Introduction to Makeflow and Work Queue

CSE 40822 – Cloud Computing – Spring 2016

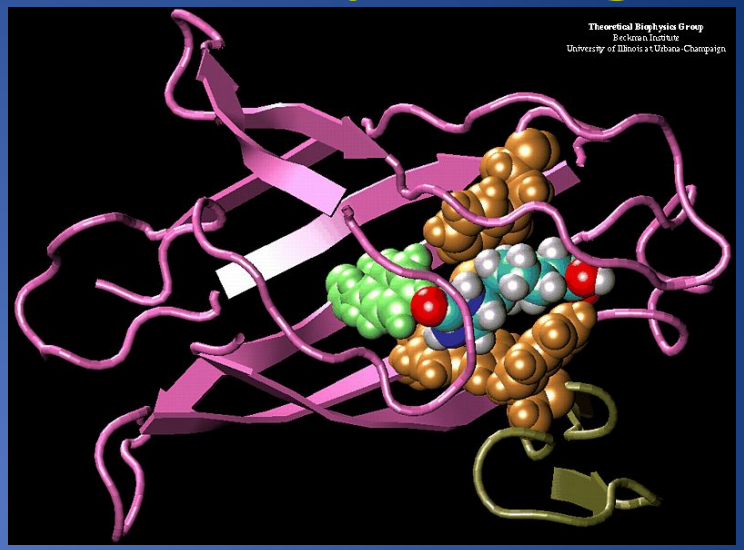
Douglas Thain / Ben Tovar

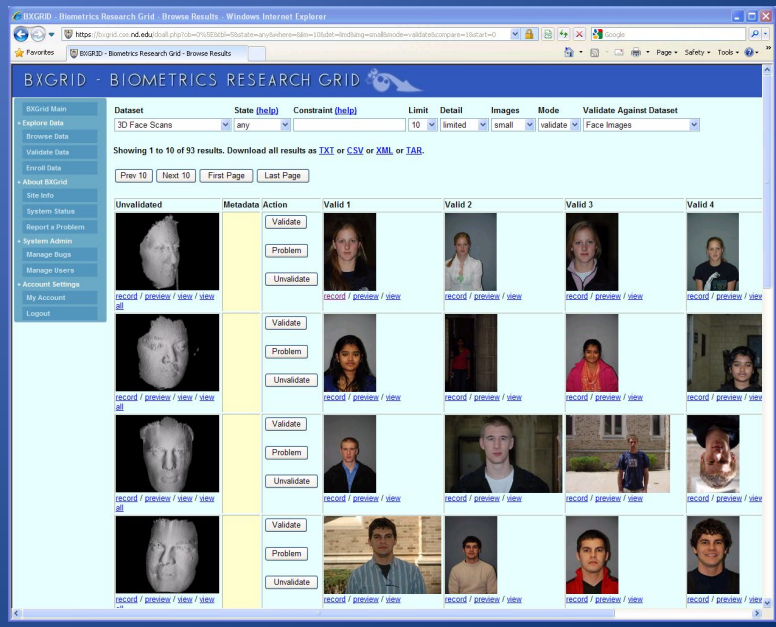
The Cooperative Computing Lab

- 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://www.nd.edu/~ccl

Science Depends on Computing!





AGTCCGTACGATGCTATTAGCGAGCGTGA

...

The Good News:
Computing is Plentiful!

Future Grid

Eucalyptus Systems

amazon web services™

CPU Utilization for the Last Month

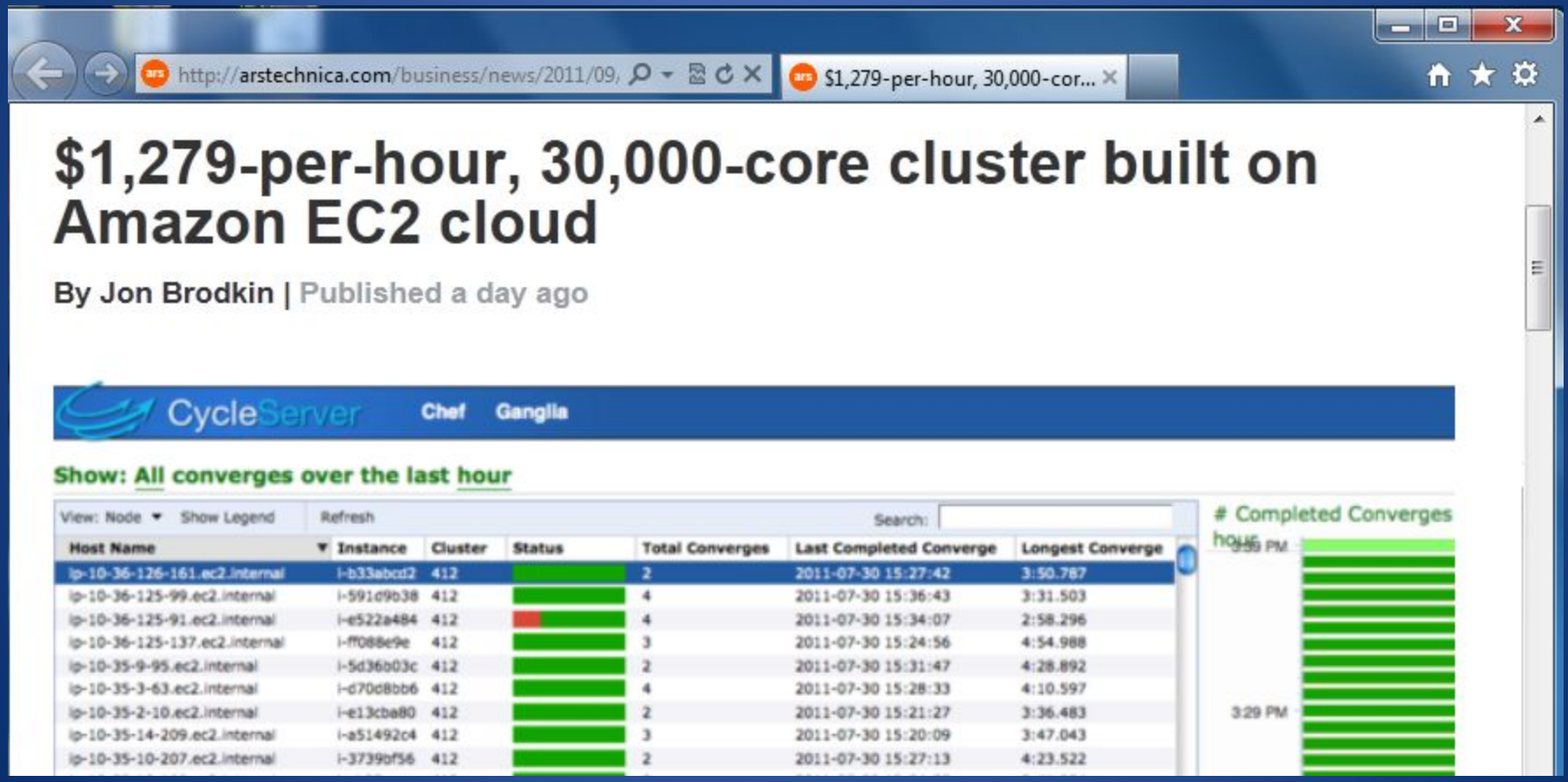
XSEDE
Extreme Science and Engineering Discovery Environment

