Introduction to OSG

Tim Cartwright

OSG User School Director & OSG Special Projects Manager

University of Wisconsin–Madison

Overview

So far, we have seen how to use HTC on one cluster

Sometimes, that is not enough!

(Don't let computing hold back your science, remember?)

Today, we see what it takes to get more resources *

* ***Caveat:*** *I will focus on compute resources; Friday will focus on data.*

Accessing compute resources

Free Resources – In Your Lab

Server or cluster in your lab

- 👍 Not your laptop, control everything
- 👎 Buy and maintain it, not a lot of resources

https://images.abmx.com/30/3004/abmx_3004_1_1200.jpg

Free Resources – Local Cluster

Department or campus cluster

- 👍 No/low direct costs, local help
- 👎 Shared; maybe Slurm, PBS/ Torque, LSF…

No campus cluster? Talk to CIO!

Note! NSF CC* Compute awards

https://www.nsf.gov/pubs/2021/nsf21528/nsf21528.htm

https://www.pngall.com/wp-content/uploads/5/Server-Rack-PNG-Free-Image.png

Free Resources – Collaborators

Collaborators

- 👍 No/low direct costs, may be tailored to project
- 👎 Shared, project-specific, hard to come by

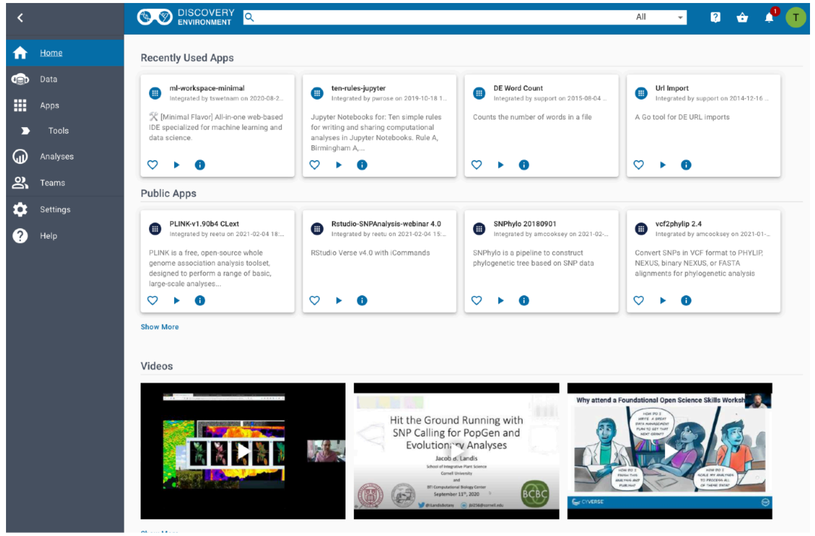
https://www.dunescience.org/about-the-collaboration/

Free Resources – Science Gateways

Science Gateways (e.g., web front-end to a cluster)

👍 Easy to use, no/low cost

👎 Only for pre-defined use cases



Commercial Resources

- **Commercial clouds** *(Amazon, Google, Microsoft, …)*
  - 👍 Don't own, high availability, many options (e.g., GPUs), …
  - 👎 Pay/hour, data out may be costly; challenging to manage
- **Managed clouds** *(Azure CycleCloud, Globus Genomics, …)*
  - 👍 As above, but *less* to manage
  - 👎 Costs more (paying someone to manage), fewer options?
- But keep commercial options in mind:
  - Credits may be available
  - May be able to write into grants
  - May be helpful for burst of activity (e.g., for a deadline)

What Do We Want?

- Lots of resources – available, stretchy, & reliable
- Submit locally, run globally (as close as possible)
- Automation to get resources, manage them, and run jobs
- Free would be nice! (Who pays?)

OSG: *Distributed* HTC

A dHTC Challenge

- What would you do with logins to 10 clusters?
  - 5 sites run HTCondor, 3 Slurm, 1 PBS, 1 LSF
  - 1 site focuses on special hardware (GPUs)
  - 2 sites have some servers with lots of memory
  - All Linux… 3 RHEL, 4 CentOS, 2 Rocky, and 1 Ubuntu
  - 1 site only takes biology-related jobs
  - 4 sites limit job duration, but different limits on each
- You want to run 2,000 jobs … go!

