC3: Virtual Clusters for Community Computation

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

http://virtualclusters.org

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.)

Open Science Grid

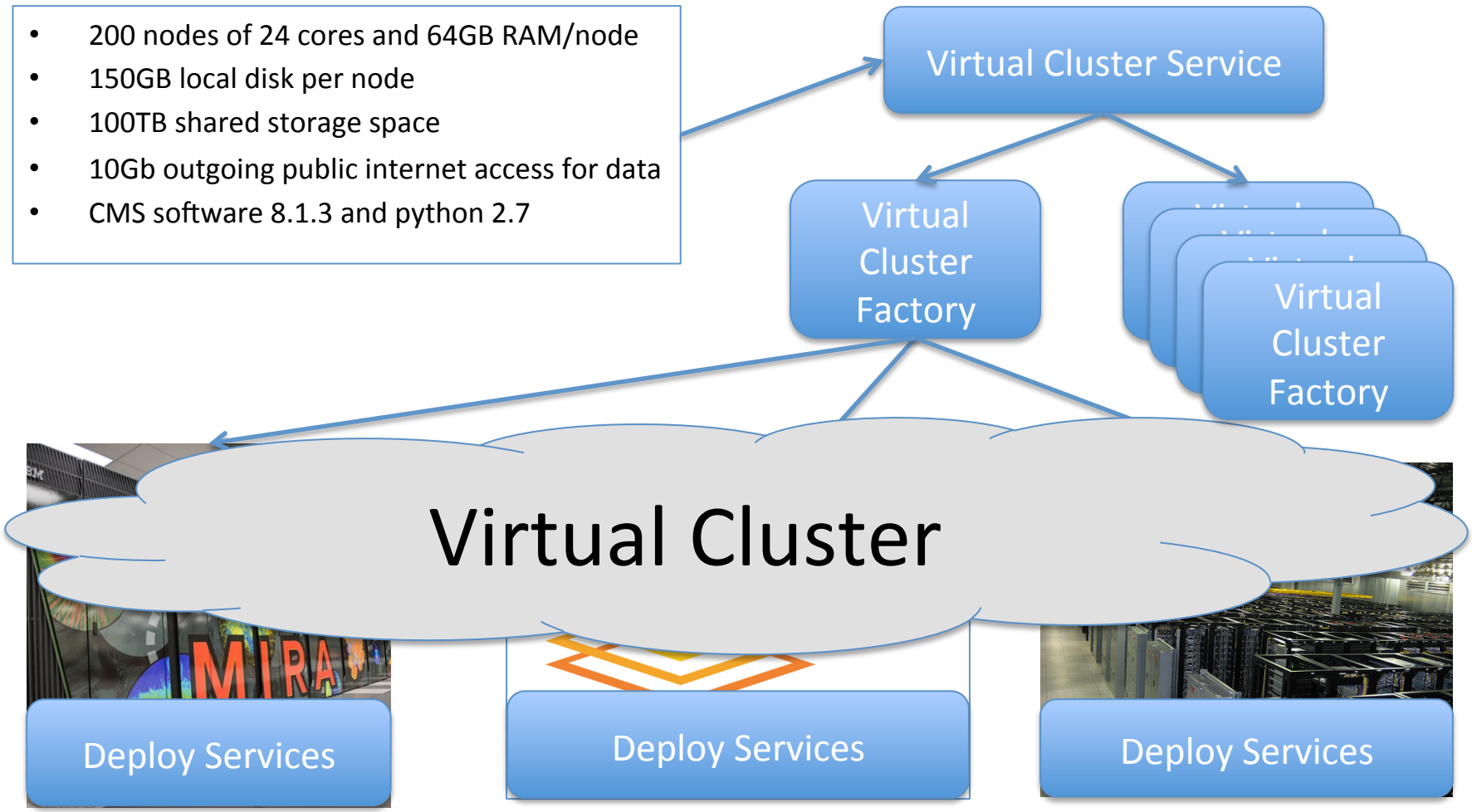
Traditional HPC Facility

Distributed HTC Facility

Commercial Cloud

Concept: Virtual Cluster

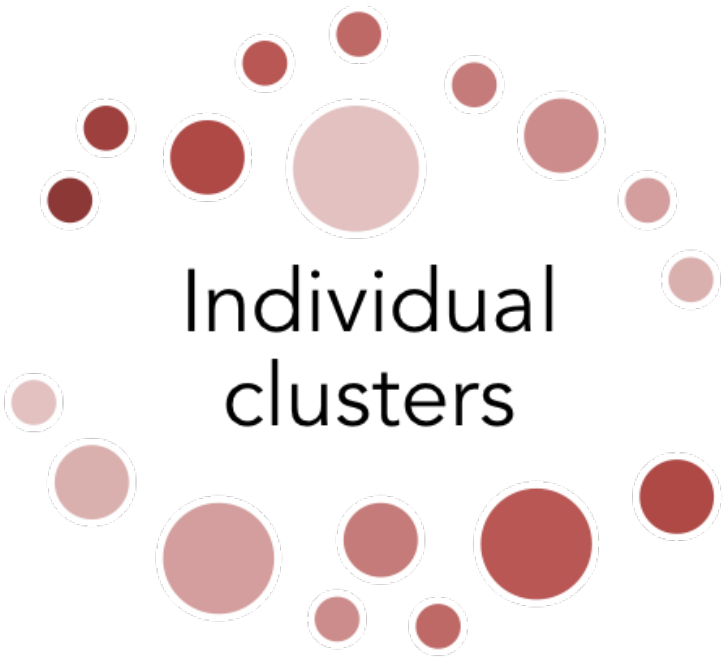
- 200 nodes of 24 cores and 64GB RAM/node
- 150GB local disk per node
- 100TB shared storage space
- 10Gb outgoing public internet access for data
- CMS software 8.1.3 and python 2.7