Windows Azure

NIMBUS

365688
466637
68086 (7%) CPU-Hours Used by Owner
900411 (100%) CPU-Hours Total

Big clusters are cheap!



http://arstechnica.com/business/news/2011/09/30000-core-cluster-built-on-amazon-ec2-cloud.ars

I have a standard, debugged, trusted application that runs on my laptop.

A toy problem completes in one hour.
A real problem will take a month (I think.)

Can I get a single result faster?
Can I get more results in the same time?

Last year,
I heard about
this grid thing.

This year,
I heard about
this cloud thing.

What do I do next?

What they want.

What they get.

I can get as many machines
on the cloud as I want!

How do I organize my application
to run on those machines?

Our Philosophy:

- Harness all the resources that are available: desktops, clusters, clouds, and grids.
- Make it easy to scale up from one desktop to national scale infrastructure.
- Provide familiar interfaces that make it easy to connect existing apps together.
- Allow portability across operating systems, storage systems, middleware…
- Make simple things easy, and complex things possible.
- ***No special privileges required.***

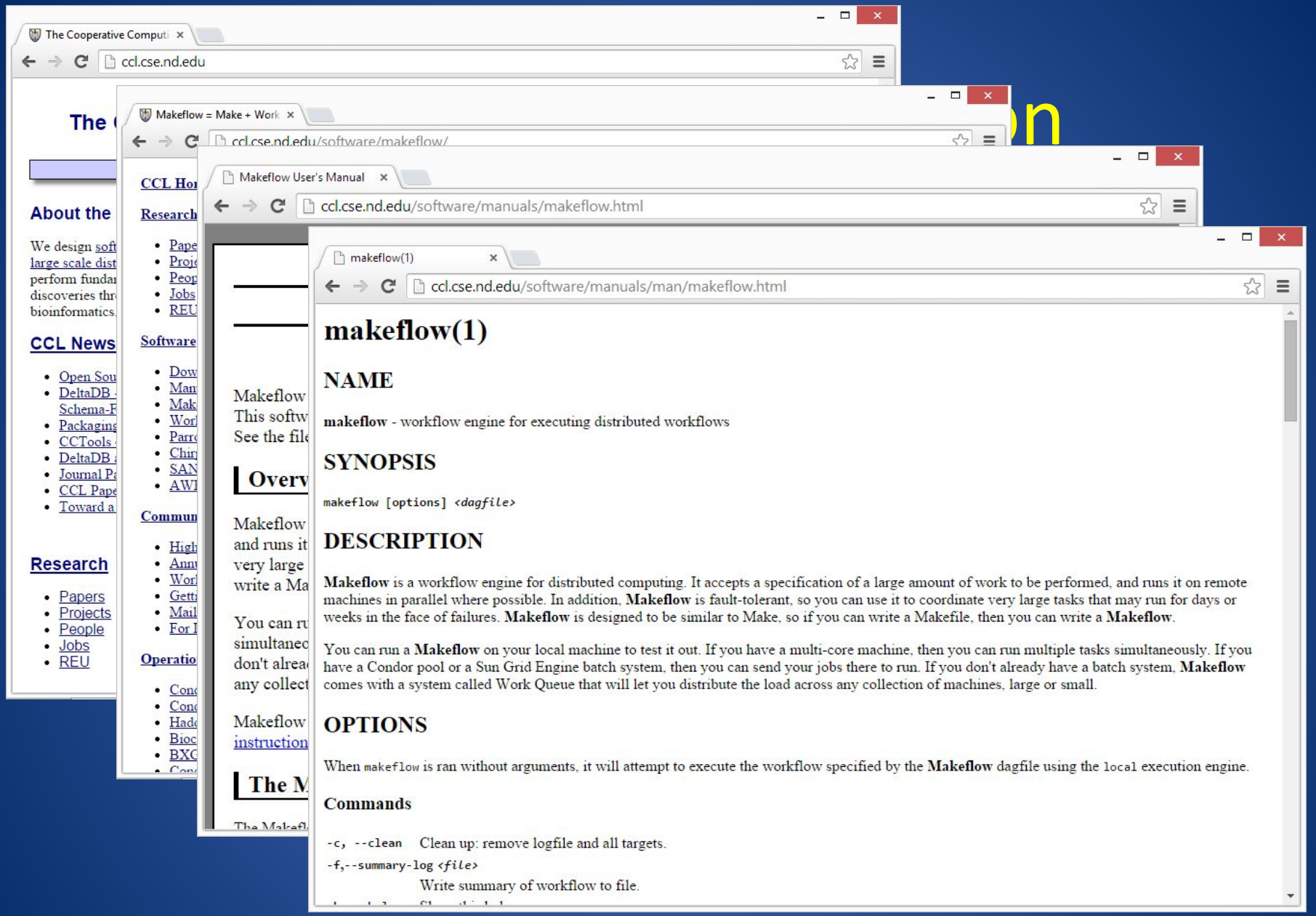
A Quick Tour of the CCTools

- Open source, GNU General Public License.
- Compiles in 1-2 minutes, installs in $HOME.
- Runs on Linux, Solaris, MacOS, Cygwin, FreeBSD, …
- Interoperates with many distributed computing systems.
  - Condor, SGE, Torque, Globus, iRODS, Hadoop…
- Components:
  - Makeflow – A portable workflow manager.
  - Work Queue – A lightweight distributed execution system.
  - All-Pairs / Wavefront / SAND – Specialized execution engines.
  - Parrot – A personal user-level virtual file system.
  - Chirp – A user-level distributed filesystem.

