Introduction to Job Submission with HTCondor

August 7, 2023

Rachel Lombardi

Slides adapted from Lauren Michaels

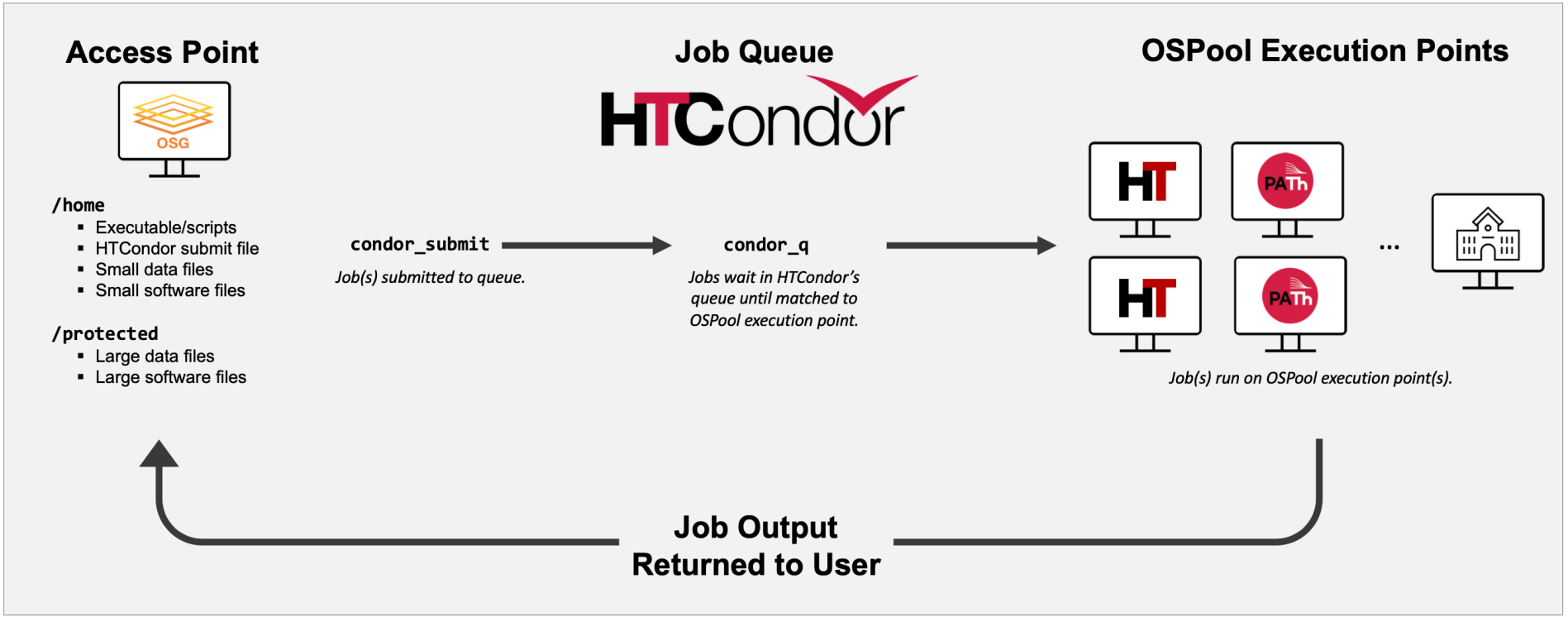
This work was supported by NSF grants MPS-1148698, OAC-1836650, and OAC-2030508

Overview

- How does the HTCondor job scheduler work?
- How do you run, monitor, and review jobs?
- Best ways to submit multiple jobs
- Testing, tuning, and troubleshooting to scale up