Traditional HPC Facility

Distributed HTC Facility

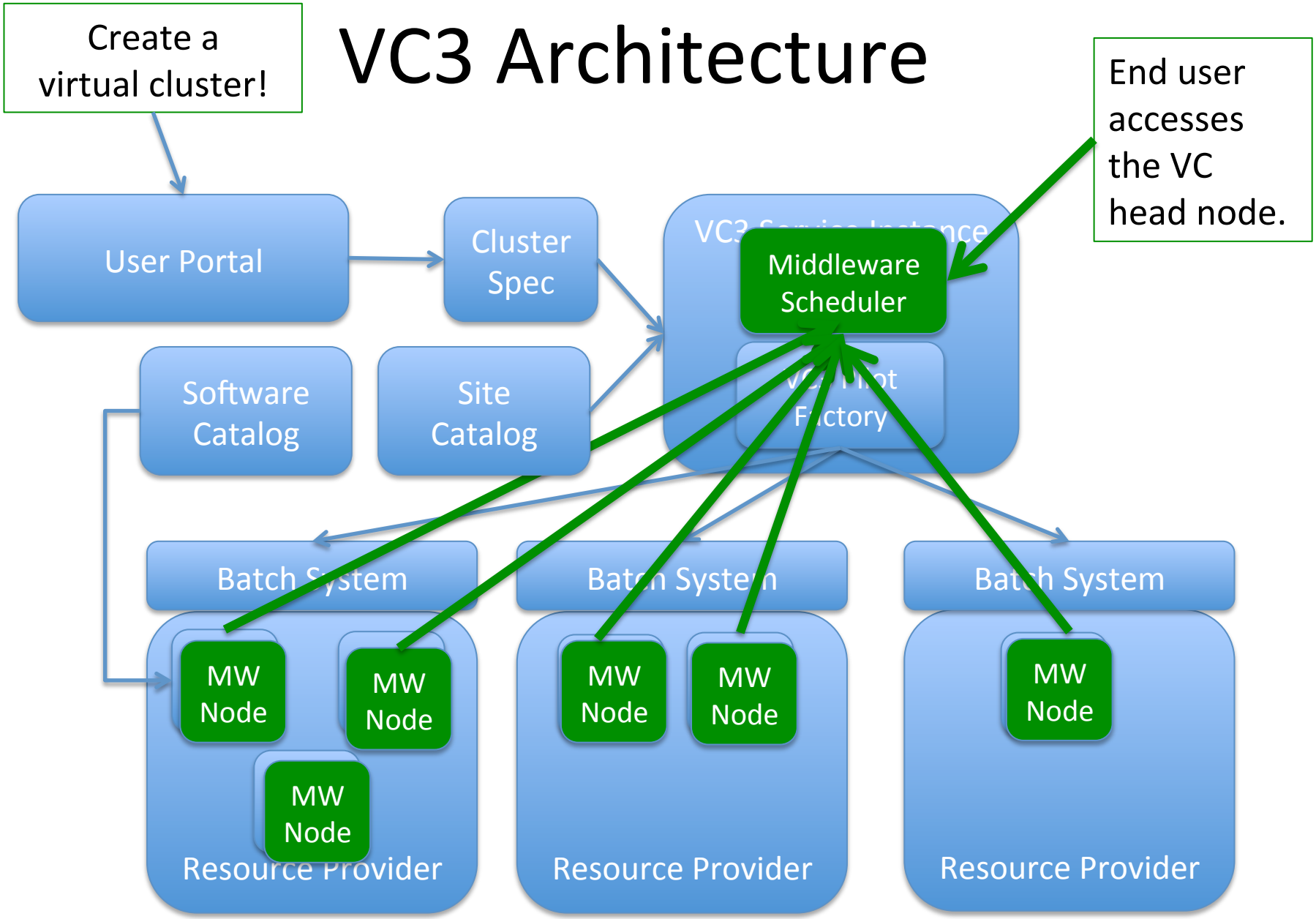
Commercial Cloud



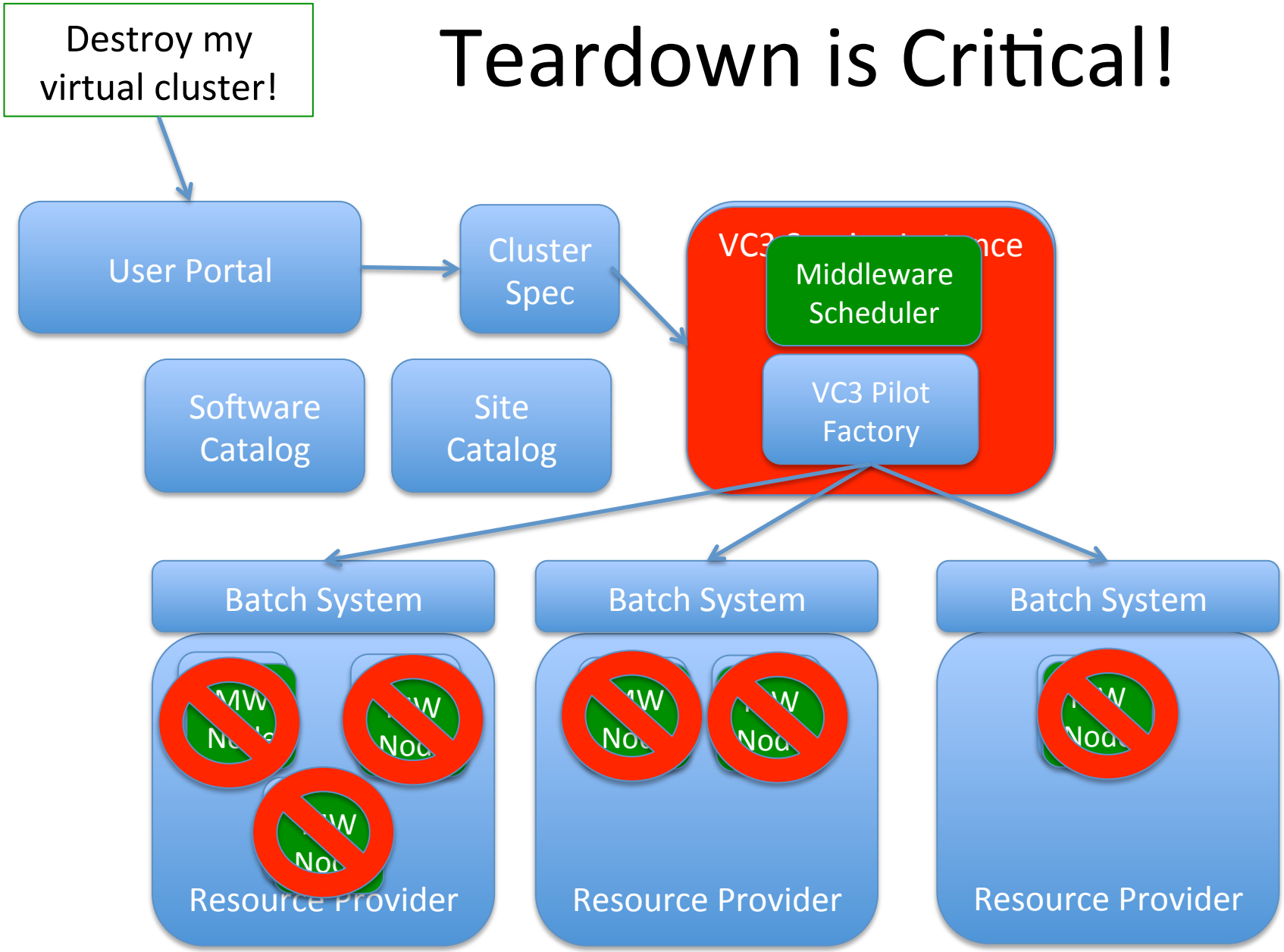
Distributed resources: diverse & complex