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

Install in Your Home Directory

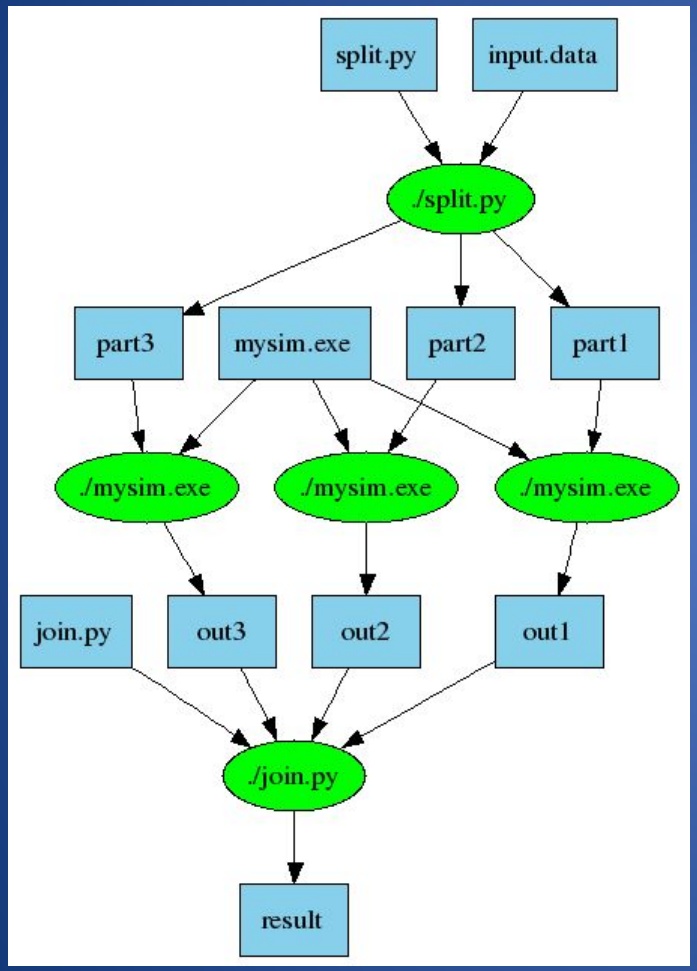
```
cd $HOME
wget http://ccl.cse.nd.edu/software/files/
                                cctools-4.2.2-source.tar.gz
tar xvzf cctools-4.2.2-source.tar.gz
cd cctools-4.2.2-source
./configure
make
make install
```



http://ccl.cse.nd.edu

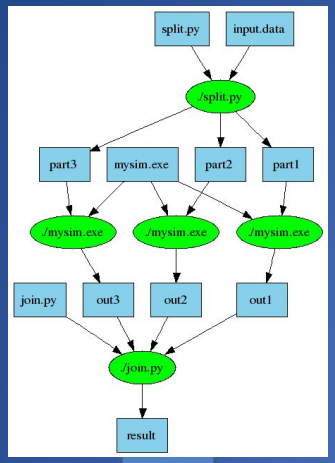
Makeflow:
A Portable Workflow System

An Old Idea: Makefiles




```
part1 part2 part3: input.data split.
py
    ./split.py input.data

out1: part1 mysim.exe
    ./mysim.exe part1 >out1

out2: part2 mysim.exe
    ./mysim.exe part2 >out2

out3: part3 mysim.exe
    ./mysim.exe part3 >out3

result: out1 out2 out3 join.py
    ./join.py out1 out2 out3 > result
```

Makeflow = Make + Workflow



- Provides portability across batch systems.
- Enable parallelism (but not too much!)
- Trickle out work to batch system.
- Fault tolerance at multiple scales.
- Data and resource management.

Makeflow

Local | Condor | Torque | Work Queue

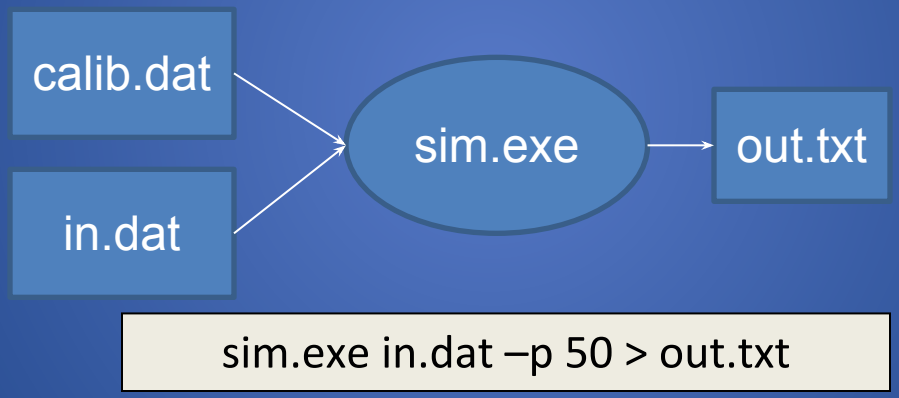
http://ccl.cse.nd.edu/software/makeflow

Makeflow Syntax

[output files] : [input files]

[command to run]

One Rule



out.txt : in.dat calib.dat sim.exe

sim.exe –p 50 in.data > out.txt


You must state
all the files
needed by the command.