Manual dHTC

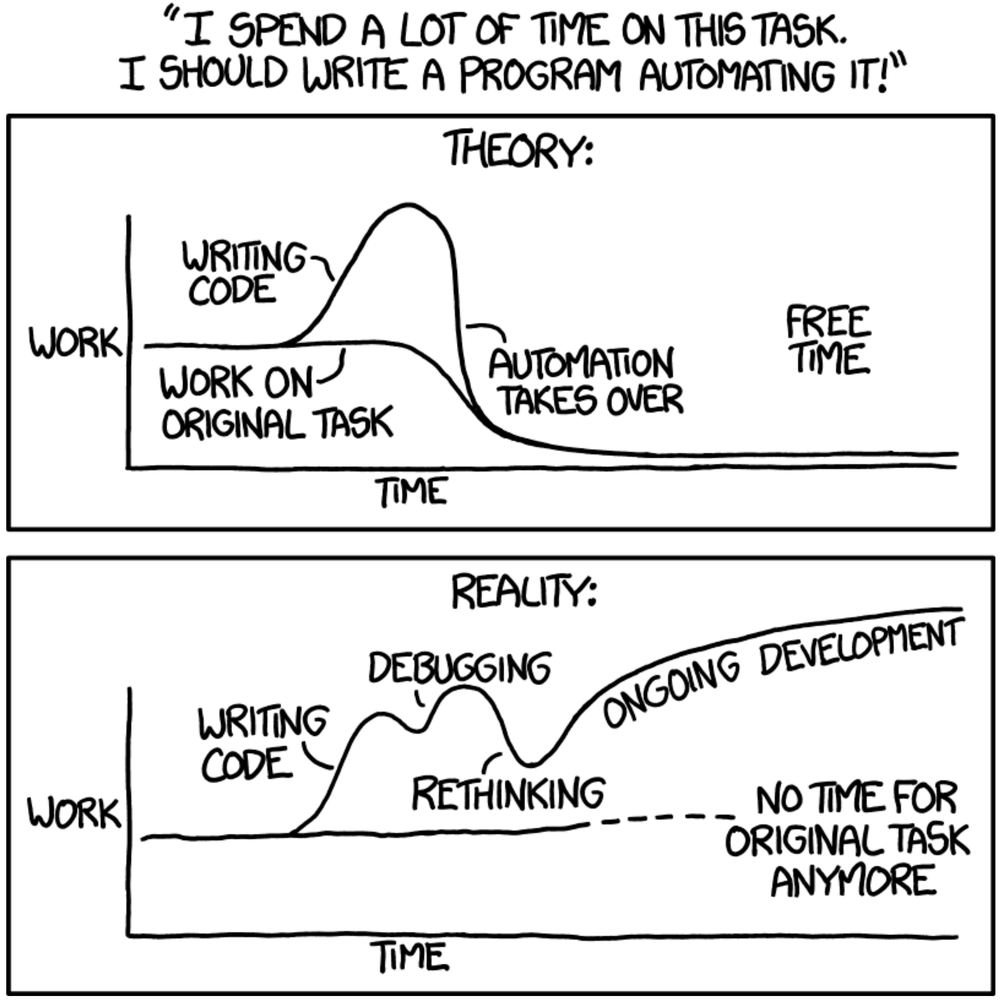
One could imagine this process:

1. Log in to System #1, check availability, submit 200
2. Log in to System #2, check availability, submit 150
3. Log in to System #3, check availability, submit 400
4. Log in to System #4, *oh wait, it's down today*
5. Log in to System #5, check availability, submit 350
6. ….

Which jobs are left to submit? Which are running? **Help!**

Automation?