Takeaway

HTCondor is a Job Scheduling Software



HISTORY OF HTCONDOR

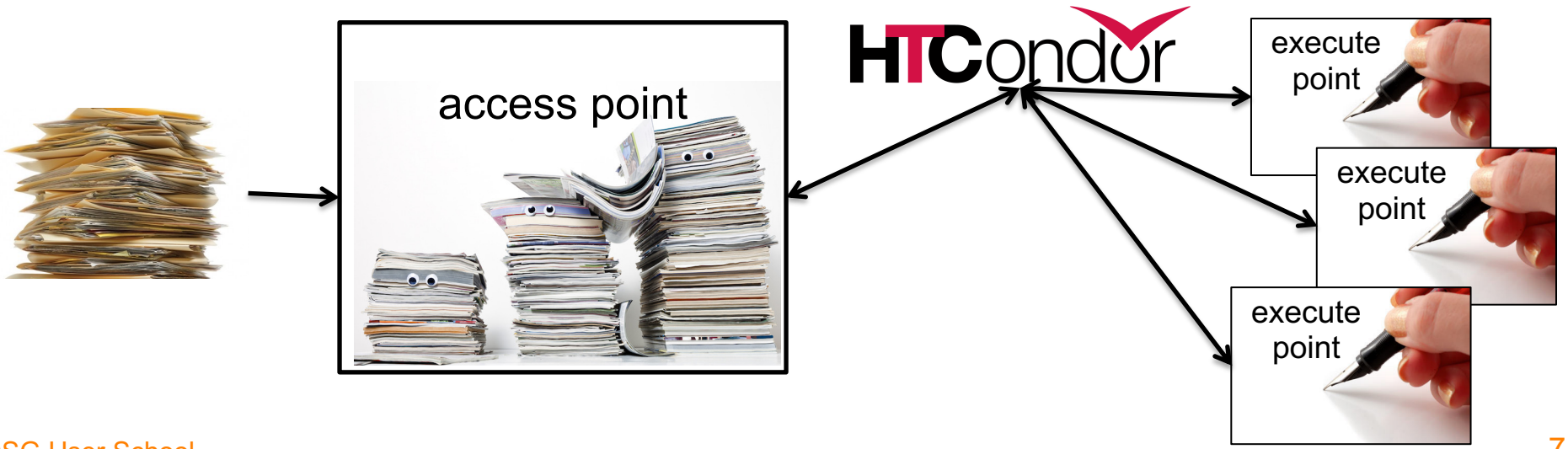
HTCondor History and Status

- History
  - Started in 1988 as a “cycle scavenger”
- Today
  - Developed within the CHTC by professional developers
  - Used all over the world, by:
    - campuses, national labs, Einstein/Folding@Home
    - Dreamworks, Boeing, SpaceX, investment firms, …
    - **The OSG!!**
- Miron Livny
  - Professor, UW-Madison Computer Sciences
  - CHTC Director, OSG Technical Director

HOW DOES HTCONDOR WORK?

HTCondor -- How It Works

- On an ***access point***, you submit tasks to a queue
- HTCondor schedules them to run on computers (***execute points***)



Terminology: *Job*

***Job*: An independently-scheduled unit of computing work**

Three main pieces:

- **Executable:** the script or program to run
- **Input:** any options (arguments) and/or file-based information
- **Output:** files printed by the executable

Note: In order to run *many* jobs, executable must run on the command-line without any graphical input from the user

Terminology: *Machine, Slot*

Machine

- A whole computer (desktop or server)
- Has multiple processors (***CPU cores***), some amount of **memory**, and some amount of file space (**disk**)