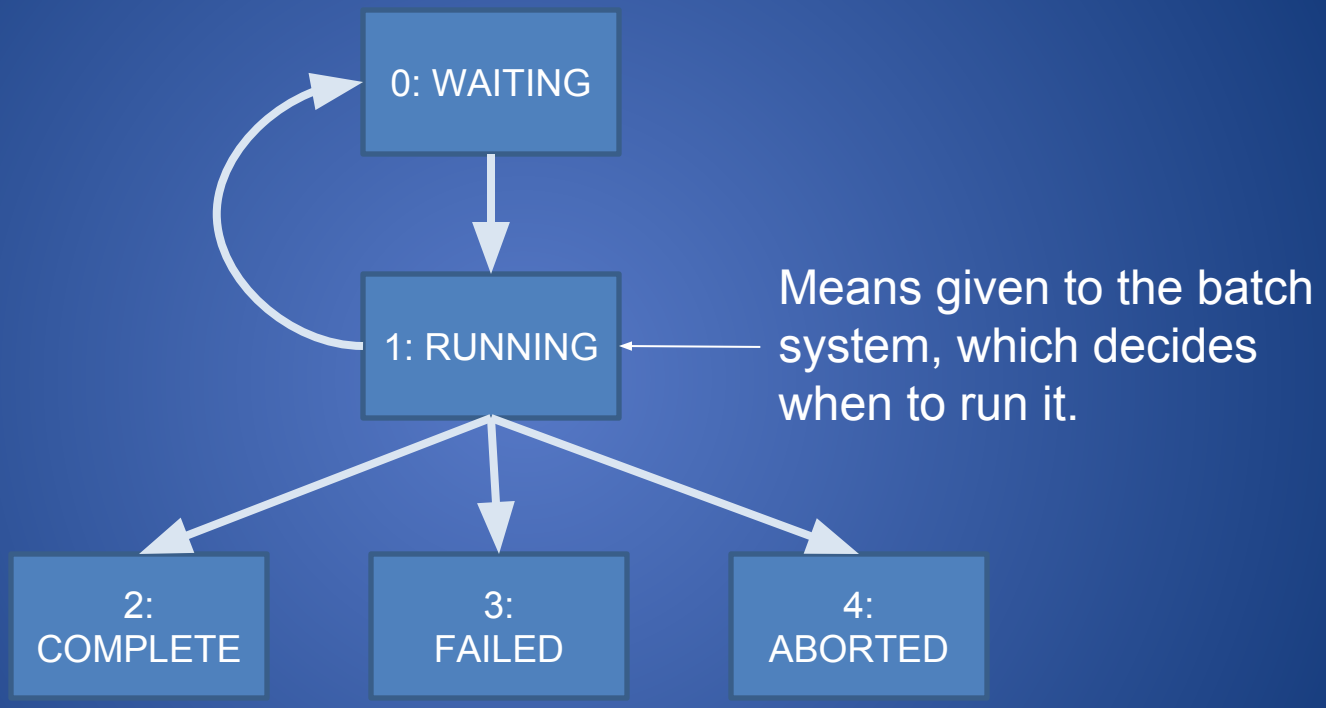
sims.mf

```
out.10 : in.dat calib.dat sim.exe
        sim.exe –p 10 in.data > out.10

out.20 : in.dat calib.dat sim.exe
        sim.exe –p 20 in.data > out.20

out.30 : in.dat calib.dat sim.exe
        sim.exe –p 30 in.data > out.30
```

How to run a Makeflow

- Run a workflow locally (multicore?)
  - makeflow -T local sims.mf
- Clean up the workflow outputs:
  - makeflow –c sims.mf
- Run the workflow on Torque:
  - makeflow –T torque sims.mf
- Run the workflow on Condor:
  - makeflow –T condor sims.mf

Job States


0: WAITING
1: RUNNING
2:
COMPLETE
3:
FAILED
4:
ABORTED
Means given to the batch system, which decides when to run it.

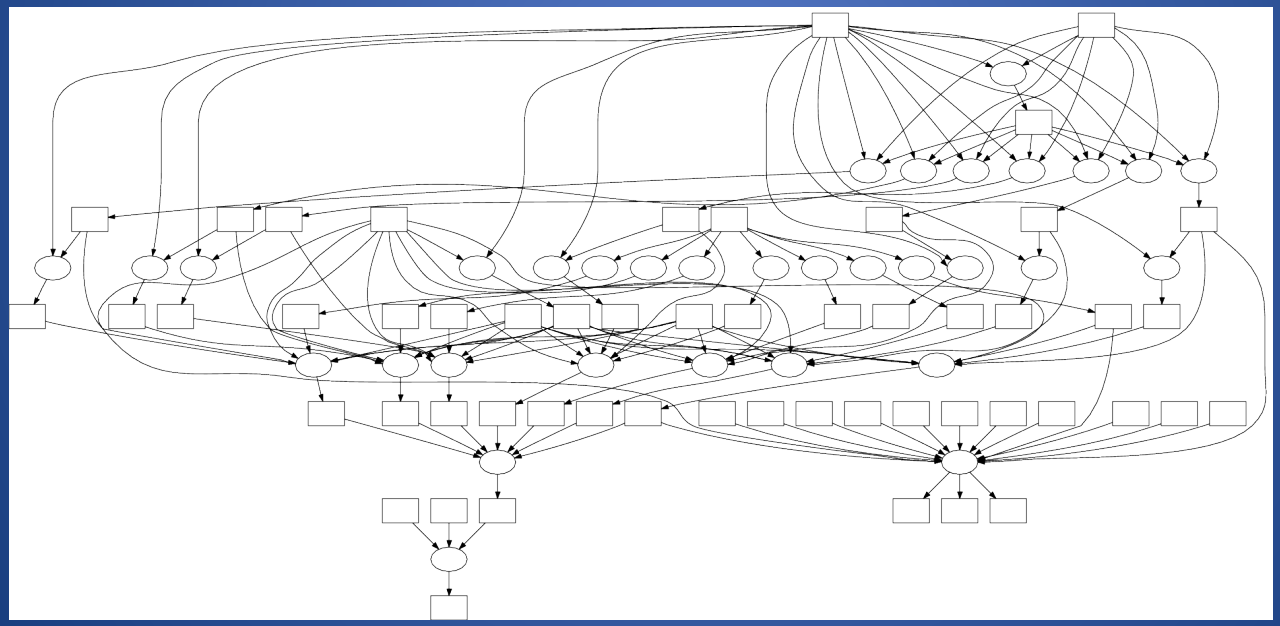
Transaction Log

TIME, TASKID, STATE, JOBID, STATE[0], STATE[1], …

1347281321284638 5 1 9206 5 1 0 0 0 6
1347281321348488 5 2 9206 5 0 1 0 0 6
1347281321348760 4 1 9207 4 1 1 0 0 6
1347281321348958 3 1 9208 3 2 1 0 0 6
1347281321629802 4 2 9207 3 1 2 0 0 6
1347281321630005 2 1 9211 2 2 2 0 0 6