One environment: uniform & simple

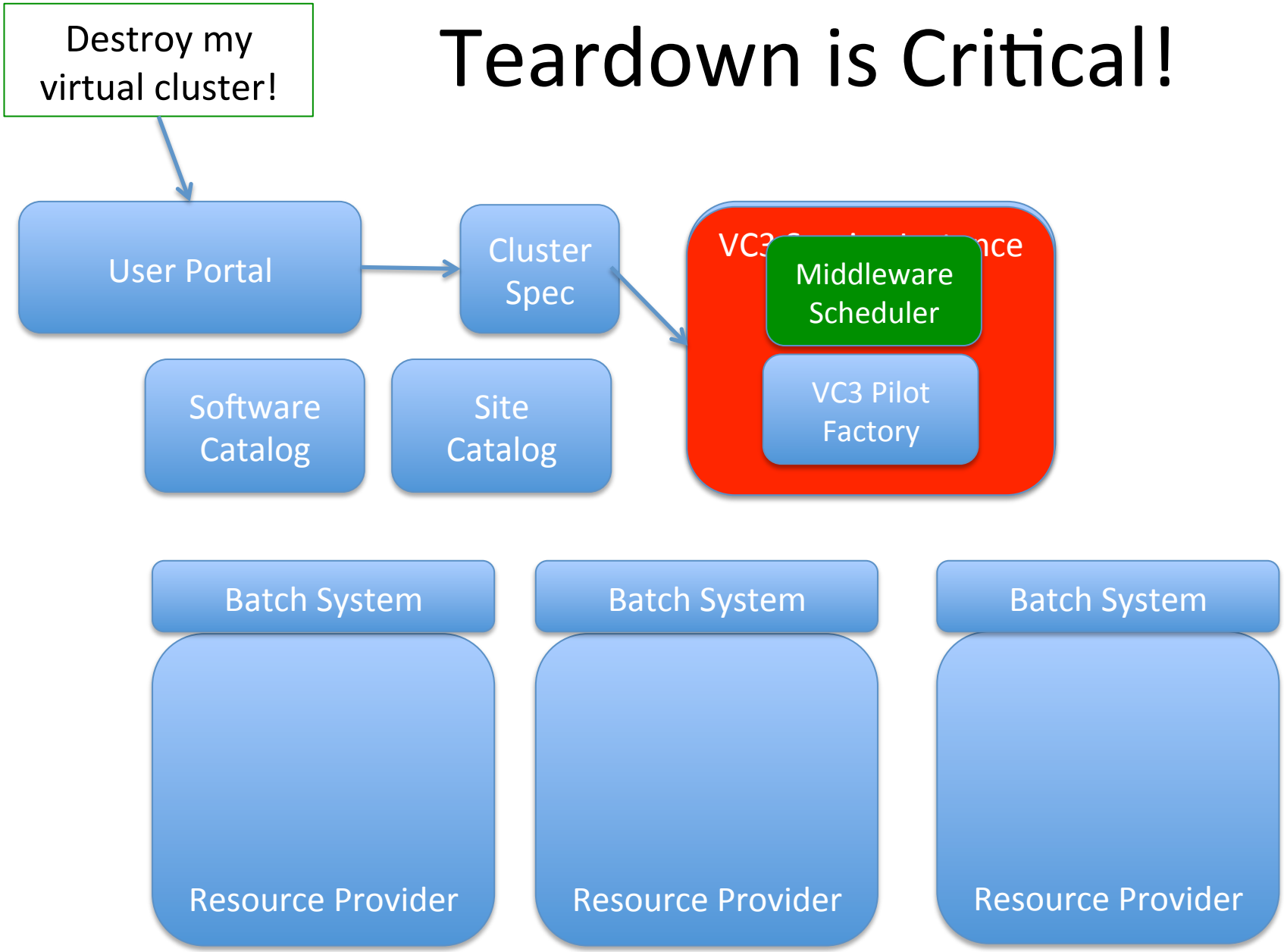
VC3 Architecture



Teardown is Critical!



Teardown is Critical!



Outline

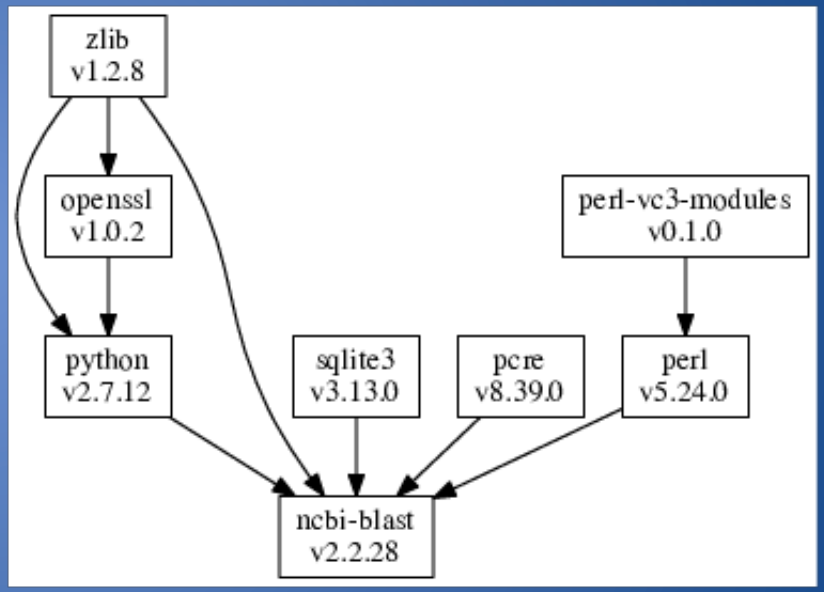
- The Laptop Perspective
- Scaling Up with Makeflow and Work Queue
- VC3: Virtual Clusters
- Problem: Software Deployment
- Problem: Resource Sizing
- Lessons Learned

Problem: Software Deployment

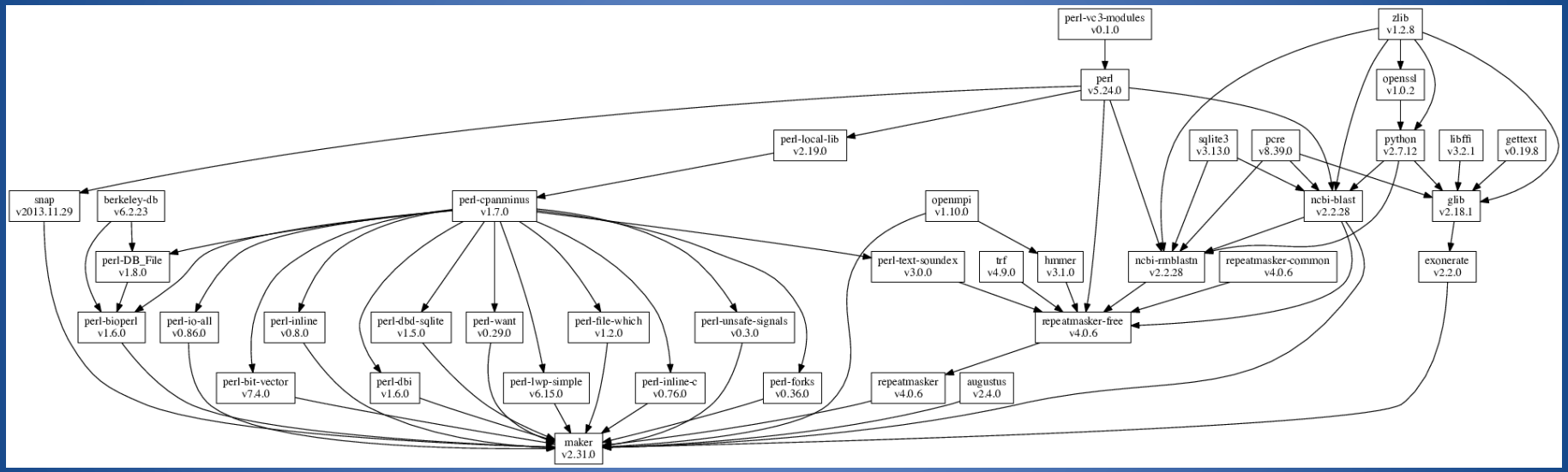
- Getting software installed on a new site is a big pain! The user (probably) knows the top level package, but doesn't know:
  - How they set up the package (sometime last year)
  - Dependencies of the top-level package.
  - Which packages are system default vs optional
  - How to import the package 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