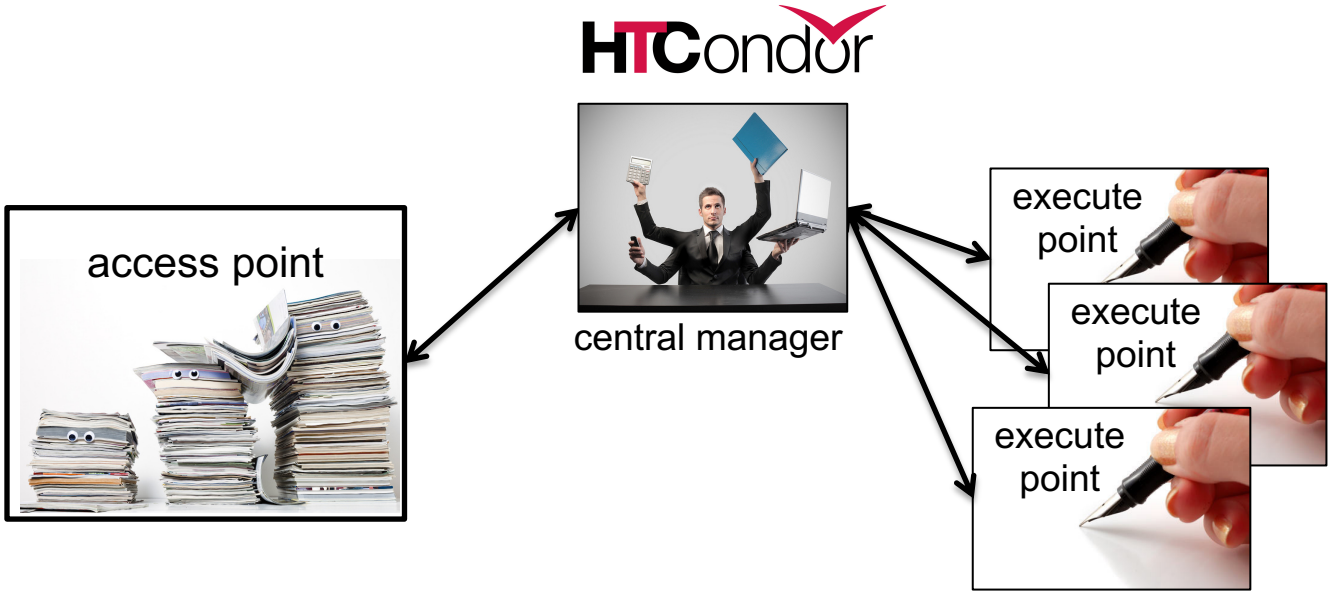
Slot

- **an assignable unit of a machine (i.e. 1 job per slot)**
- may correspond to one core with some memory and disk
- a typical machine will have multiple slots

HTCondor can break up and create new slots, dynamically, as resources become available from completed jobs

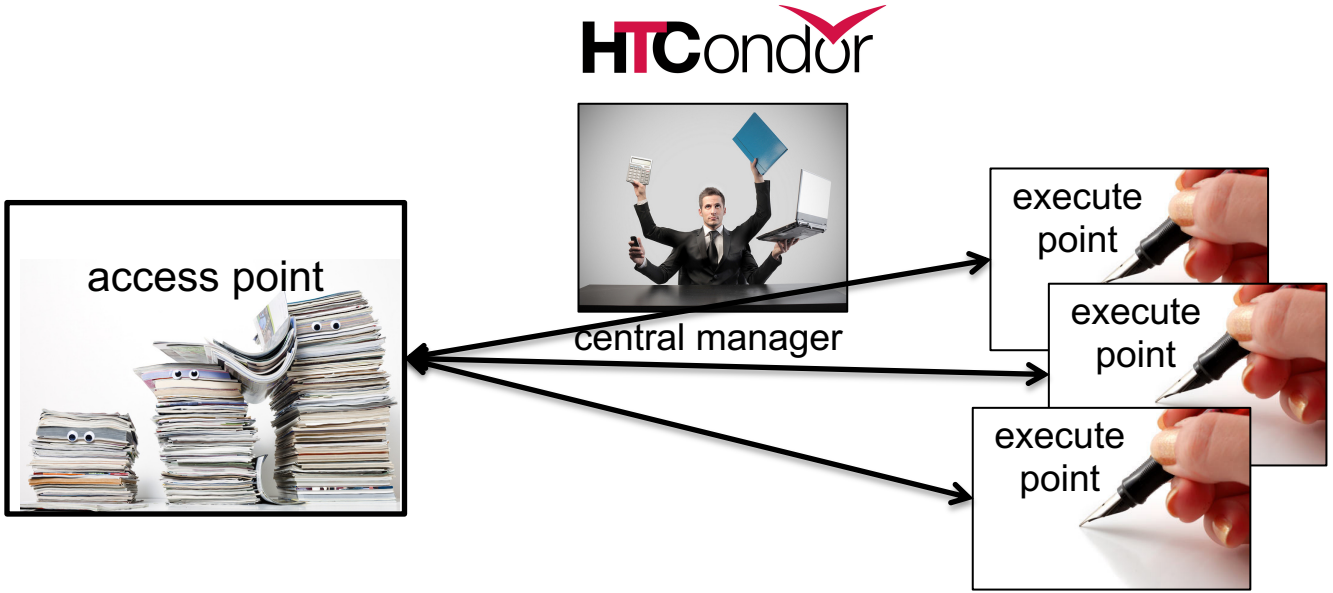
Job Matching

On a regular basis, the ***central manager*** reviews ***Job*** and ***Machine*** attributes and matches jobs to ***Slots***.