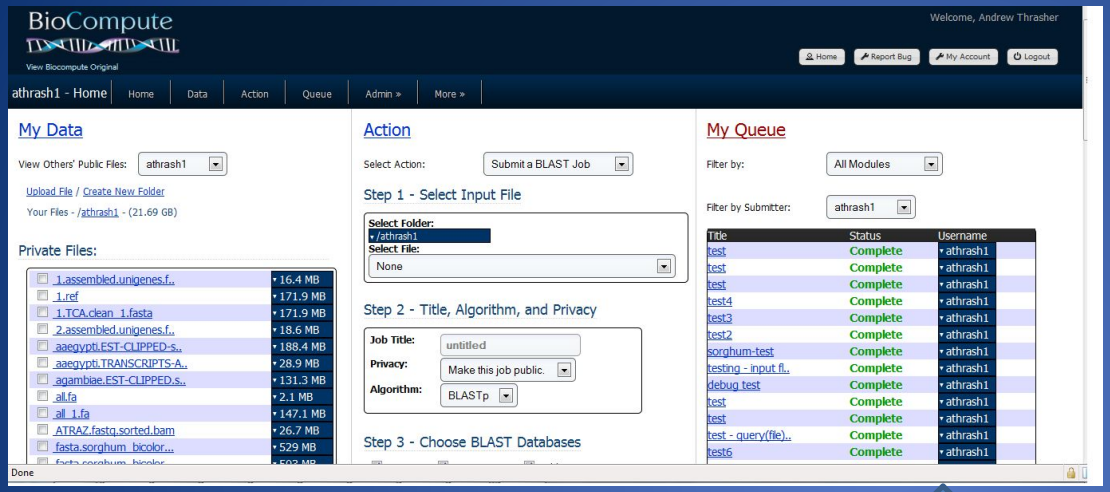
Visualization with DOT

- makeflow_viz –D example.mf > example.dot
- dot –T gif < example.dot > example.gif



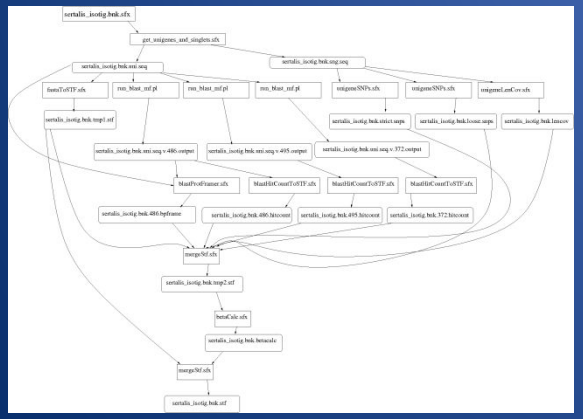
DOT and related tools:
http://www.graphviz.org

