Schedule

12:30	**Lunch**
13:00	**Part I** ● [15 min] Placing lists of jobs ● [20 min] What happened to my job? ● [15 min] Data placement ● [30 min] Troubleshooting strategies ● [20 min] GPU jobs
14:40	**Break**
14:50	**Part II** ● [40 min] Principles of DAGMan ● [40 min] Hands-on: DAGMan ● [20 min] Python bindings ● [20 min] Hands-on Python bindings ● [10 min] Computing at Nikhef ● [25 min] Philosophy & architecture
17:25	**Social**

Troubleshooting strategies

How to identify the cause of a problem

Identifying the cause of a problem requires understanding

(a) what you are attempting to do, *versus*

(b) what actually happened

How to identify the cause of a problem

Identifying the cause of a problem requires understanding

(a) what you are attempting to do, *versus*

(b) what actually happened

(hopefully you know, but if not…)

Examine your `.sub` submit file and your executable file (and related scripts)

- What data transfers are you doing?
- What is the structure of your working directory?
- What software/commands are you using?

How to identify the cause of a problem

Identifying the cause of a problem requires understanding

(a) what you are attempting to do, *versus*

(b) what actually happened

Use `condor_q`, `condor_history`, and the user `.log` file to understand the lifetime of the job as managed by HTCondor

Examine your `.out`, `.err` files for the messages your executable generates

Look at any other files that may have been generated/returned

Identifying the problem

Four general categories of problem:

1) The job is held
2) The job completed, but was unsuccessful
3) The job doesn't start
4) The job is running longer than expected

Identifying the problem

Four general categories of problem:

1) **The job is held**
2) The job completed, but was unsuccessful
3) The job doesn't start
4) The job is running longer than expected

1) The job is held

HTCondor is managing your job for you. When something goes wrong in that process, HTCondor puts the job on "hold".

HTCondor provides a "Hold Reason" that explains what happened.

1) The job is held

The Hold Reason should explain what the problem is. Use `condor_q -hold` to see the message:

```
$ condor_q -hold

ap40.uw.osg-htc.org : <128.105.68.62:9618?... @ 08/01/24 15:01:50
 ID          OWNER            HELD_SINCE  HOLD_REASON
130.0        alice             8/1   14:56 Transfer input files failure at …
```

The message will also be in the user `.log` file.

1) The job is held

Most common issues:

a) Data transfer failure
b) Job using too much resources (memory/disk/runtime)
c) Problem with execution/container

1) The job is held

Example message:

> **Transfer output files failure** **at the execution point** while sending files to access point ap40. Details: reading from file **/var/lib/condor/execute/hold_this_job.txt**: (errno 2) **No such file or directory**

The file at that path doesn't exist at the EP. Could be

(a) file doesn't exist anywhere on the EP
(b) file is in a different directory on the EP
(c) the submit file has a typo in the file path in `transfer_output_files` line

1) The job is held

Note

The actual cause may originate **earlier** in the job lifecycle than where the failure occurred

→ If you declare that an output file is to be transferred back to the AP, *but if your script didn't create the file* at the EP because of an error, it causes a `Transfer output file failure`

The files/jobstate at the execution point are **NOT** saved when job is held!

→ The job is *stopped* and the slot *reset* (files deleted) just like any completed job

Identifying the problem

Four general categories of problem:

1) The job is held
2) **The job completed, but was unsuccessful**
3) The job doesn't start
4) The job is running longer than expected

2) The job completed but was unsuccessful

What does "success" mean for your job? Why was it "unsuccessful"?

Typically:

a) Code didn't execute correctly or as expected
b) Necessary output files were not generated/returned/transferred

Need a good understanding of **what actually happened**

2) The job completed but was unsuccessful

Having various `ls`, `echo` statements in your executable script helps a lot

Instead of only:

```
my_command
```

Add statements to understand the state before & after execution:

```
echo "Here are the files at the start of the job:"
ls -R
echo "Executing main command..."
my_command
echo "... finished. Here are the files at the end of the job:"
ls -R
```

Compare files before and after. Was your output file created?