Automation can help, but…



https://xkcd.com/1319/

Fundamental Flaws

There are fundamental flaws to this approach:

- Commits jobs to clusters *before* getting resources!
- Uses only a snapshot of availability
- Things could turn out very differently than planned
  - Other users get resources first (and run for days)
  - Your jobs don't actually match resources
  - Your jobs start, but fail every time

A Better Approach to dHTC

- Get resources first (due to demand or being idle)
- Consolidate resources into a pool
- Provide users with an Access Point into the pool (not quite *submit locally*, but at least just 1 place)
- Automate management of resources and jobs
- Sounds easy, right? 😉

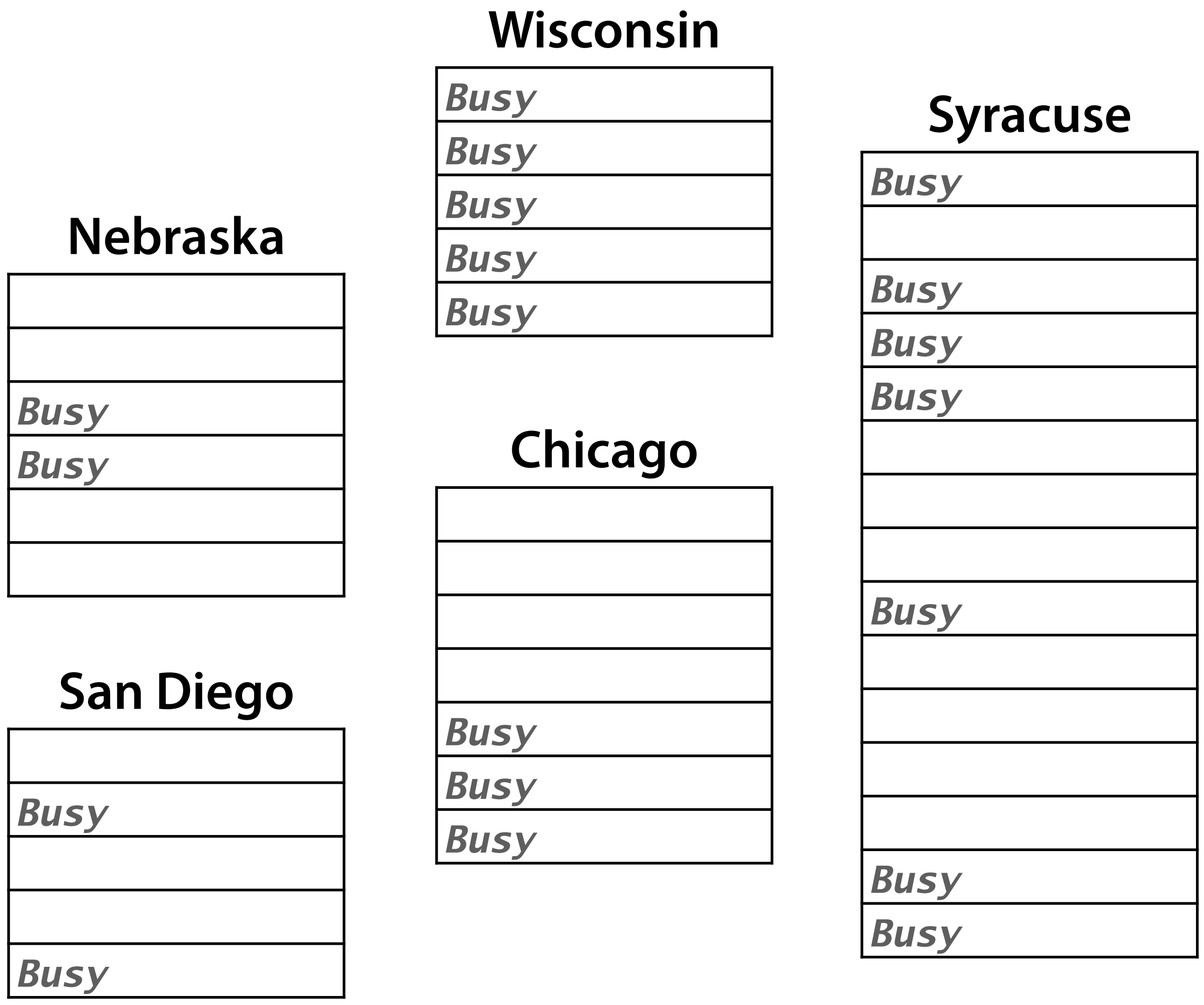
OSG dHTC Diagram

Step 1: Before OSG

Nothing available at Wisc. 😭

Queue

```
Job1.0
Job1.1
Job1.2
Job1.3
…
Job1.1999
```



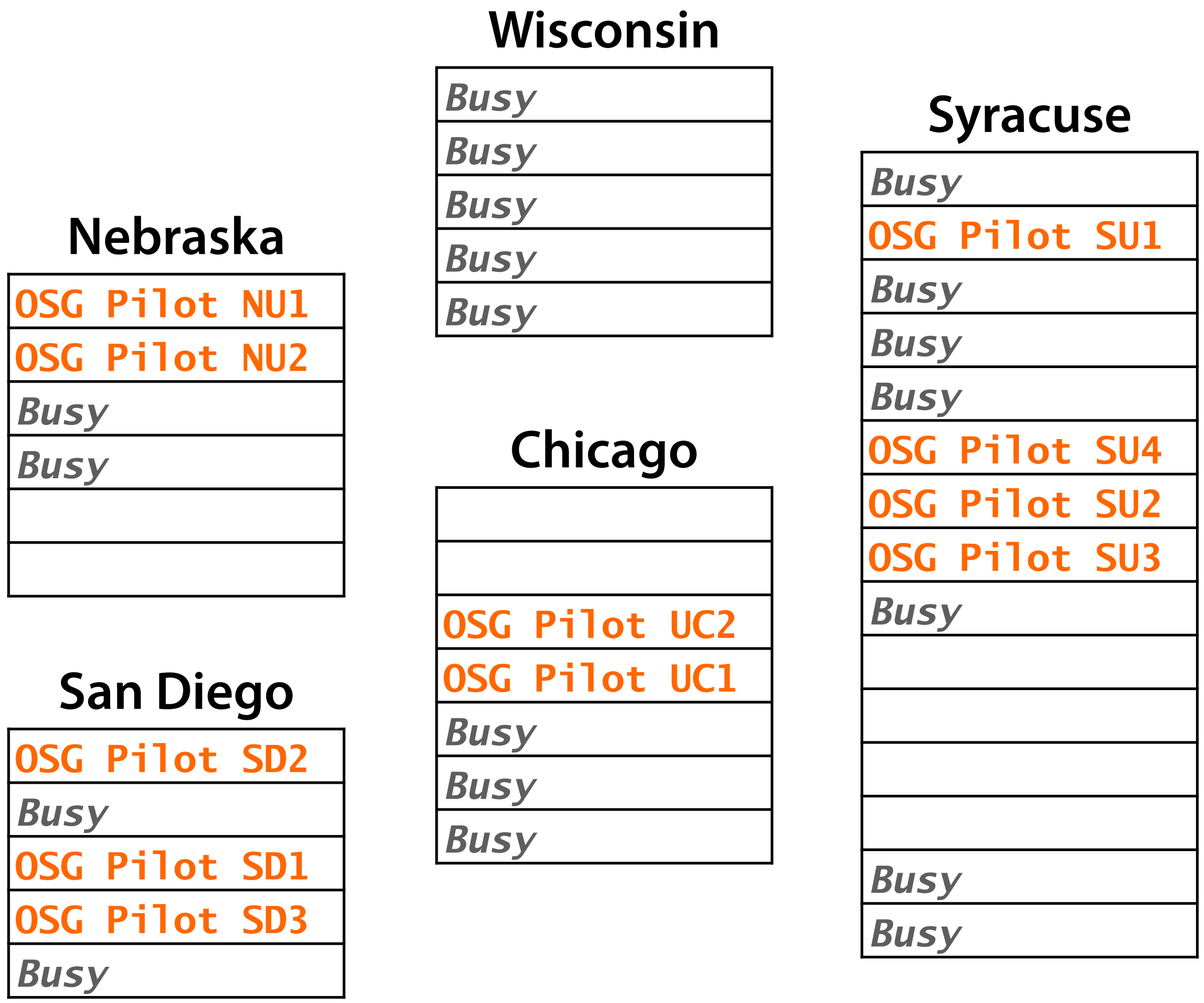
OSG dHTC Diagram

Step 2: Run OSG Pilots

Getting resources!