Job Execution

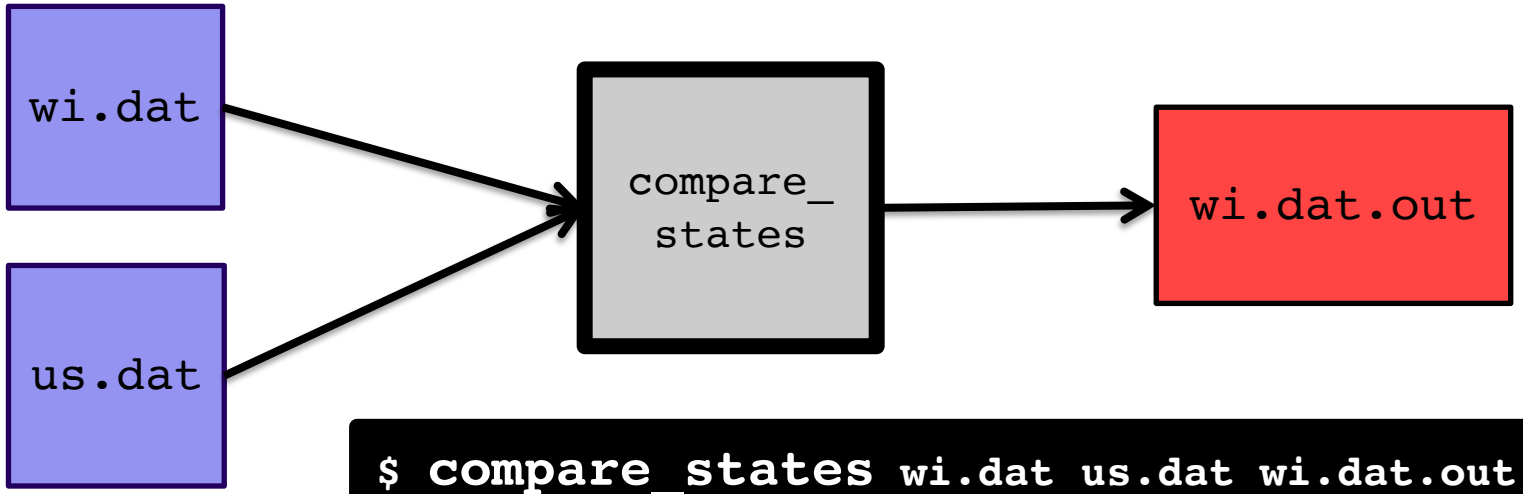
Then the access and execute points communicate directly.



BASICS OF SUBMITTING JOBS

Job Example

Example: program called “compare_states” (executable), which compares two data files (input) and produces a single output file.




```
$ compare_states wi.dat us.dat wi.dat.out
```

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

List your **executable** and any **arguments** it takes

Arguments are any options passed to the executable from the command line

```
$ compare_states wi.dat us.dat wi.dat.out
```

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

Provide HTCondor a comma-separated list of **input files to transfer** to the slot

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

HTCondor will transfer back all new and changed files (output) from the job, automatically.

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

log: file created by HTCondor to track job progress

- *Explored in exercises!*

output/error: captures stdout and stderr from your program (what would otherwise be printed to the terminal)

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

request_cpus, request_disk, request_memory:

the resources your job needs.

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

Very important to request appropriate resources (***memory***, ***cpus***, ***disk***)

- **requesting too little**: causes problems for your jobs; jobs might by ‘held’ by HTCondor
- **requesting too much:** jobs will match to fewer “slots” than they could, and you’ll block other jobs

Basic Submit File

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

queue: keyword indicating the number of jobs to queue

- must be the last line of the submit file

- has different syntax options we will learn later!

SUBMITTING AND MONITORING HTCONDOR JOBS

Submitting and Monitoring

- To submit a job/jobs: **`condor_submit`** ***`submit_file`***
- To monitor submitted jobs: **`condor_q`**

```
$ condor_submit job.submit
Submitting job(s).
1 job(s) submitted to cluster 128.

$ condor_q
-- Schedd: learn.chtc.wisc.edu : <128.104.101.92> @ 05/01/22 10:35:54
OWNER  BATCH_NAME              SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
alice  CMD: compare_states   5/9  11:05      _      _      1      1 128.0

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

HTCondor Manual: condor_submit
HTCondor Manual: condor_q

More about `condor_q`

- By default, **`condor_q`** shows <u>your jobs only</u> and <u>batches</u> jobs that were submitted together:

```
$ condor_q
-- Schedd: learn.chtc.wisc.edu : <128.104.101.92> @ 05/01/22 10:35:54
OWNER  BATCH_NAME              SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
alice  CMD: compare_states   5/9  11:05      _      _      1      1 128.0

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

JobId = **ClusterID.ProcID**

- Limit **`condor_q`** by username, *`ClusterId`* or full *`JobId`*, (denoted `[U/C/J]` in following slides).