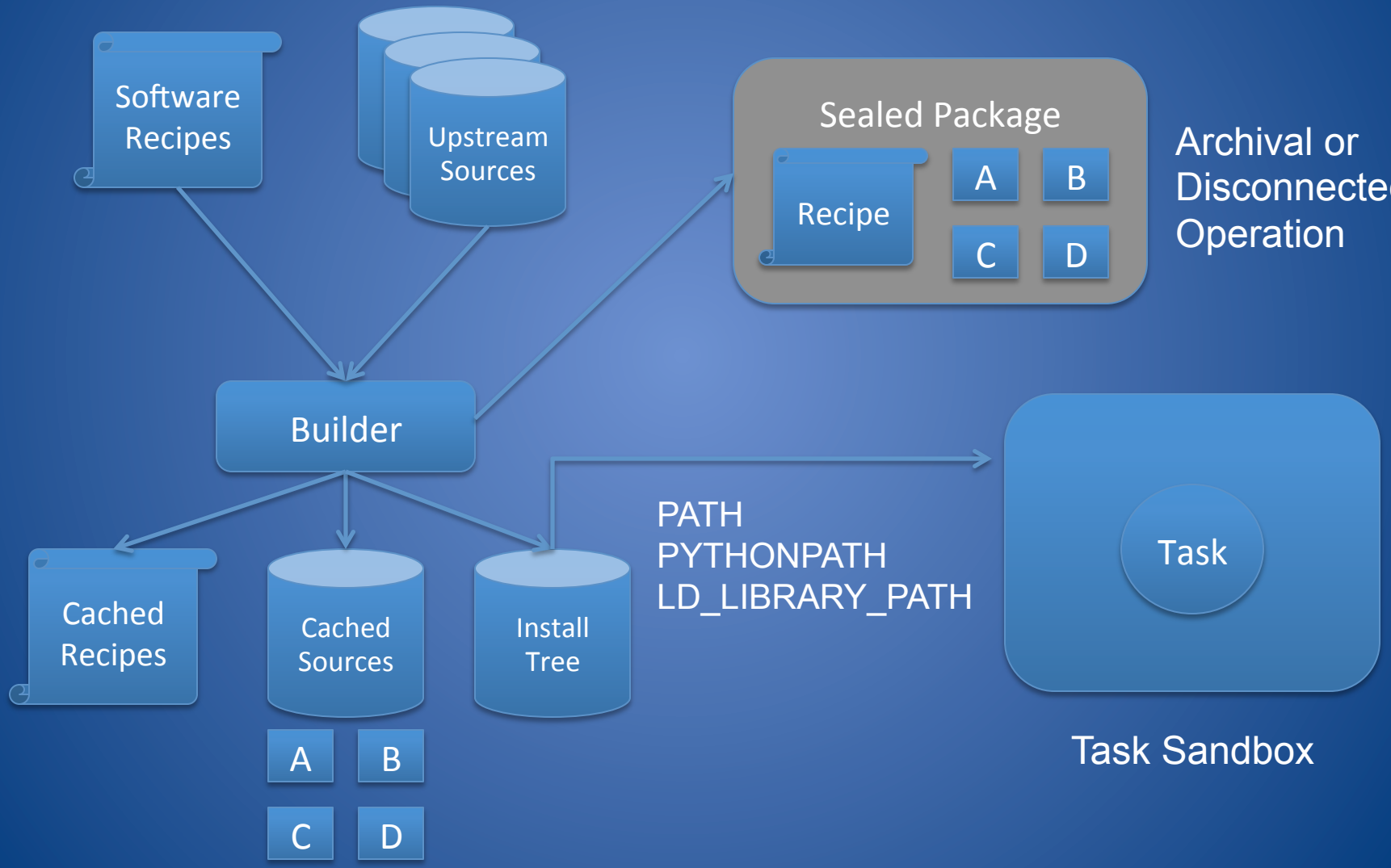
"I just need BLAST."
"Oh wait, I need Python!"
"Sorry, Python 2.7.12"
"Python requires SSL?"
"What on earth is pcre?"
"I give up!"



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: pyt
....could not
....Try: pyt
......Plan: c
……
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