Queue

```
Job1.0
Job1.1
Job1.2
Job1.3
…
Job1.1999
```



OSG dHTC Diagram

Step 3: OSG Pilots

Add resources to Pool

Queue

```
Job1.0
Job1.1
Job1.2
Job1.3
…
Job1.1999
```

Pool

OSG Pilot NU1	*idle*
OSG Pilot NU2	*idle*
OSG Pilot SD1	*idle*
OSG Pilot SD2	*idle*
OSG Pilot SD3	*idle*
OSG Pilot UC1	*idle*
OSG Pilot UC2	*idle*
OSG Pilot SU1	*idle*
OSG Pilot SU2	*idle*
OSG Pilot SU3	*idle*
OSG Pilot SU4	*idle*

Nebraska

OSG Pilot NU1
OSG Pilot NU2
Busy
Busy

San Diego

OSG Pilot SD2
Busy
OSG Pilot SD1
OSG Pilot SD3
Busy

Wisconsin

Busy
Busy
Busy
Busy
Busy

Chicago

OSG Pilot UC2
OSG Pilot UC1
Busy
Busy
Busy

Syracuse

Busy
OSG Pilot SU1
Busy
Busy
Busy
OSG Pilot SU4
OSG Pilot SU2
OSG Pilot SU3
Busy
Busy
Busy

OSG dHTC Diagram

Step 4: Run jobs

HTCondor with Queue & Pool

Queue

```
Job1.0
Job1.1
Job1.2
Job1.3
…
Job1.1999
```

Pool

OSG Pilot NU1	Job1.4
OSG Pilot NU2	*idle*
OSG Pilot SD1	Job1.0
OSG Pilot SD2	Job1.3
OSG Pilot SD3	*idle*
OSG Pilot UC1	Job1.2
OSG Pilot UC2	Job1.6
OSG Pilot SU1	Job1.8
OSG Pilot SU2	Job1.12
OSG Pilot SU3	Job1.10
OSG Pilot SU4	*idle*

Nebraska

NU1 > Job1.4
NU2 > *idle*
Busy
Busy

San Diego

SD2 > Job1.3
Busy
SD1 > Job1.0
SD3 > *idle*
Busy

Wisconsin

Busy
Busy
Busy
Busy
Busy

Chicago

UC2 > Job1.6
UC1 > Job1.2
Busy
Busy
Busy

Syracuse

Busy
SU1 > Job1.8
Busy
Busy
Busy
SU4 > *idle*
SU2 > Job1.12
SU3 > Job1.10
Busy
Busy
Busy

OSG dHTC Diagram

Getting other resources

E.g., SU starts Pilots when idle

Queue

```
Job1.0
Job1.1
Job1.2
Job1.3
…
Job1.1999
```

Pool

OSG Pilot NU1	Job1.4
OSG Pilot NU2	*idle*
OSG Pilot SD1	Job1.0
OSG Pilot SD2	Job1.3
OSG Pilot SD3	*idle*
OSG Pilot UC1	Job1.2
OSG Pilot UC2	Job1.6
OSG Pilot SU1	Job1.8
OSG Pilot SU2	Job1.12
OSG Pilot SU3	Job1.10
OSG Pilot SU4	*idle*
SU Pilot OG1	*idle*

Nebraska

NU1 > Job1.4
NU2 > *idle*
Busy
Busy

San Diego

SD2 > Job1.3
Busy
SD1 > Job1.0
SD3 > *idle*
Busy

Wisconsin

Busy
Busy
Busy
Busy
Busy

Chicago

UC2 > Job1.6
UC1 > Job1.2
Busy
Busy
Busy

Syracuse

Busy
SU1 > Job1.8
Busy
Busy
Busy
SU4 > *idle*
SU2 > Job1.12
SU3 > Job1.10
Busy
SU Pilot OG1
Busy
Busy

OSG dHTC – A Few Details

- OSG *pilots* get resources
  - *OSG Factory* submits pilots to clusters at known sites; some will run
  - Site starts pilots itself when cluster resources are idle
  - Similar processes for XD (HPC) and Cloud resources
- **A pilot runs part of HTCondor itself**
  - A pilot *leases* the resources it is given for a while
  - Can expire after time, when idle, or when kicked out
  - A pilot doesn't really use resources, just holds on to them, and reports them to a central service, adding to a pool
- An *Access Point* is a place to submit jobs to a pool
- OSG and HTCondor manage/automate the details!