Example: Biocompute Portal



BLAST
SSAHA
SHRIMP
EST
MAKER
…

Generate Makefile



Progress Bar

Transaction Log

Update Status

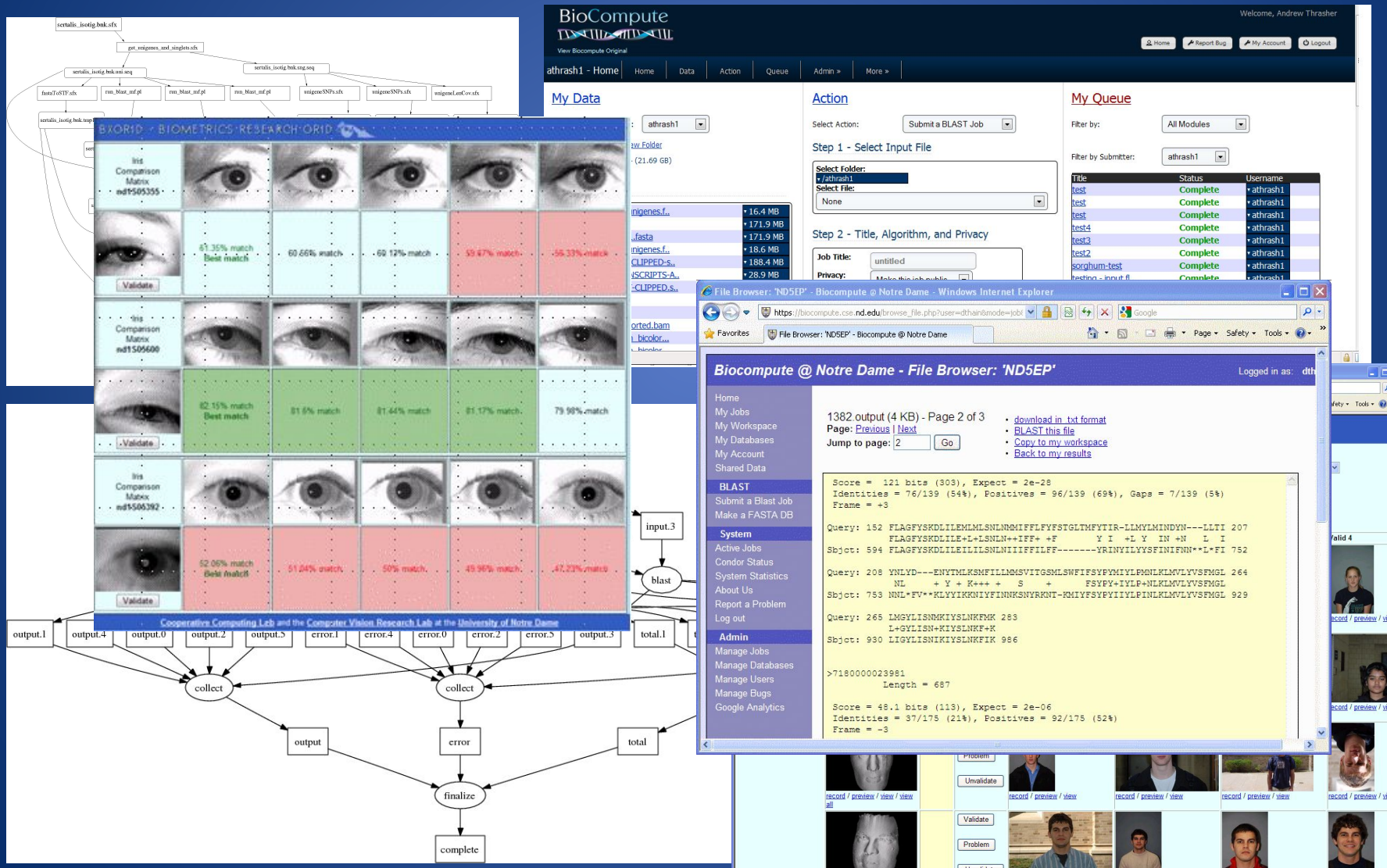
Run Workflow

Make flow

Submit Tasks

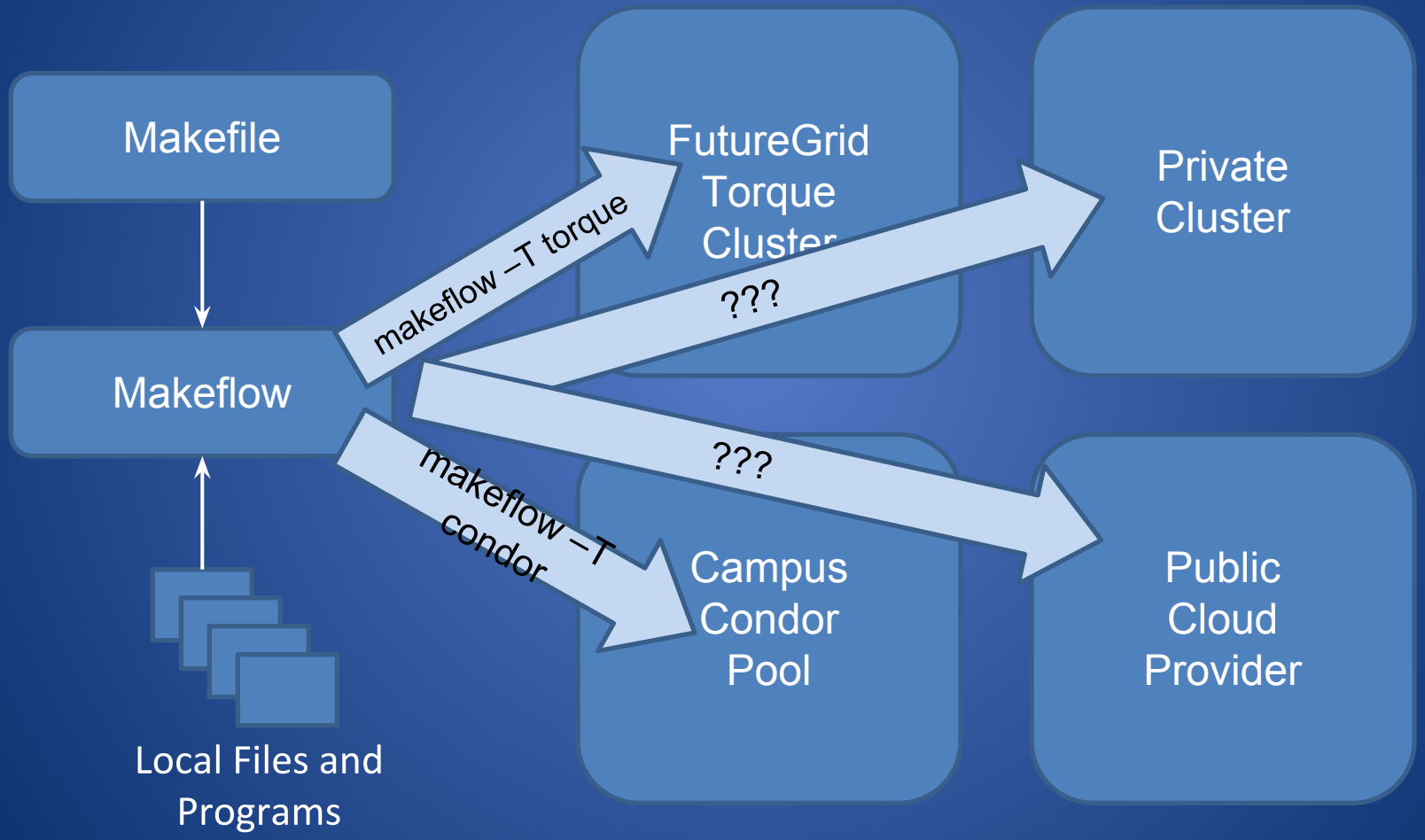
Condor Pool

Makeflow Applications

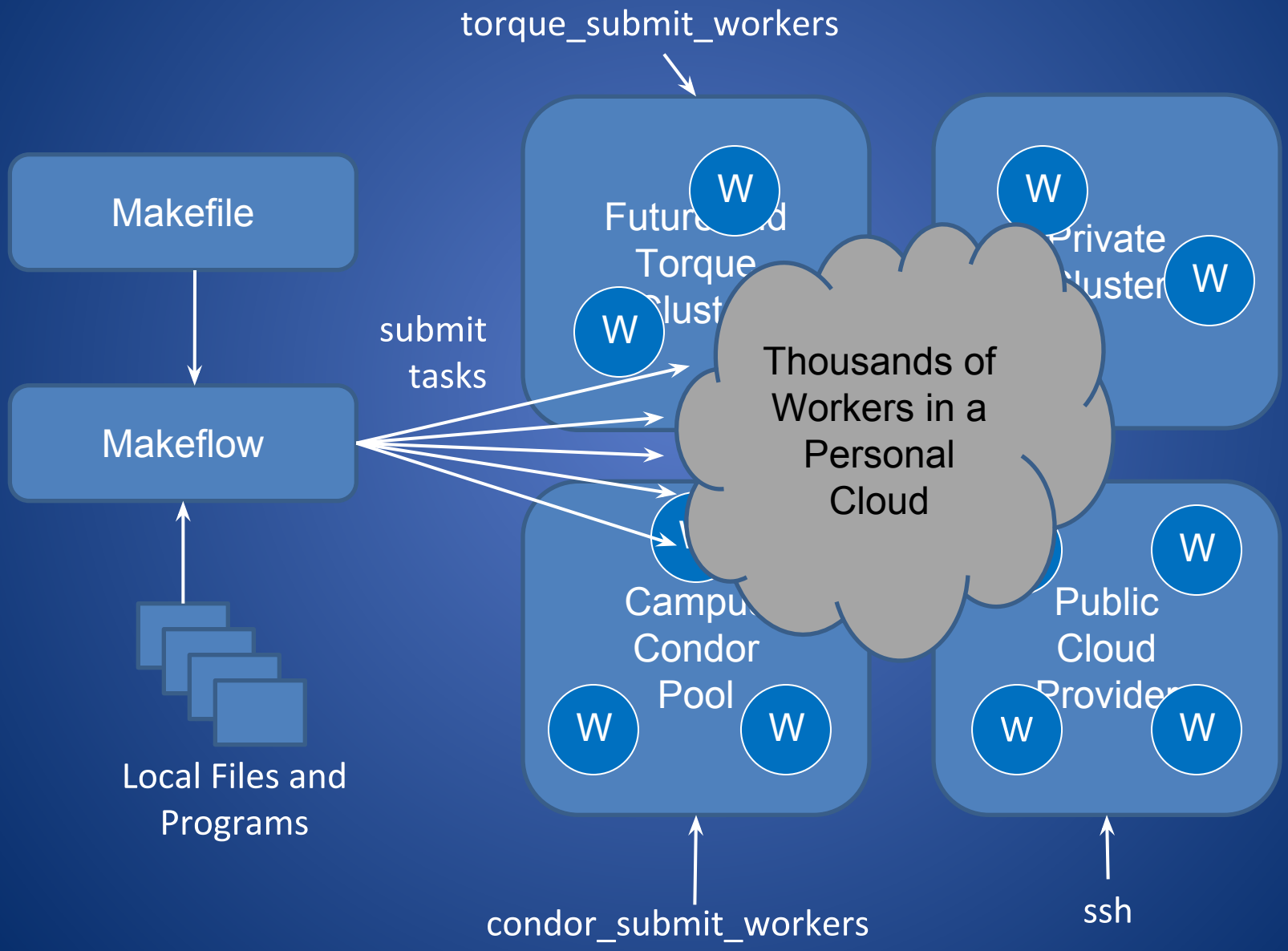


Makeflow + Work Queue

Makeflow + Batch System



Makeflow + Work Queue



Advantages of Work Queue

- Harness multiple resources simultaneously.
- Hold on to cluster nodes to execute multiple tasks rapidly. (ms/task instead of min/task)
- Scale resources up and down as needed.
- Better management of data, with local caching for data intensive tasks.
- Matching of tasks to nodes with data.

Makeflow and Work Queue

To start the Makeflow