If available for your command, consider enabling debugging or verbose logging

2) The job completed but was unsuccessful

Can also utilize the `condor_chirp` command to have information sent directly back to the Access Point

For example, the following will add a statement to your `.log` file:

```
command1
condor_chirp ulog "Finished executing command1"
command2
```

For more information, see the manual page

Identifying the problem

Four general categories of problem:

1) The job is held
2) The job completed, but was unsuccessful
3) **The job doesn't start**
4) The job is running longer than expected

3) The job doesn't start

You submitted your job, but it's idle

Have patience!

Matchmaking cycle can take 5+ minutes to complete, longer if server is busy

The more / more-special resources requested, the longer the wait

- Fewer slots are capable of running jobs with larger resource requests
- Special resources (such as GPUs) may be limited but in high demand

3) The job doesn't start

You've been waiting a while, but the job is still idle..

Are you requesting too many resources?

Requesting 128 CPUs, maximum is 64 → job will *never* start!

Does your system have special submit requirements?

Have you already used the system a lot? (Your relative priority may be low)

For more information, can try `condor_q -better-analyze`

This is a <u>tool</u> for looking into matchmaking — may not give the definitive reason why the job isn't starting.

Identifying the problem

Four general categories of problem:

1) The job is held
2) The job completed, but was unsuccessful
3) The job doesn't start
4) **The job is running longer than expected**

4) The job is still running

Check the events in the job `.log` file and see if

a) The job has been continuously running on the same slot
b) The job has been interrupted and restarted on another slot
c) The job is stuck on the file transfer step

4) The job is still running

Check the events in the job `.log` file and see if

a) **The job has been continuously running on the same slot**
b) The job has been interrupted and restarted on another slot
c) The job is stuck on the file transfer step

For example, you expected the job to run for only 1 hour, and has instead been running for 24 hours.

- difference in resource amounts?
- problem with code/software/machine?

4) The job is still running

Check the events in the job `.log` file and see if

a) **The job has been continuously running on the same slot**
b) The job has been interrupted and restarted on another slot
c) The job is stuck on the file transfer step

To investigate a still-running job, can use

- `condor_tail` - return the last X bytes of the job output (manual page)
- `condor_ssh_to_job` - log in directly to the execution slot and poke around (doesn't work on all systems; manual page)

Or add a `timeout` command to your executable and resubmit with debugging statements.

4) The job is still running

Check the events in the job `.log` file and see if

a) The job has been continuously running on the same slot
b) The job has been interrupted and restarted on another slot
c) The job is stuck on the file transfer step

If just once or twice, adjust your expectation of the runtime (start counting from when the job last started running instead of when you submitted the job)

If happening many times, your job runtime may be too long for the system, or there is a problem with the job/system. Check with system admin/facilitator

4) The job is still running

Check the events in the job `.log` file and see if

a) The job has been continuously running on the same slot
b) The job has been interrupted and restarted on another slot
c) The job is stuck on the file transfer step

In addition to the `.log` entry, can see this with `condor_q -nobatch` if the job is stays in `>` or `<` states

Can happen if you or someone else is transferring a lot of data (large size or many files) and the AP is overwhelmed.

If this happens, notify your system administrator.

Identifying the problem

Four general categories of problem:

1) The job is held
2) The job completed, but was unsuccessful
3) The job doesn't start
4) The job is running longer than expected

In General

A problem is when the **actual outcome** does not match your expectations for a **"successful" outcome**. What are the criteria for your jobs to be "successful"?

If your jobs/workflow do not meet your success criteria, then

1) Identify the **specific** criteria that is not being met
2) Identify the **cause** of why that criteria is not being met*
3) **Change** your implementation
4) **Evaluate** the change

You may need to cycle through these steps, or you may need to change your criteria for "success"

Example

Success criteria: my job will complete (without errors) and return the file `output.csv` that contains my processed data.

Scenario: my job completed but I don't see the file `output.csv`.