(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
```

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

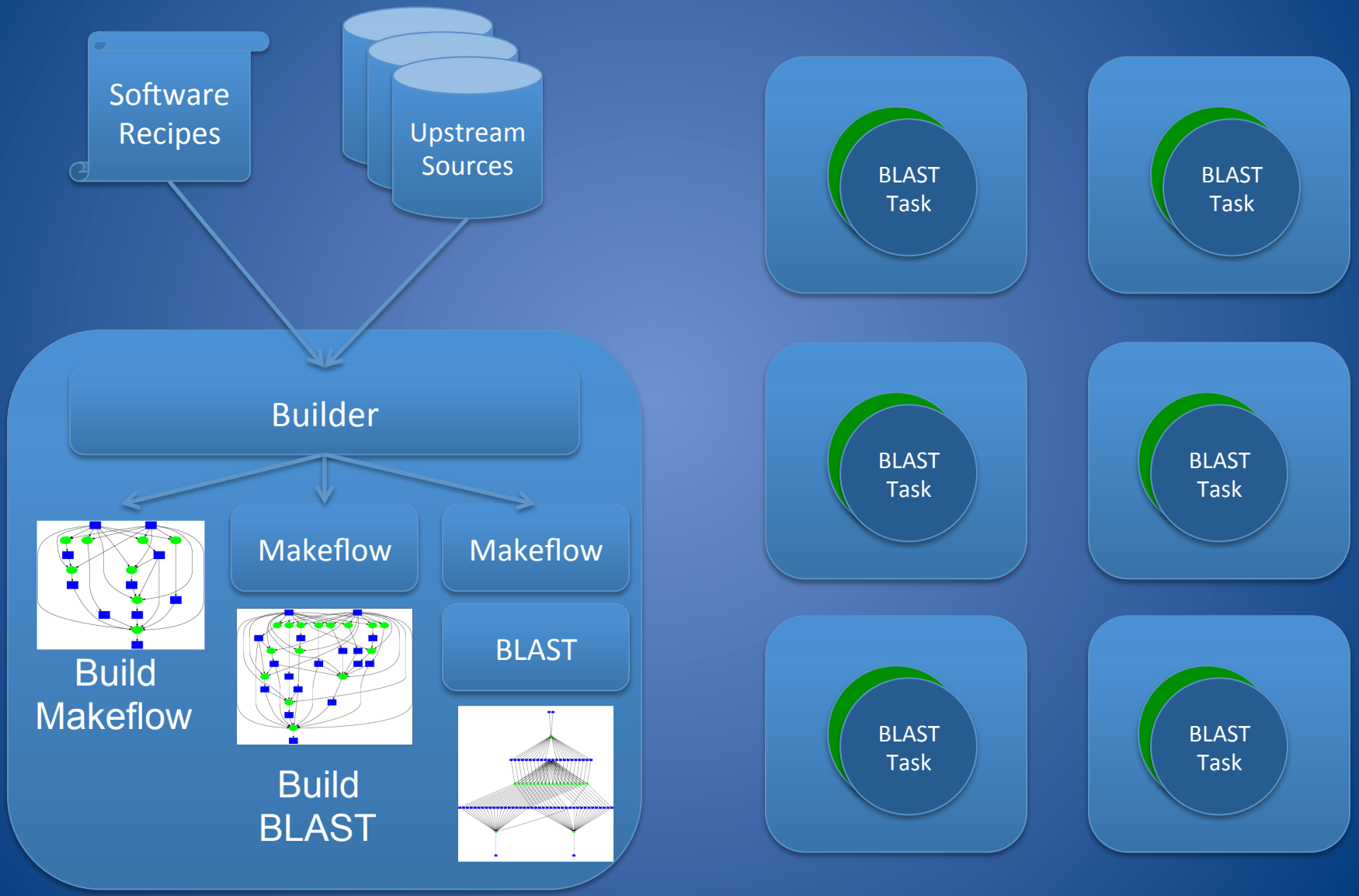
--require makeflow

--require ncbi-blast

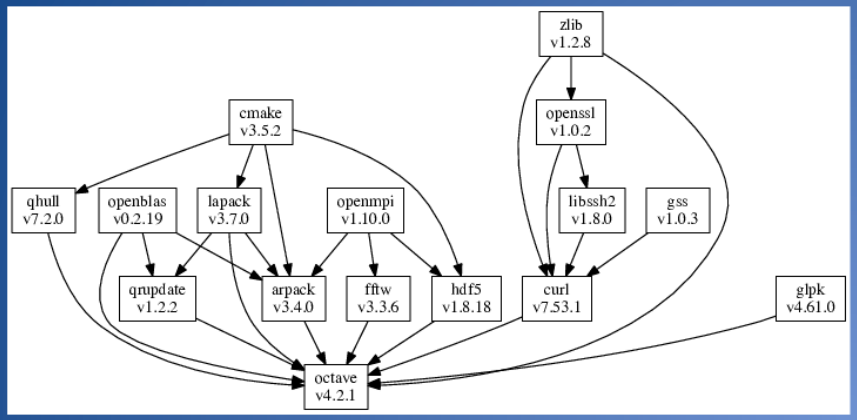
--

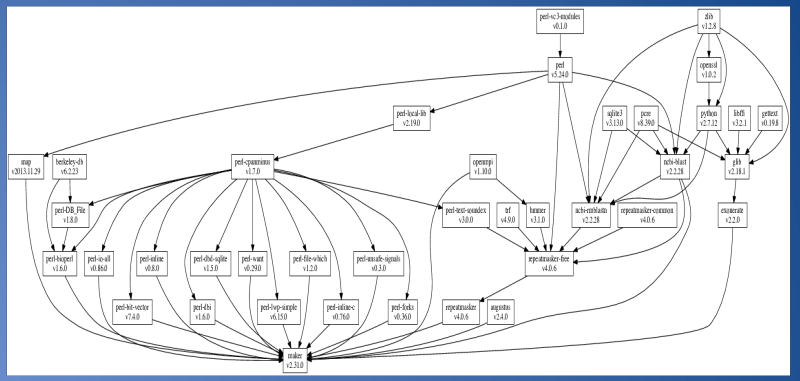
makeflow –T condor blast.mf

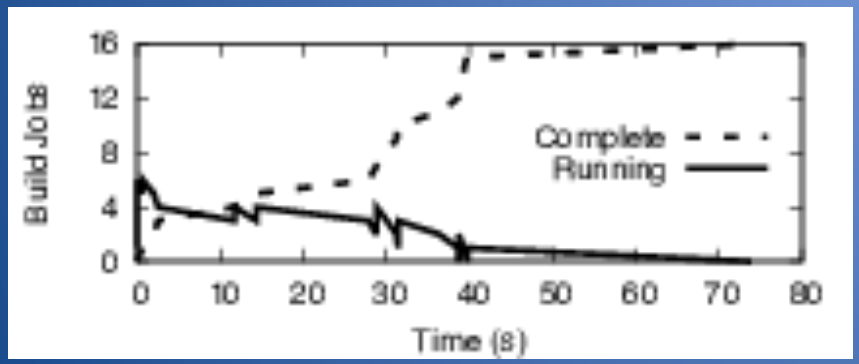
Bootstrapping a Workflow

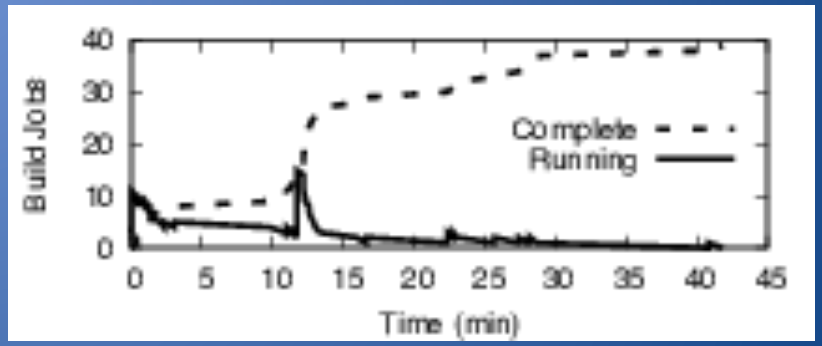


Example Applications









Octave

MAKER

Typically, 2-3x faster overall. But more importantly, filesystem-intensive jobs run on the cluster resources, not on the head node!

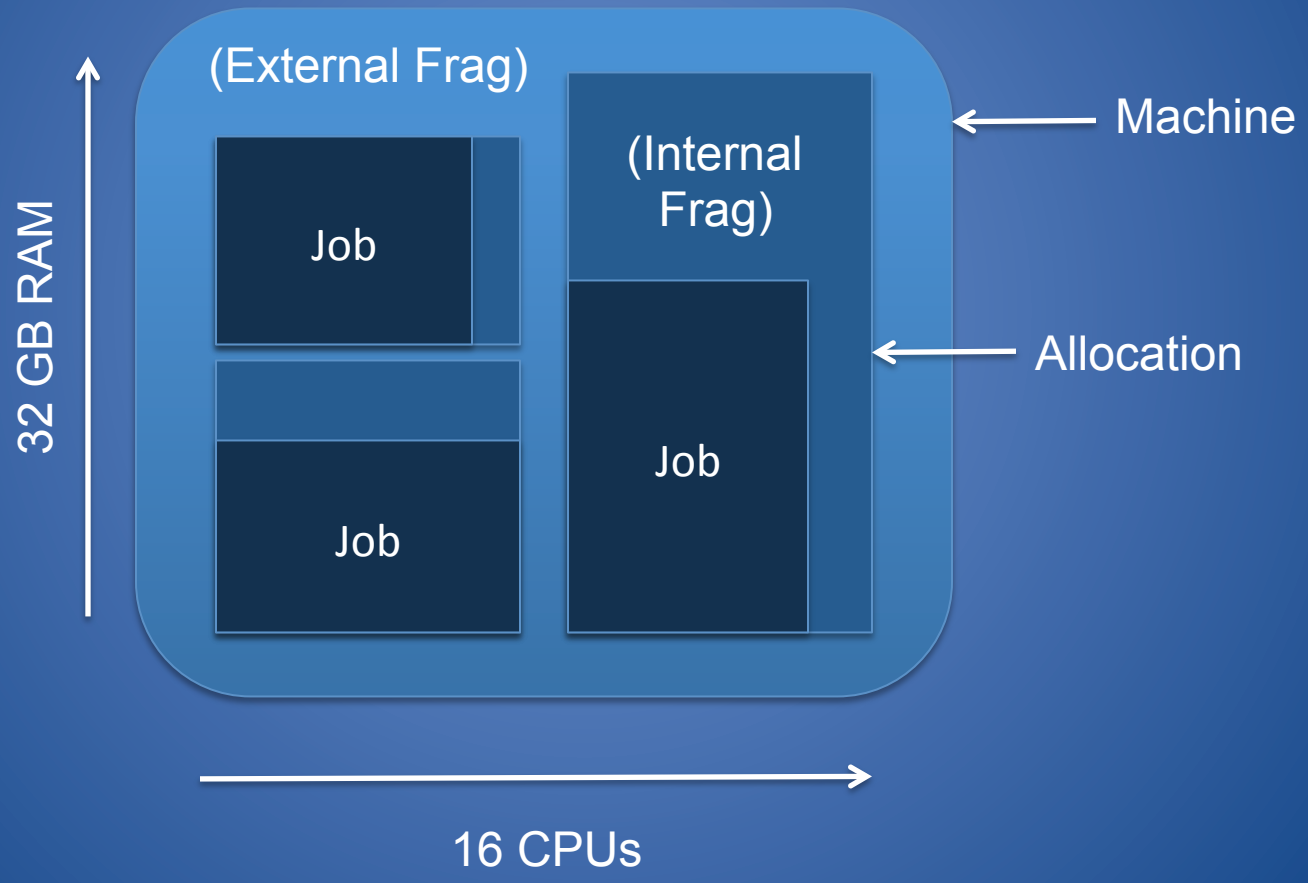
Outline

- The Laptop Perspective
- Scaling Up with Makeflow and Work Queue
- VC3: Virtual Clusters
- Problem: Software Deployment
- Problem: Resource Sizing
- Lessons Learned

Problem: Resource Selection

