CERN

# LSF – HTCondor migration

Ben Jones IT-CM-IS

# Agenda

- Batch Service
- Why exit LSF?
- What is HTCondor?
- Benefits of HTCondor
- Timescale
- How can IT help?
- Usage patterns
- Questions

# Batch Service

- Service used for both grid and “local” submission, with HPC on the way
- Local means open to all CERN users, kerberos, shared filesystem, managed submission nodes
- ~100k cores in LSF pools
- ~50k cores in HTCondor
  - Till now just grid
- ~800k jobs per day

# Why exit LSF?

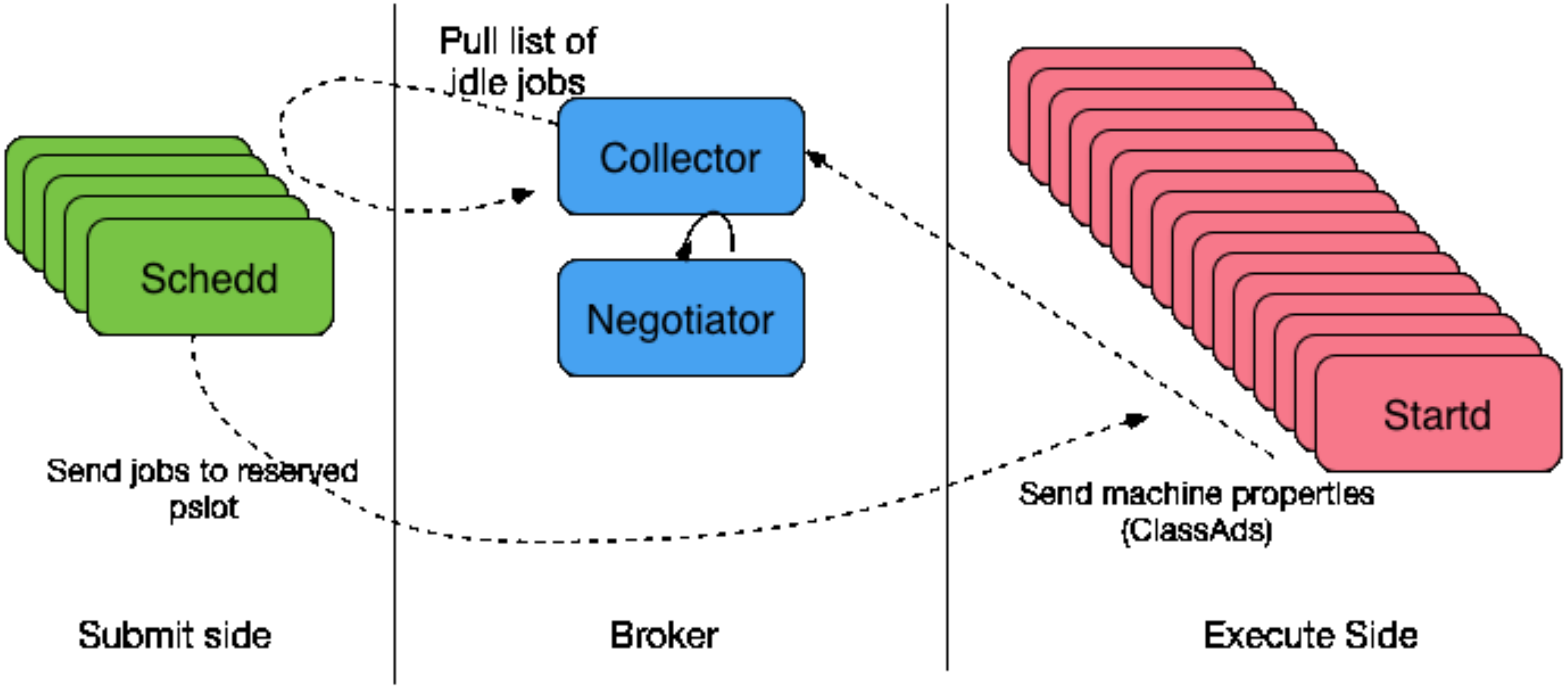
- Proprietary product
- Limits to number of nodes (>5K not advisable)
- Doesn’t scale very well past 180K jobs
- Slow queries, submission
  - All goes through one master
- Security model limits flexibility of submission hosts
- Product seems to be diverging from our use case
  - Scaling into machines, rather than jobs + nodes