1) The expected output file does not exist
2) I examined the `.err` file and see a message `no module named 'numpy'`. Since my code requires the Python `numpy` package, my script did not run or generate the `output.csv` file.
3) I modified my software files to include the `numpy` package
4) I tested the job, and this time it produces the expected `output.csv`

Different Scales of Problems

When submitting many jobs, several dimensions come into play:

1 job fails	All jobs fail

volume

Rarely/randomly fails	Always fails

frequency

Fails on a specific machine	Fails on every machine

location

Transient	***Systemic***

cause

Different Scales of Problems

If **Transient** cause, generally best to try again later

- add more debugging statements
- automate handling of transient errors with HTCondor

If **Systemic** cause, need to identify the pattern

- what do all the failed jobs have in common?
- how are the failed jobs different from the successful jobs?

There are ways of getting more information and identifying patterns, but first…

Asking for Help

When to ask for help:

If you have spent a couple of hours actively troubleshooting and have made little to no progress

Who to ask for help:

- Colleagues/peers running similar calculations
- Local facilitators or admins (for system specific problems)
- Internet/broader community
- OSG facilitators (general questions or OSG-specific)
- HTCondor facilitators or developers (for high level HTCondor problems)

Getting more information - jobs

Every job submitted with HTCondor has a set of attributes called "Class Ads"

- For *active* jobs, you can view them using `condor_q`
- For *inactive* jobs, you can view them with `condor_history`

View all ClassAds with the `-l` or `-long` option (best for one job). For example:

```
$ condor_q -long 129.0
AllowedExecuteDuration = 259200
Args = "8"
BytesRecvd = 253.0
BytesSent = 0.0
ClusterId = 129
Cmd = "/home/alice/compare_states"
  ⋮
```

Generally best to pipe into a viewing program:
`condor_q -long JobID | less`

Getting more information - jobs

There are LOTS of ClassAds. But if you know what ClassAd you want to look at, you can have only that value printed with the `-af` or `-autoformat` option

- good for lots of jobs
- Job identifier has to come before the flag

```
$ condor_history 129 -af LastRemoteHost
slot1_2@e2465.chtc.wisc.edu
slot1_78@e2606.chtc.wisc.edu
slot1_4@e2614.chtc.wisc.edu
slot1_2@dsigpu4000.chtc.wisc.edu
slot1_3@gpu4001.chtc.wisc.edu
backfill2_3@gpu2000.chtc.wisc.edu
  ⋮
```

Additional options:
`-af:h` → shows headings
`-af:j` → includes a JobID column
`-af:hj` → both of the above

Getting more information - execution points

You can use `condor_status` to get information about the execution points in the HTC pool

For an overview, use the `-compact` option

Every execution slot has its own set of ClassAds

You can use the `-long`, `-autoformat`, `-const` flags with `condor_status` to look at the slot ClassAds

Questions?

Getting more information - jobs

You can also constrain your search of `condor_q`, `condor_history` to only show jobs that have a particular ClassAd value using the `-constraint` flag

For example, to only show your *active* jobs that are on hold (JobStatus = 5):

```
    condor_q -constraint 'JobStatus == 5'
```

Getting more information - jobs

You can combine these options with some simple shell commands to find patterns:

```
$ condor_q 142 -const 'JobStatus == 5' -af LastRemoteHost | \
    cut -d '@' -f 2 | sort | uniq -c

    193 e2596.chtc.wisc.edu
      1 e4004.chtc.wisc.edu
      1 e4007.chtc.wisc.edu
```

Looks like a problem with this machine. Maybe this machine specifically, or maybe the OS/software environment of this machine?

Schedule

12:30	**Lunch**
13:00	**Part I** ● [15 min] Placing lists of jobs ● [20 min] What happened to my job? ● [15 min] Data placement ● [30 min] Troubleshooting strategies ● [20 min] GPU jobs
14:40	**Break**
14:50	**Part II** ● [40 min] Principles of DAGMan ● [40 min] Hands-on: DAGMan ● [20 min] Python bindings ● [20 min] Hands-on Python bindings ● [10 min] Computing at Nikhef ● [25 min] Philosophy & architecture
17:25	**Social**