- Common misconceptions:
  - Users know what resources their jobs need.
    - "I need 4GB RAM, 8 cores, and 1TB disk."
  - Jobs in the same batch are usually the same.
- In reality:
  - Users have no idea what they jobs actually need!
    - "Well, it runs on my laptop, what does that have?"
  - Jobs in the same batch can have a very complex distribution of resource needs.

The Resource Sizing Problem



Client vs. Resource Provider

Client selects allocation:

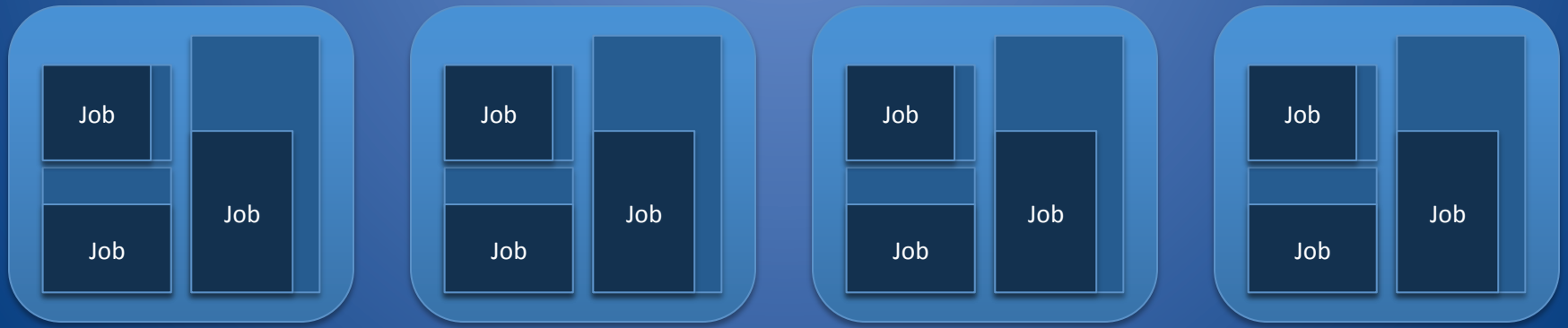
Too big? Wasted resources.
Too small? Job fails, retry.

Provider places the allocation:

Too big? Get paid.
Too small? Still get paid!
Scheduling is not the client's problem.

Client

Provider



"Slicing the Infinite Cake"

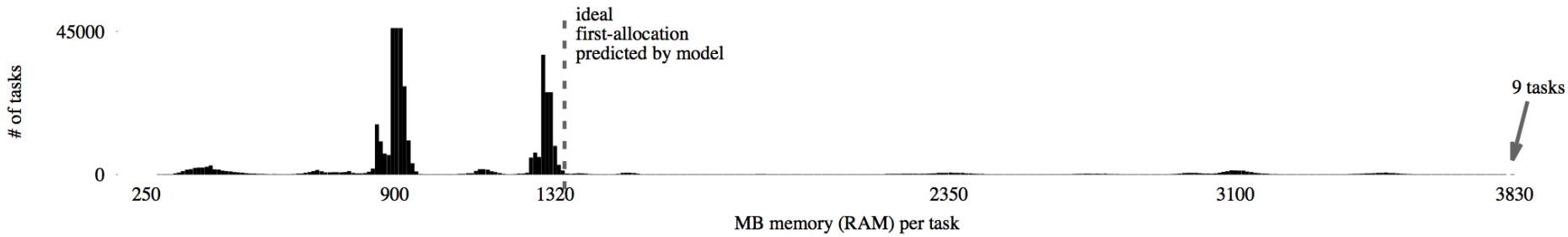
Allocations Too Big

Allocations Just Right: 2x Throughput

How do we know how big?


