Summer School
Cluster Computing

June 4, 2024

Vedant Basu
(with input from D. Schultz)

What is a cluster? When do I need one?

- A Cluster is a network of processing units linked together as a single system, which can run scripts offline

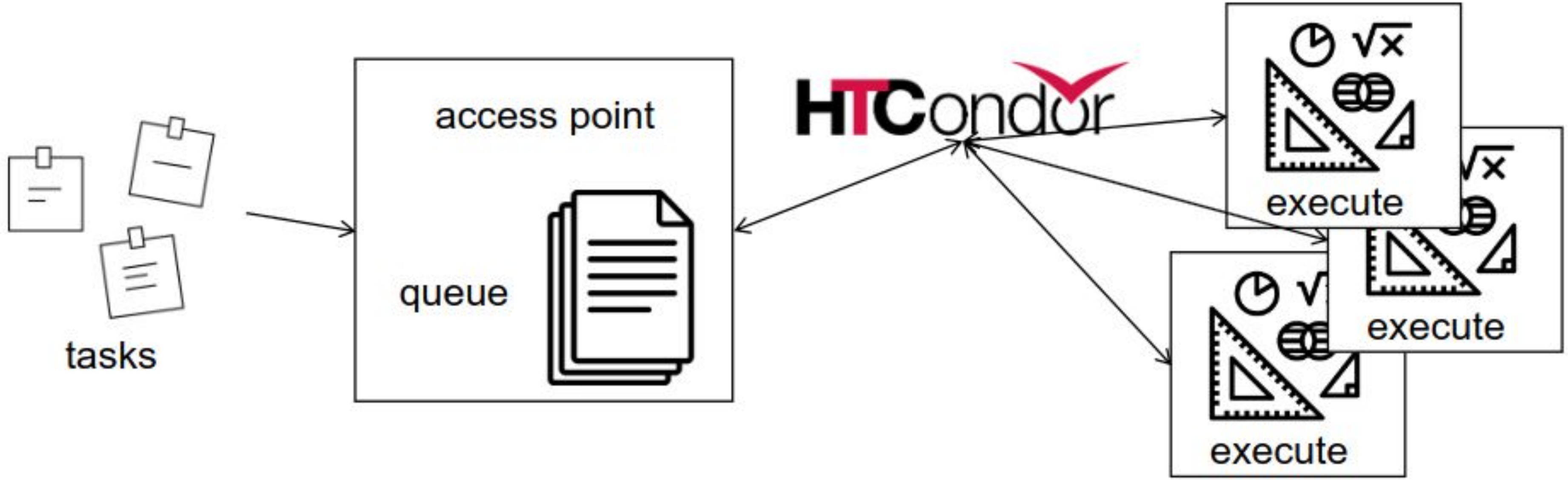
- The typical use case is to run a script over ‘n’ input data files, each of which would take ‘t’ time to complete. Running interactively on Cobalt, this would take ‘n * t’ time to complete, while running on the cluster, the n jobs get distributed to ‘n’ sites, so they all run in parallel

- When you have a load of jobs to run! Especially if
    - They can be run in parallel
    - They do not require large resource allocations
    - Each job should complete in < 48hrs

Does IceCube have clusters?

- We have access to 2! These are
  - NPX: Internal network hosted by UW-Madison. Has access to internal filesystems like /home and /data
  - The Grid: Network of CPUs and GPUs hosted by universities and computing sites around the world. IceCube files need to be transferred over

Condor



HTCondor: open source software framework for distributed computing

How do I use NPX?

1. Create a submit file, which details which script to run, what arguments to pass to the script, what resources to allocate…

2. The submit file can directly call the script to run, but some jobs have a wrapper shell script for finer control of environment variables

3. To run on NPX, ssh to **submit-1** from pub2 or pub4, and run *condor_submit job.sub*

Anatomy of a submit file

Basic HTCondor submit file:

- executable: a script or wrapper to run
- arguments: any cmdline arguments

- log: where to write the HTCondor job log
- output: where to write the job stdout
- error: where to write the job stderr
- notification: whether to send status emails

- transfer_input_files: send scripts to the job
- request_*: job resources
- queue N: number of jobs to run

```
executable = job.sh
arguments = physics.py

log = job.log
output = job.out
error = job.err
notification = never

transfer_input_files = physics.py

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1

queue 1
```

Monitoring Jobs

After submitting a job

```
submitter ~ $ condor_submit job.sub

Submitting job(s).

1 job(s) submitted to cluster 12898721.
```

You can monitor it with

```
submitter ~ $ condor_q

-- Schedd: submit-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:43:06

To monitor submitted jobs: condor_q

OWNER   BATCH_NAME       SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS

gmerino ID: 101524801   6/7  11:43      _      _      1      1 101524801.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for gmerino: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for all users: 1736 jobs; 0 completed, 0 removed, 1548 idle, 188 running, 0 held, 0
suspended
```

```
submitter ~ $ condor_q -nobatch

-- Schedd: submit-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:50:11

 ID            OWNER              SUBMITTED     RUN_TIME ST PRI SIZE CMD

101524801.0    gmerino           6/7  11:49   0+00:00:00 I  0    0.0 job.sh 10
```

ST = status

Most Common Job status:

- Idle "I": Job has not started yet… waiting in queue
- Running "R": job is currently running
- Completed "C": If the job has completed, it will not appear in condor_q
- Held "H": Stalled jobs. Something you need to fix

  A job that goes on hold is interrupted (all progress is lost) and kept from running again. It remains in the queue in the "H" state.