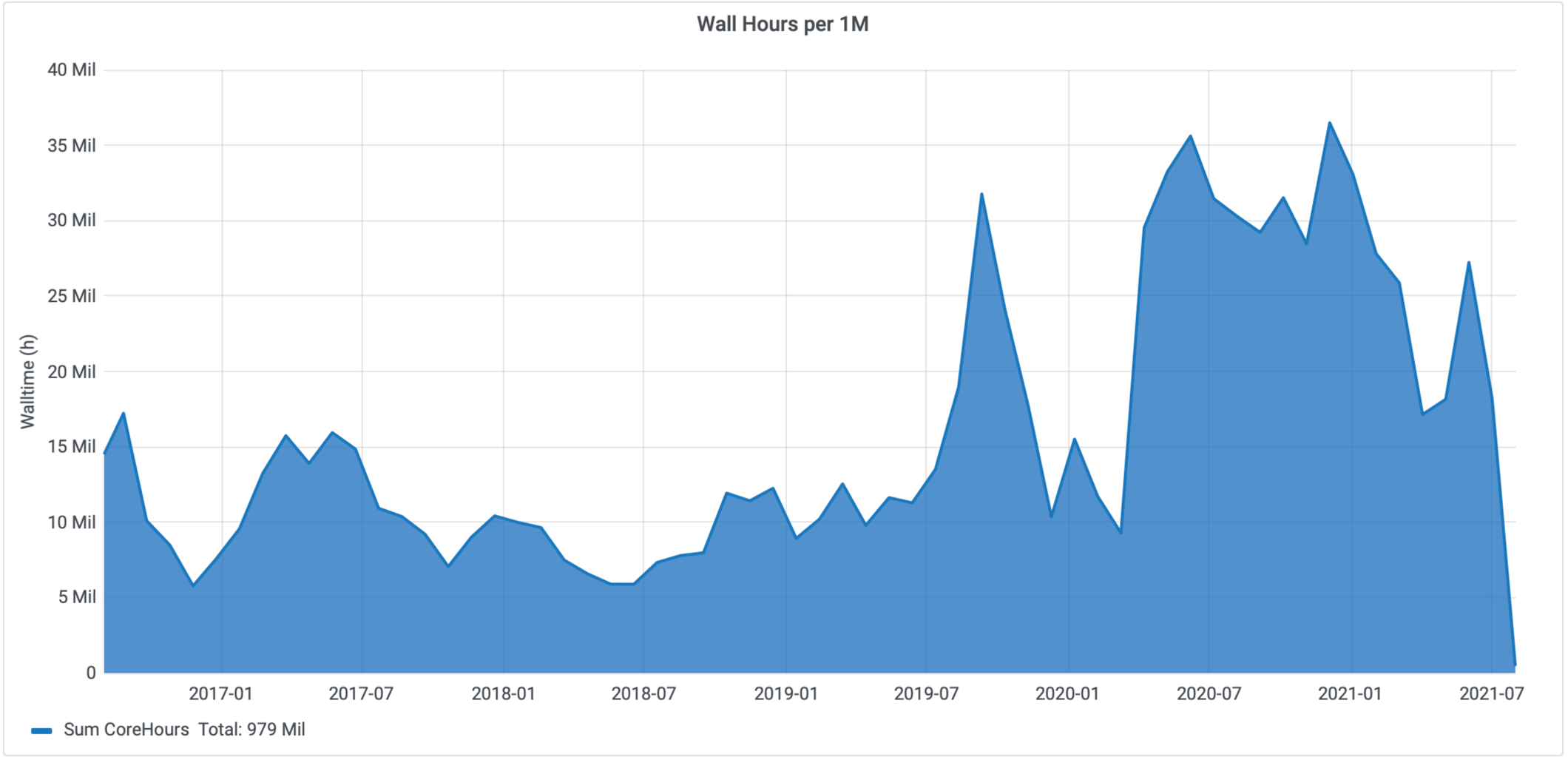
Note: Terms in *italics* are jargon. You may hear these terms, but it is not critical to memorize them.