% makeflow –T wq sims.mf

Could not create work queue on port 9123.

% makeflow –T wq –p 0 sims.mf

Listening for workers on port 8374…

To start one worker:

% work_queue_worker studentXX.cse.nd.edu 8374

Start Workers Everywhere

Submit workers to Condor:

condor_submit_workers studentXX.cse.nd.edu 8374 25

Submit workers to SGE:

sge_submit_workers studentXX.cse.nd.edu 8374 25

Submit workers to Torque:

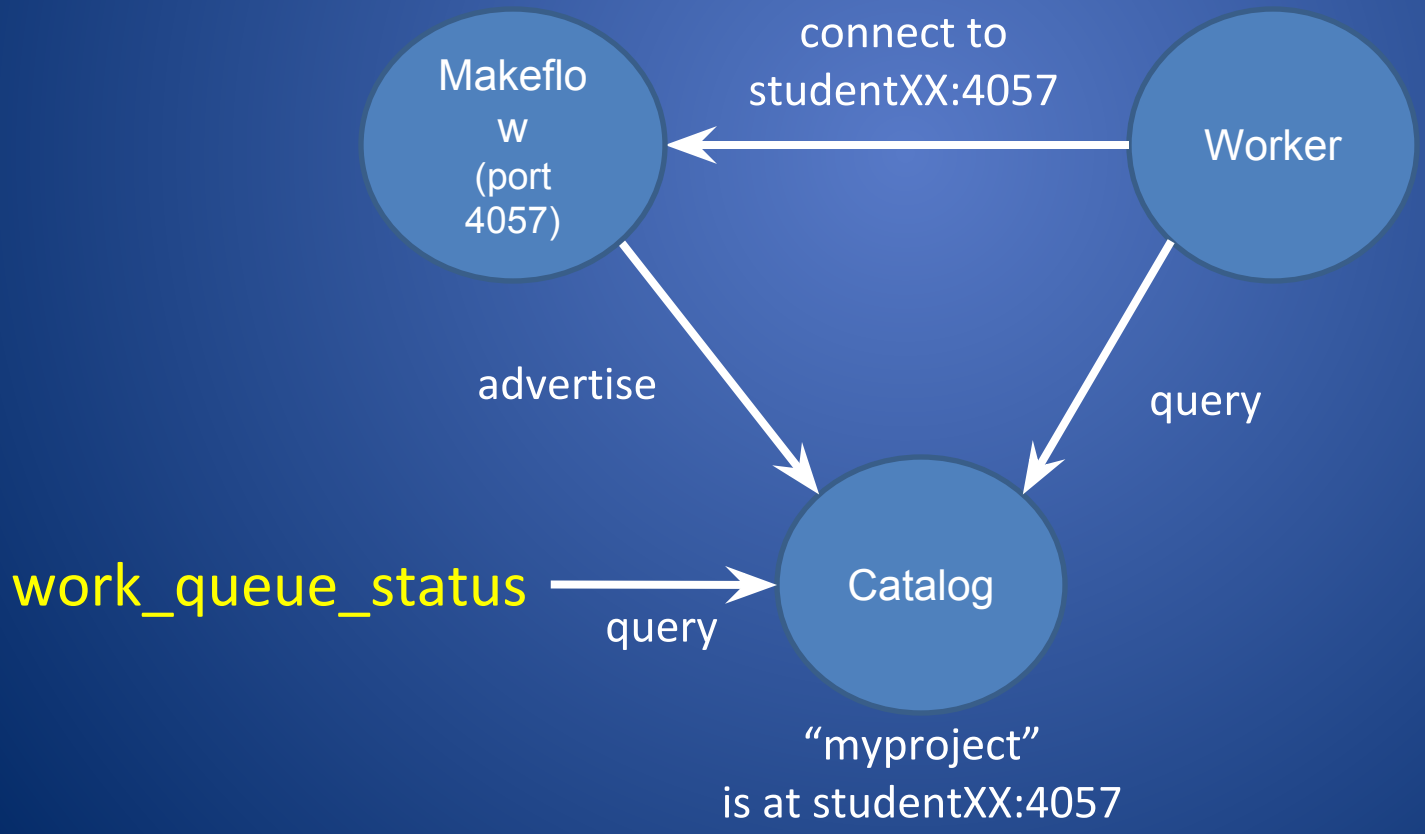
torque_submit_workers studentXX.cse.nd.edu 8374 25

Keeping track of port numbers gets old fast…

Project Names

makeflow …
–N myproject

work_queue_worker
–N myproject



Project Names

Start Makeflow with a project name:

```
% makeflow –T wq –N myproject sims.mf
Listening for workers on port XYZ…
```