NPX Tips

- Use the provided scratch space for logfiles
- Logfiles in network filesystems (/home, /data/user) can generate instability and are prohibited
- Job duration is limited to 48 hours
  - If you want more, specify the 1_week or 2_week accounting groups
- If you want to run interactively, use `condor_submit -interactive`

```
executable = job.sh
arguments = physics.py

log = /scratch/$ENV(USER)/job.log
output = job.out
error = job.err
notification = never

+FileSystemDomain = "foo"
transfer_input_files = physics.py

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1+
AccountingGroup="1_week.$ENV(USER)"

queue 1
```

How do I use the Grid?

1. Set up a proxy certificate

2. Submit files created in a similar manner. Require specification of files to transfer to remote sites

3. Input and output files need to be transferred to remote sites using either GridFTP or HTTP file transfer (new!)

4. To run on the Grid, ssh to **sub-2** from pub2 or pub4, and run *condor_submit job.sub*

Grid Computing - Data Transfer

2. GridFTP file transfer - the old way of handling large data

```
#!/bin/bash
set -e
eval $(/cvmfs/icecube.opensciencegrid.org/py3-v4.2.1/setup.sh)
infile = $1
shift;
outfile = $1
shift;
echo “transferring input $infile”
globus-url-copy gsiftp://gridftp.icecube.wisc.edu/$infile file:/$PWD/infile

echo “running program”
$SROOT/metaprojects/icetray/v1.5.1/env-shell.sh python physics.py
infile outfile $@

echo “transferring output $outfile”
globus-url-copy file:/$PWD/outfile
gsiftp://gridftp.icecube.wisc.edu/$outfile

echo "Job complete!"
```

```
transfer_input_files = physics.py, x509
transfer_output_files =
arguments = /data/user/me/my_file.i3
/data/user/me/out_file.i3 --prog args
```

```
dschultz@sub-2 $ grid-proxy-init -valid 24:0 -out x509
Your identity: /DC=org/DC=cilogon/C=US/O=University of
Wisconsin-Madison/CN=David Schultz B47305562
Enter GRID pass phrase for this identity:
Creating proxy
..............................................................................
..............................................................................
....... Done
Your proxy is valid until: Thu Jun  8 21:22:03 2023

dschultz@sub-2 $ condor_submit job.sub
```

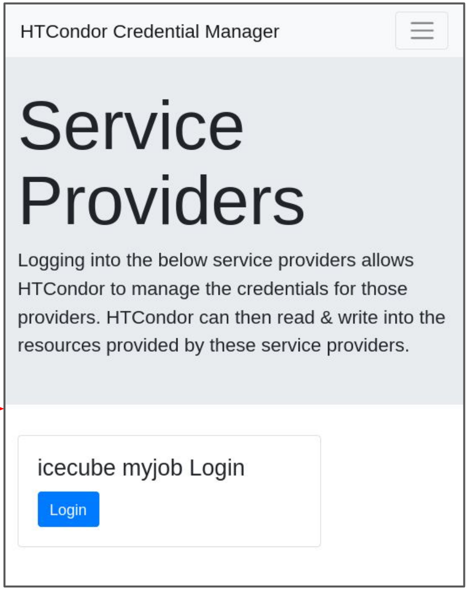
Grid Computing - Data Transfer

3. HTTP file transfer - the **new** way of handling large data

```
# required lines for http transfers
use_oauth_services = icecube
icecube_oauth_permissions_myjobs = offline_access

transfer_input_files =
physics.py,icecube.myjobs+https://data.icecube.aq/data/user/me/my_input.i3
transfer_output_files = out.hdf5
transfer_output_remaps = "out.hdf5 =
icecube.myjobs+https://data.icecube.aq/data/user/me/out.$(ClusterId).$(ProcId).hdf5"
```

```
dschultz@sub-2:~$ condor_submit job.sub
Submitting job(s)
Hello, dschultz.
Please visit:
http://localhost:22280/key/5b2dfca80ec4b5ebce55c40b114c40ab57290
3171e37efba065de29a2789999e
```


HTCondor Credential Manager

Service Providers

Logging into the below service providers allows HTCondor to manage the credentials for those providers. HTCondor can then read & write into the resources provided by these service providers.

icecube myjob Login

After logging in, `condor_submit` will work

Multiple jobs with DAGman

- What is a DAG?
    - A Directed Acyclic Graph is a tool which comes with HTCondor, which enables control and queueing of multiple jobs simultaneously.
    - Handles relationships between jobs
    - Saves state of a run, can rerun failed jobs

DAGMan

Let's make a basic DAG submit file and a regular submit file:

```
# file name: dagman.submit
JOB job1 job.condor
VARS job1 Filenum="001"
JOB job2 job.condor
VARS job2 Filenum="002"
JOB job3 job.condor
VARS job3 Filenum="003"
JOB job4 job.condor
VARS job4 Filenum="004"
```

```
# file name: job.condor
# special variables:
#  Filenum = Filenum var defined in dagman.submit
Executable = job.sh
Arguments = physics.py $(Filenum)

output = job.$(ProcId).out
error = job.$(ProcId).err
log = job.log

notification = never

queue
```

DAGMan - Job Dependencies

Now let us look at an example with dependencies, where one job must run before another one.

Let's make a dag with 3 parents and one child (maybe processing and cleanup?):

```
# file name: dagman.submit
JOB job1 job.condor
VARS job1 Filenum="001"
JOB job2 job.condor
VARS job2 Filenum="002"
JOB job3 job.condor
VARS job3 Filenum="003"
JOB job4 job.condor
VARS job3 Filenum="004"
# define the DAG relationship
Parent job1 job2 job3 Child job4
```

Summary

- Overview of IceCube cluster computing resources
- How do I use NPX?
- How do I use the Grid?
- DAGs and DAGman