# What is HTCondor

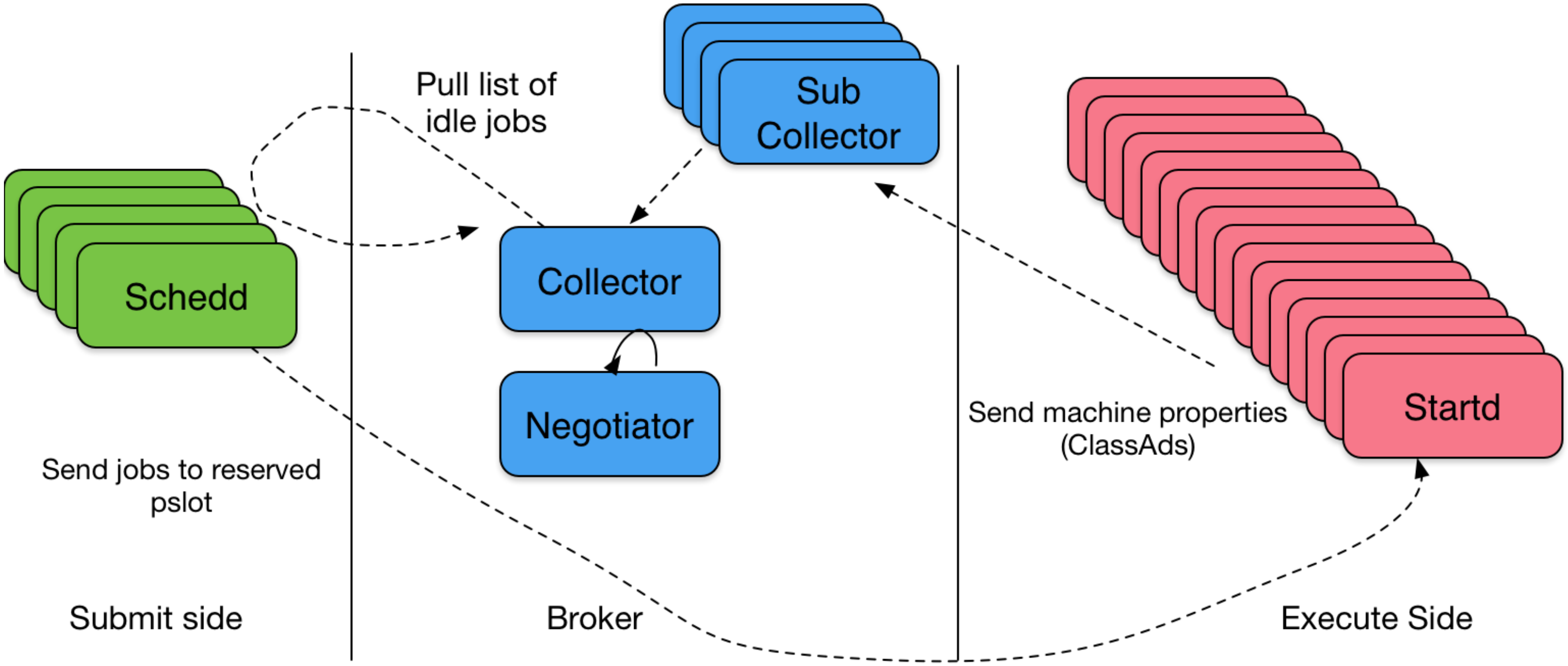
High Throughput Computing

- Open Source batch system developed at the CHTC at the University of Wisconsin
- “High Throughput Computing”
- Long history in HEP and elsewhere (including previously at CERN)
- Used extensively in OSG, and things like the CMS global pool (160K++ cores)
- System of symmetric matching of job requests to resources using ClassAds of job requirements and machine resources