```
"command": "python task.py parameters.json",
"taskid":            "195",
"user":              "mfrohike",
"category":          "ttZ_mAODv2",
"executable_type":   "dynamic",
"monitor_version":   "6.0.0.0bdd9ca9",
"exit_status":       143,
"exit_type":         "limits",
"limits_exceeded":   { "memory": [2000,"MB"] },
"start":             [1454163721,"s"],
"end":               [1454165828,"s"],
"wall_time":         [2107,"s"],
"cpu_time":          [2066,"s"],
"cores":             2,
"cores_avg":         0.98,
"concurrent_procs":  8,
"total_procs":       273,
"virtual_memory":    [2255,"MB"],
"memory":            [2142,"MB"],
"swap_memory":       [0,"MB"],
"bytes_read":        [2274265095,"MB"],
"bytes_written":     [104022016,"MB"],
"bytes_sent":        [0,"MB"],
"bytes_received":    [0,"MB"],
"bandwidth":         [0,"Mbps"],
"total_files":       1175,
"disk":              [104,"MB"]
```

Surprise: Complex Distributions!



How to pick the first allocation?

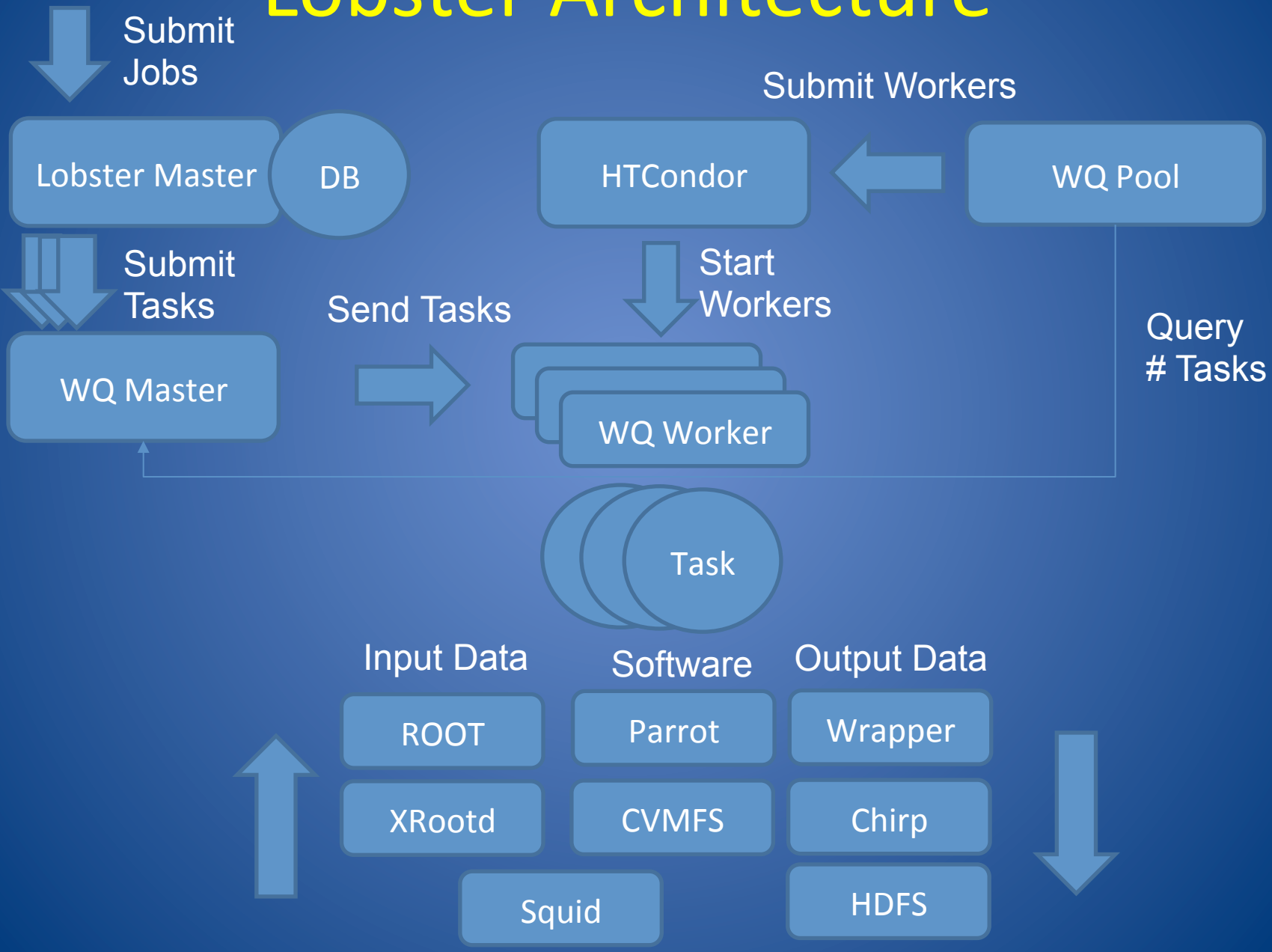
Ben Tovar says:
Minimize probability of first attempt succeeding + fallback succeeding, weighted by resources.

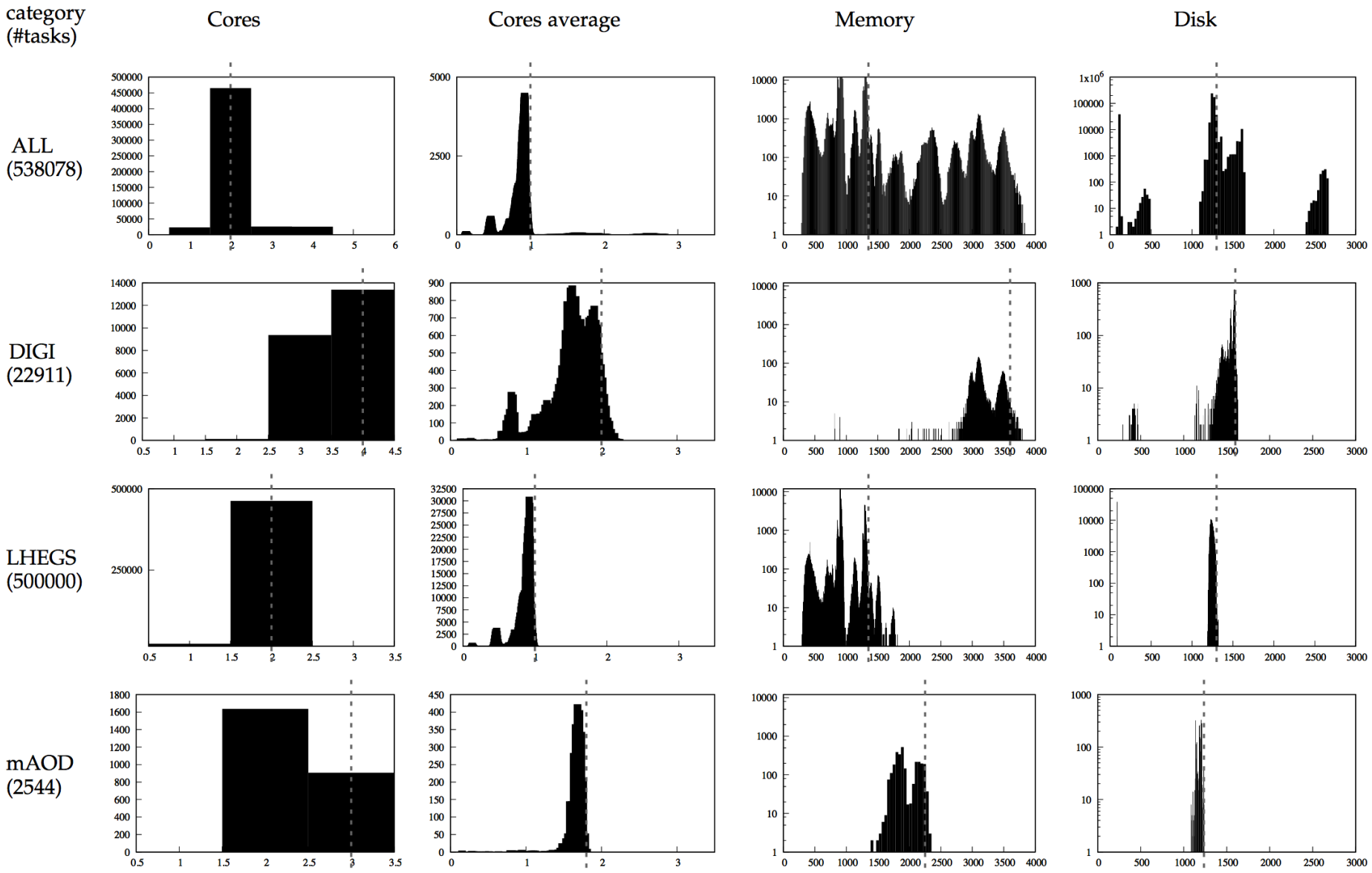
$$
\begin{aligned}
\mathrm{E}[\mathrm{waste}(\mathrm{r},\tau,a_1)] = \int_0^\infty \Bigg(& \underbrace{\int_0^{a_1} (a_1 - \mathrm{r})\tau p(\mathrm{r},\tau)\mathrm{d}\mathrm{r}}_{\text{first-allocation succeds}} \\
& + \underbrace{\int_{a_1}^{a_m} ((a_m + a_1 - \mathrm{r})\tau p(\mathrm{r},\tau)\mathrm{d}\mathrm{r}}_{\text{final allocation succeds}} \Bigg) \mathrm{d}\tau \\
= & a_1 \underbrace{\int_{a_1}^{a_m} \int_0^\infty \tau p(\mathrm{r},\tau)\mathrm{d}\tau\mathrm{d}\mathrm{r}}_{\text{mean wall-time for all tasks}} \\
& + a_m \int_{a_1}^{a_m} \underbrace{\int_0^\infty \tau p(\tau|\mathrm{r})\mathrm{d}\tau}_{\text{mean wall-time taks w. peak r}} \quad p(\mathrm{r})\mathrm{d}\mathrm{r} \\
& - \underbrace{\int_0^\infty \int_0^\infty \mathrm{r}\tau p(\mathrm{r},\tau)\mathrm{d}\tau\mathrm{d}\mathrm{r}}_{\text{used resources}},
\end{aligned}
$$

Production Application: Lobster

- Lobster: High energy physics analysis workload harnesses heterogeneous non-dedicated resources at Notre Dame.
- 535,078 tasks run on 25,000 core cluster over several months with the resource monitor.
- Five categories of tasks identified by user: DIGI (22911), LHEGS(500K), mAOD (2544), RECO (11582)

Lobster Architecture




category (#tasks)
Cores
Cores average
Memory
Disk
ALL (538078)
DIGI (22911)
LHEGS (500000)
mAOD (2544)

Resource Selection Approaches

	naive		brute-force		min. waste	max. through
resource	max. peak	$P(0.95 > r)$	min. waste	max. throug.	Equation 2	Equation 3
first allocation						
cores (cores)	5	3	2	2	2	2
cores_avg (cores)	2.9	1.5	1	1	1	1
memory (MB)	3830	2416	1350	1350	1350	1350
disk (MB)	2657	1338	1300	1300	1300	1300
proportion of wasted resources per task						
cores	58%	34%	13%	13%	13%	13%
cores_avg	70%	48%	23%	23%	23%	23%
memory	72%	57%	32%	32%	32%	32%
disk	55%	16%	15%	15%	15%	15%
throughput normalized						
cores	1.00	1.58	2.18	2.18	2.18	2.18
cores_avg	1.00	1.74	2.69	2.69	2.69	2.69
memory	1.00	1.51	2.54	2.54	2.54	2.54
disk	1.00	1.88	1.91	1.91	1.91	1.91
percentage of tasks retried						
cores	0%	5%	9%	9%	9%	9%
cores_avg	0%	5%	7%	7%	7%	7%
memory	0%	5%	8%	8%	8%	8%
disk	0%	5%	6%	6%	6%	6%
overhead						
overhead (s)	—	—	0.78	0.83	0.07	0.06
538078 tasks read in 27.60 seconds						

What's the upshot?

- By selecting first allocations appropriately, we double the throughput of the system while accepting a 9 percent task failure rate.
- This approach is applied entirely from the client side, without provider assistance.