Open Science Pool

- **Open Science Pool** (OS Pool) for all of Open Science
- It has many Access Points (e.g., projects, campuses)
- **OSG Connect** is an Access Point for US projects (incl. collaborators)
- Other pools exist for specific groups
  - Collaborations (e.g., gravitational-wave projects)
  - Projects (e.g., DUNE neutrino physics project)
  - Campuses (e.g., HCC at University of Nebraska)

OSG Sites *(Many are in Open Science Pool)*

Open Science Pool Usage



Using OSG

OSG Is HTCondor

- OSG (e.g., OS Pool) is like a local HTCondor pool:
  You have condor_q, condor_submit, DAGMan, etc.
- OS Pool bonus features!
  - More resources (usually) than a typical local system
  - Some pre-built software packages (SW lecture, Thu.)
  - Some storage on Access Point (Data lecture, Fri.)
  - Some special resources, like GPUs (GPU topic, Mon.)

So Why Learn How OSG Works?

- More “moving parts” means that there are more ways in which things can go wrong
- Greater variation than typical local system:
  - Varied hardware
  - Varied operating systems and software
  - Varied policies
  - Varying in availability
- Not all HTCondor features work or work well in OSG (e.g., condor_ssh_to_job)

Varied Hardware

- Request what you need in submit files
  - CPUs (“cores”): **`request_cpus`**
  - Memory: **`request_memory`**
  - Disk on execute server: **`request_disk`**
- Some other hardware requirements can be specified; search for documentation or contact us
  - Often in submit-file **`requirements`** expression
  - Example: GPU needs (GPU topic, Mon.)

Varied OSs and Software

- Varied operating systems
  - All Linux, *mostly* recent, but lots of variation
  - Software on Access Point may not exist on execute! (e.g., Python 2 versus Python 3)
- Your software
  - Never assume your software is on the execute server
  - Attend tomorrow's (Thu.) lecture on this topic!

Varied Policies

- Individual sites/clusters have their own policies
  - Example: Maximum run-time of a job (or its pilot)
- If possible, set requirements for what you need
  - But this does not help with, e.g., maximum run-time
- Generally, try to make “OSG-sized” jobs (see next)

What Makes a Good OSG Job?

Per-Job **Resources**	**Ideal Jobs!** (up to 10,000 cores, per user!)	**Still Very Advantageous!**	**Probably not…**
cores (GPUs)	**1** (1; non-specific)	**<8** (1; specific GPU type)	**>8 (or MPI)** (multiple)
Walltime *(per job)*	**<10 hrs*** *or checkpointable	**<20 hrs*** *or checkpointable	**>20 hrs**
RAM *(per job)*	**<few GB**	**<10 GB**	**>10 GB**
Input *(per job)*	**<500 MB**	**<10 GB**	**>10 GB**
Output *(per job)*	**<1 GB**	**<10 GB**	**>10 GB**
Software	*'portable' (pre-compiled binaries, transferable, containerizable, etc.)*	*most other than →→→*	*licensed software; non-Linux*

More OSG Tips – Testing

- Test early, test often
- Specify output, error, and log files
- If possible, add strategic logging to software (but don't fill disk with logs!)

More OSG Tips – Scaling Up

- 1 job
  - Did it work?
  - Check and adjust resource usage!
- 10 jobs
  - Check everything again
  - Check and adjust resource usage again
- Maybe another intermediate stage
- At each scale: Fix issue and repeat until solid

More OSG Tips – Access Point

- Access Points are shared resources
  - No long-running and resource-intensive processes
  - Do those things in jobs instead
- Estimate **AP** resource needs for *whole* run(s)
  - Especially input and output data (Data lecture, Fri.)
  - Also: directories, logs, file transfers, etc.

More OSG Tips – Security

- Computer security is hard — read the headlines!
- OSG does its best, but no system is perfect
- Some suggestions:
  - Use strong, distinct passwords for each account
  - Do not share your account
  - Avoid world-writable directories and files
  - Avoid sensitive software and data (no HIPAA!)
  - Do not try to work around security barriers; contact us to help meet your goals in a safe way

Troubleshooting

General Troubleshooting Tips

- Comparing expectations vs. what happened: Either might be wrong!
- Read messages carefully — even if some parts make no sense, what hints can you get?
- Search online … but evaluate what you find
- Collect links and other resources that help
- Ask for help! And provide key details: versions, commands, files, messages, logs, etc.