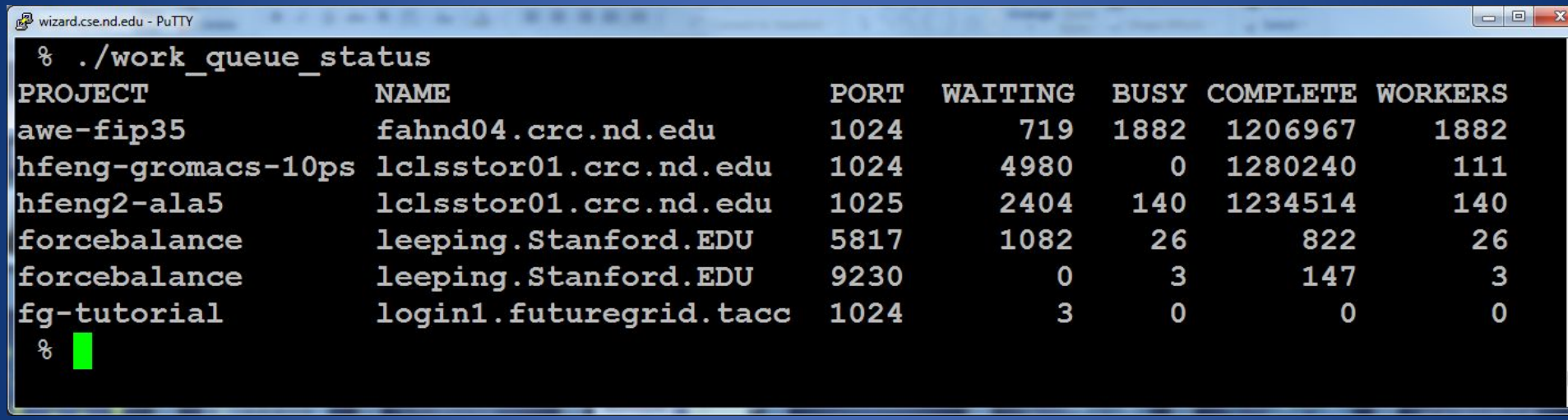
Start one worker:

```
% work_queue_worker -N myproject
```

Start many workers:

```
% torque_submit_workers –N myproject 5
```

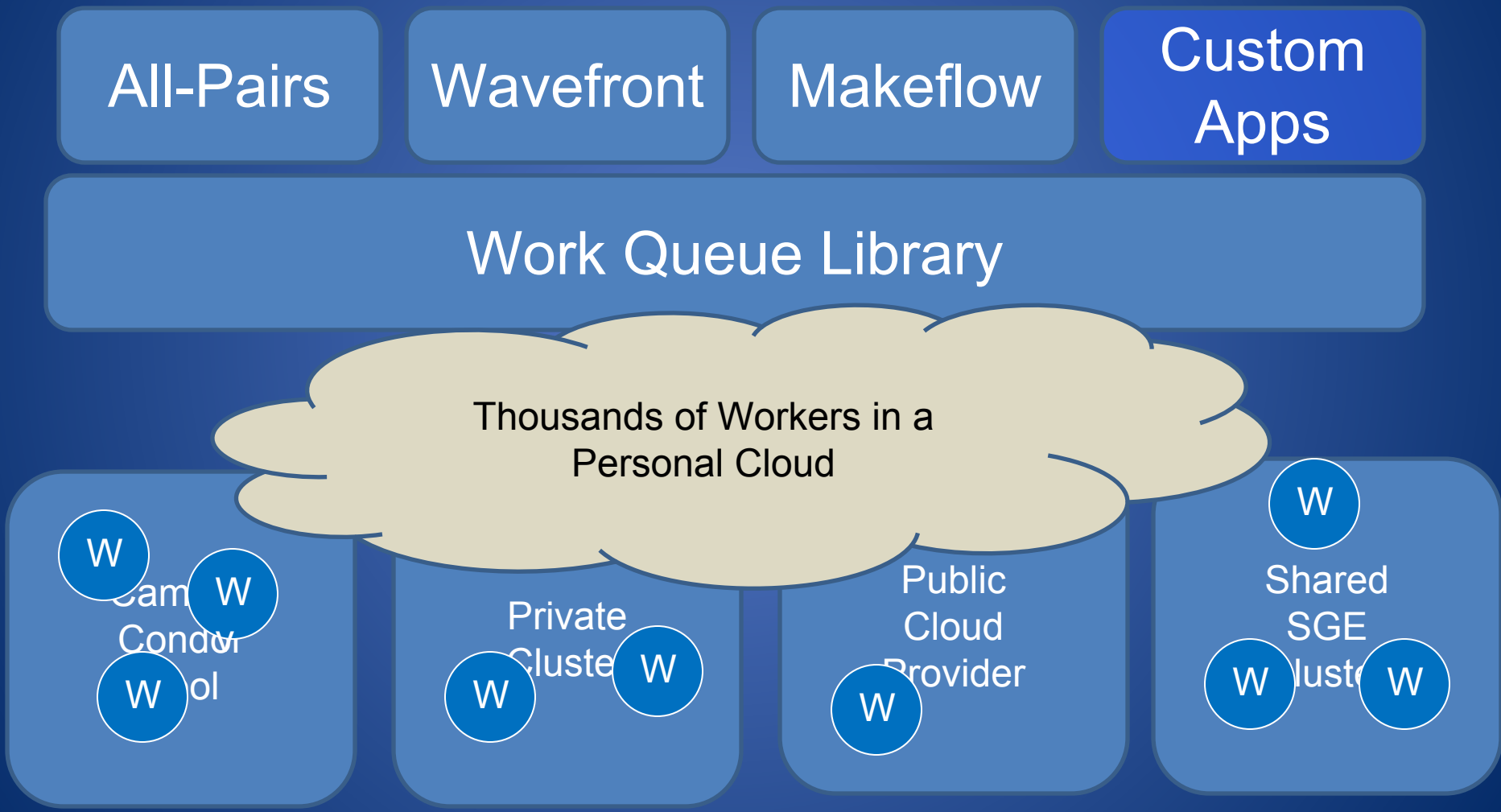
work_queue_status



Resilience and Fault Tolerance

- MF +WQ is fault tolerant in many different ways:
  - If Makeflow crashes (or is killed) at any point, it will recover by reading the transaction log and continue where it left off.
  - Makeflow keeps statistics on both network and task performance, so that excessively bad workers are avoided.
  - If a worker crashes, the master will detect the failure and restart the task elsewhere.
  - Workers can be added and removed at any time during the execution of the workflow.
  - Multiple masters with the same project name can be added and removed while the workers remain.
  - If the worker sits idle for too long (default 15m) it will exit, so as not to hold resources idle.

Elastic Application Stack



Makeflow is an example of the
Directed Acyclic Graph
(or Workflow)
model of programming.

Work Queue is an example of the Submit-Wait model of programming.

(more on that next time)

The Directed Acyclic Graph model

can be implemented using

the Submit-Wait model