More about condor_q

- To see individual job details, use:

  condor_q –nobatch

```
$ condor_q -nobatch
-- Schedd: learn.chtc.wisc.edu : <128.104.101.92>
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 I  0    0.0 compare_states
128.1        alice      5/9  11:09   0+00:00:00 I  0    0.0 compare_states
...

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

- We will use the **-nobatch** option in the following slides to see extra detail about what is happening with a job

OBSERVING JOB STATES WITH CONDOR_Q

Job Idle

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92>
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 I  0   0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

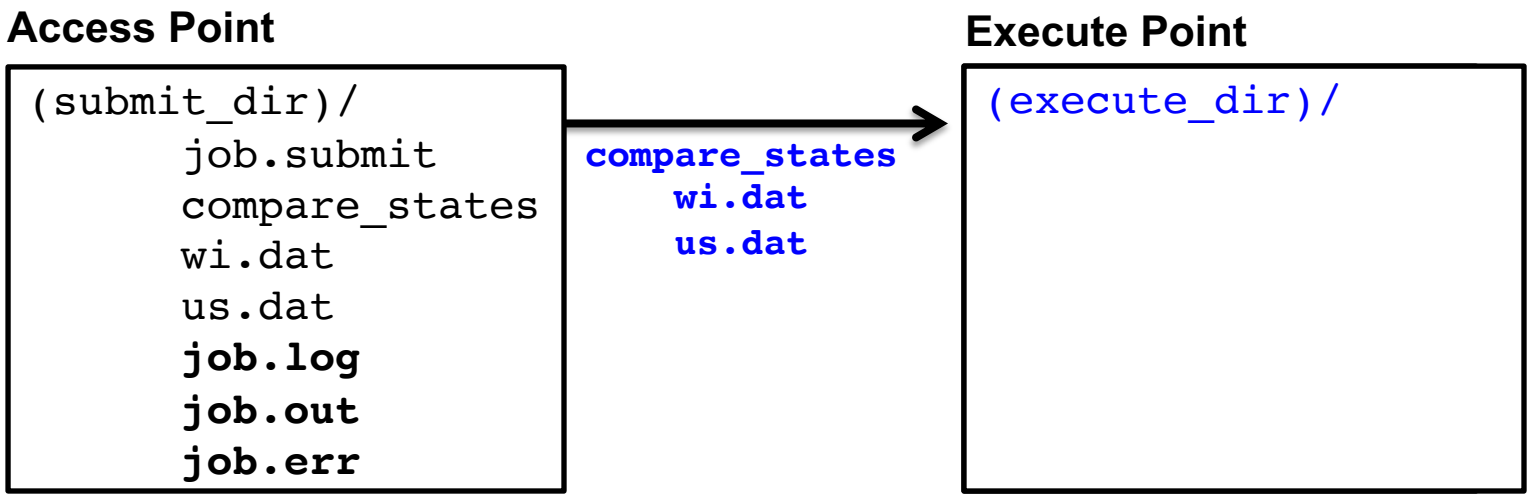
Access Point

```
(submit_dir)/
        job.submit
        compare_states
        wi.dat
        us.dat
        job.log
        job.out
        job.err
```

Job Starts

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618>
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09    0+00:00:00 <  0   0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```



Job Running

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92>
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:01:03 R  0   0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```

Access Point