- Same approach can be applied to any cluster/ cloud/grid with simple techniques.

Outline

- The Laptop Perspective
- Scaling Up with Makeflow and Work Queue
- VC3: Virtual Clusters
- Problem: Software Deployment
- Problem: Resource Sizing
- Lessons Learned

Thoughts and Lessons Learned

- Make software dependencies explicit!
  - Proposed: Nothing should be available by default, all software should require an "import" step.
- Make resource consumption more visible!
  - The laconic nature of the shell hides too much about resource consumption.
- Model observed behavior, but have a fallback when the model fails.
  - Example: first allocation based on previous behavior falls back to maximum task/machine size.

Acknowledgements

People in the Cooperative Computing Lab

Douglas Thain
Director

Benjamin Tovar
Research Soft. Engineer

Peter Ivie

Nicholas Hazekamp

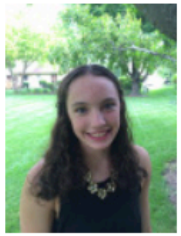
Charles Zheng

Nathaniel Kremer-Herman

Tim Shaffer

Kyle Sweeney

James Sweet



Caitlin Guccione

Cameron Lamir

Apply Today!

DE-SC0015711
VC3: Virtual Clusters for Community Computation

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

ccl.cse.nd.edu

The Cooperative Computing Lab

Software | Download | Manuals | Papers

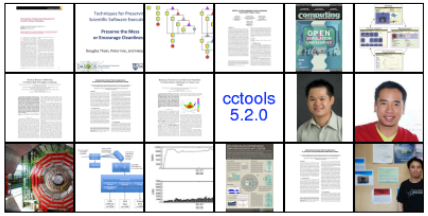
Take the ACIC 2015 Tutorial on Makeflow and Work Queue

About the CCL

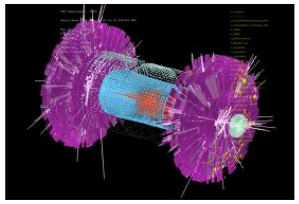
We design software that enables our collaborators to easily harness large scale distributed systems such as clusters, clouds, and grids. We perform fundamental computer science research in that enables new discoveries through computing in fields such as physics, chemistry, bioinformatics, biometrics, and data mining.

CCL News and Blog

- Global Filesystems Paper in IEEE CiSE *(09 Nov 2015)*
- Preservation Talk at iPres 2015 *(03 Nov 2015)*
- CMS Case Study Paper at CHEP *(20 Oct 2015)*
- OpenMalaria Preservation with Umbrella *(19 Oct 2015)*
- DAGVz Paper at Visual Performance Analysis Workshop *(13 Oct 2015)*
- Virtual Wind Tunnel in IEEE CiSE *(09 Sep 2015)*
- Three Papers at IEEE Cluster in Chicago *(07 Sep 2015)*
- CCTools 5.2.0 released *(19 Aug 2015)*
- Recent CCL Grads Take Faculty Positions *(18 Aug 2015)*
- (more news)



Community Highlight



Scientists searching for the Higgs boson have profited from Parrot's new support for the CernVM Filesystem (CVMFS), a network filesystem tailored to providing world-wide access to software installations. By using Parrot, CVMFS, and additional components integrated by the Any Data, Anytime, Anywhere project, physicists working in the Compact Muon Solenoid experiment have been able to create a uniform computing environment acro
Instead of maintaining large software
participating institution, Parrot is use
highly-available CVMFS installation
files are downloaded as needed and a
efficiency. A pilot project at the Univ
demonstrated the feasibility of this ap
compute jobs to run in the Open Scie
harnessing 370,000 CPU-hours acros
access to 400 gigabytes of software i
repository.

- Dan Bradley, University of Wiscons

http://ccl.cse.nd.edu

Twitter, Inc.

Search Twitter | Have an account? Log in

TWEETS 28 | FOLLOWING 52 | FOLLOWERS 35 | LIKES 8 | Follow

Douglas Thain
@ProfThain

Distributed computing for big data problems in science and engineering.

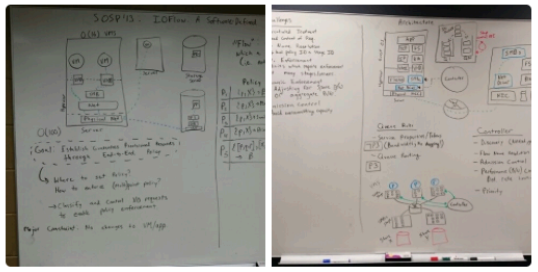
Notre Dame, IN
nd.edu/~dthain

13 Photos and videos

Tweets | Tweets & replies | Photos & videos

Douglas Thain @ProfThain · Nov 10

My grad students now summarize research papers by preparing a whiteboard in advance. Much better than a slide deck!



1 | 4

New to Twitter?

@ProfThain