Issue: Failed to Parse

```
$ condor_submit job.sh
Submitting job(s)
ERROR: on Line 6 of submit file:
ERROR: Failed to parse command file (line 6).
```

- Completely failed to submit!
- **Notice:** `Failed to parse`
- **Why:** You tried to submit your executable (or other file), not an HTCondor submit file
- **Fix:** Submit an HTCondor submit file (e.g., `.sub`)

Issue: Typos in Submit File

```
$ condor_submit sleep.sub
Submitting job(s)
• ERROR: No 'executable' parameter was provided
• ERROR: Parse error in expression:
         RequestMemory = 1BG
• ERROR: Executable file /bin/slep does not exist
```

- Also failed to submit (missing `job(s) submitted`)
- **Why:** Typos in your submit file (e.g., `BG` for `GB`)
- **Fix:** Correct typos!

Issue: Jobs Idle for a Long Time

```
$ condor_q
OWNER    BATCH_NAME     SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
cat      ID: 123456    6/30 12:34      _      _      1      1 123456.0
```

Jobs are `idle` for a **long** time – *can be hard to judge!*

```
$ condor_q -better-analyze 123456.0
...
                Slots
Step    Matched  Condition
-----  --------  ---------
[0]       13033  TARGET.PoolName == "CHTC"
[9]       13656  TARGET.Disk >= RequestDisk
[11]          0  TARGET.Memory >= RequestMemory
```

Issue: Jobs Go on Hold

```
$ condor_q
OWNER    BATCH_NAME     SUBMITTED   DONE   RUN    IDLE   HOLD  TOTAL JOB_IDS
cat      ID: 123456   7/11 11:23      _      _      _      1      1 123456.0
```

Jobs are held when HTCondor doesn't know what to do

```
$ condor_q -held
...
 ID          OWNER            HELD_SINCE  HOLD_REASON
123456.0   cat               7/11 11:24 Error from
slot1_16@e122.chtc.wisc.edu: Failed to execute '/var/lib/
condor/execute/slot1/dir_19728/condor_exec.exe': (errno=8:
'Exec format error')
```

Issue: Some Common Hold Reasons

`Failed to initialize user log to /path or /dev/null`

- Could not create log file, check */path* carefully

`Error from …: Job has gone over memory limit of AAA megabytes. Peak usage: BBB megabytes.`

- Job used too much memory
- Request more – at least *BBB* megabytes!

`Error from …: STARTER at … failed to send file(s) to <…>: error reading from /path: (errno 2) No such file or directory; SHADOW failed to receive file(s) from <…>`

- Job specified **`transfer_output_files`**
- But */path* on remote server was not found
- Jargon: SHADOW is Access Point, STARTER is Execute Point

What To Do About Held Jobs

1. If the situation can be fixed while job is held (e.g., you forgot to create directory for output):
   a. Fix the situation
   b. Release the job(s): **`condor_release`** ***`JOB_IDs`***
2. Otherwise (and this is common):
   a. Remove the held jobs: **`condor_rm`** ***`JOB_IDs`***
   b. Fix the problems
   c. Re-submit

Issue: Missing or Unexpected Results

- Job runs … but something does not seem right
  - Short or zero-length output file(s)
  - Very short runtime (almost instant)

- May be problems with app, inputs, arguments, …
  - Check log files for unexpected exit codes, etc.
  - Check output and error files for messages from app
  - Can't find anything? Add more debugging output

Issue: Badput

- What is *badput*?
  - Basically, wasted computing
    - Job runs for *97 minutes*, gets kicked off, starts over on another server
    - Job runs for *97 minutes*, is removed
  - **Not** jobs that must be re-run due to code changes! (that's just part of science, right?)
- Badput uses resources that others could have used
- Tools for self-monitoring are in development
- If contacted, help us help you and others!

More Troubleshooting Resources

- Brian Lin's OSG User School 2019 talk
  (TBH: I copied a lot from there!)
- OSG Connect documentation
- support@opensciencegrid.org

Acknowledgements

You Can Acknowledge OSG!

If you publish or present results that benefitted from using OSG, please acknowledge us!

https://support.opensciencegrid.org/support/solutions/articles/5000640421-acknowledging-the-open-science-grid

Acknowledgements

- OSG team, especially ***Brian Lin***, Mats Rynge, and Jason Patton
- This work was supported by NSF grants MPS-1148698, OAC-1836650, and OAC-2030508

Demonstrations