```
(submit_dir)/
        job.submit
        compare_states
        wi.dat
        us.dat
        job.log
        job.out
        job.err
```

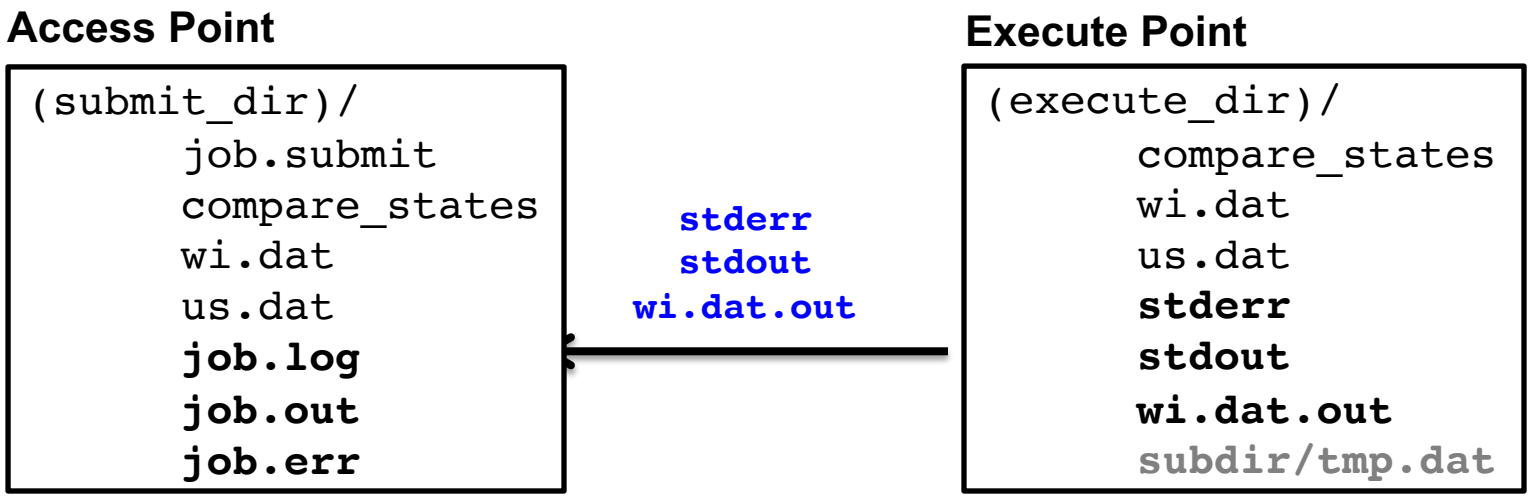
Execute Point

```
(execute_dir)/
        compare_states
        wi.dat
        us.dat
        stderr
        stdout
        wi.dat.out
        subdir/tmp.dat
```

Job Completes

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92>
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128          alice      5/9  11:09   0+00:02:02 >  0    0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```



Job Completes (cont.)

```
$ condor_q -nobatch

-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
```

Access Point

```
(submit_dir)/
        job.submit
        compare_states
        wi.dat
        us.dat
        job.log
        job.out
        job.err
        wi.dat.out
```

REVIEWING COMPLETED JOBS

Log File

```
000 (128.000.000) 05/09 11:09:08 Job submitted from host: <128.104.101.92&sock=6423_b881_3>
...
001 (128.000.000) 05/09 11:10:46 Job executing on host: <128.104.101.128:9618&sock=5053_3126_3>
...
006 (128.000.000) 05/09 11:10:54 Image size of job updated: 220
        1  -  MemoryUsage of job (MB)
        220  -  ResidentSetSize of job (KB)
...
005 (128.000.000) 05/09 11:12:48 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
        0  -  Run Bytes Sent By Job
        33  -  Run Bytes Received By Job
        0  -  Total Bytes Sent By Job
        33  -  Total Bytes Received By Job
        Partitionable Resources :    Usage  Request Allocated
           Cpus                 :                1         1
           Disk (KB)            :       14    20480  17203728
           Memory (MB)          :        1       20        20
```

Reviewing Jobs

- To review a large group of jobs at once, use **`condor_history`**

As **`condor_q`** is to the present, **`condor_history`** is to the past

```
$ condor_history alice
 ID      OWNER    SUBMITTED   RUN_TIME    ST  COMPLETED   CMD
189.1012 alice    5/11 09:52   0+00:07:37 C   5/11 16:00 /home/alice
189.1002 alice    5/11 09:52   0+00:08:03 C   5/11 16:00 /home/alice
189.1081 alice    5/11 09:52   0+00:03:16 C   5/11 16:00 /home/alice
189.944  alice    5/11 09:52   0+00:11:15 C   5/11 16:00 /home/alice
189.659  alice    5/11 09:52   0+00:26:56 C   5/11 16:00 /home/alice
189.653  alice    5/11 09:52   0+00:27:07 C   5/11 16:00 /home/alice
189.1040 alice    5/11 09:52   0+00:05:15 C   5/11 15:59 /home/alice
189.1003 alice    5/11 09:52   0+00:07:38 C   5/11 15:59 /home/alice
189.962  alice    5/11 09:52   0+00:09:36 C   5/11 15:59 /home/alice
189.961  alice    5/11 09:52   0+00:09:43 C   5/11 15:59 /home/alice
189.898  alice    5/11 09:52   0+00:13:47 C   5/11 15:59 /home/alice
```


HTCondor Manual: condor_history

QUESTIONS?