# Benefits: scalability



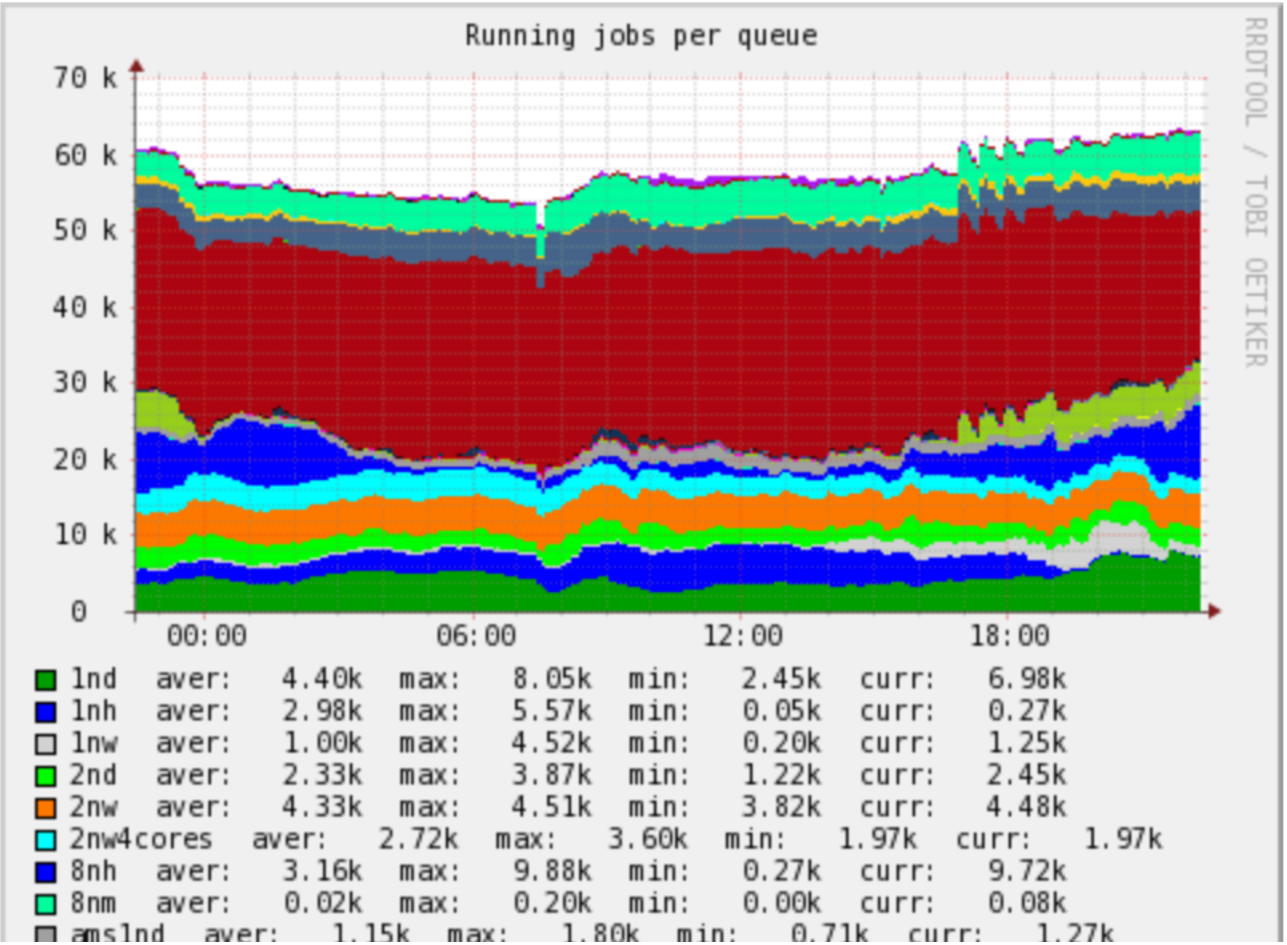
# Split the Collectors



# Benefits: Flexibility

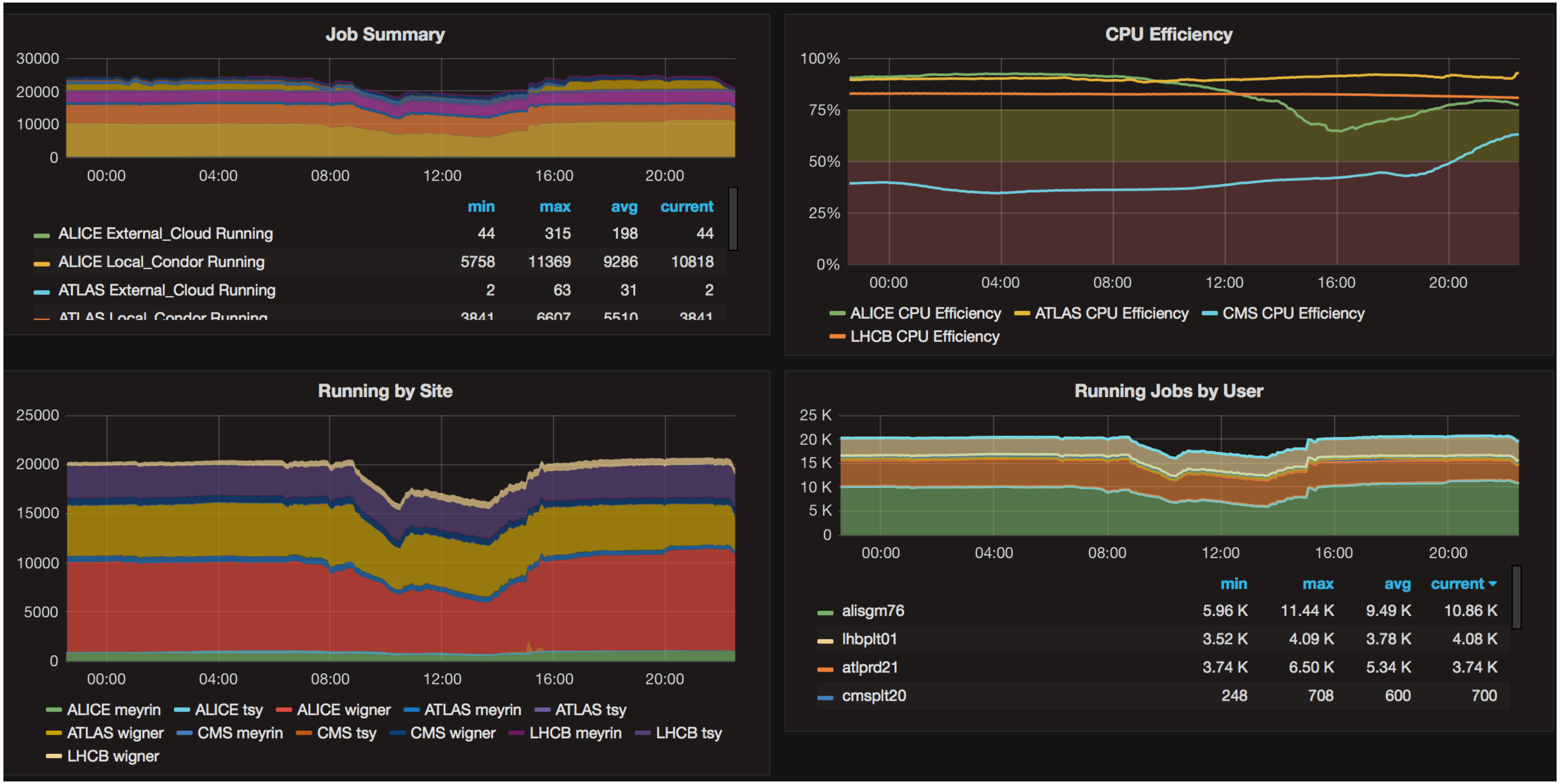
- Extra “Universes”
  - Docker, Parallel as well as Vanilla
- DAGs
  - Job dependencies between different submit files
- Condor-G to submit to other systems
  - For example, condor submission to boinc
- Flexible configuration allows routes to clouds, or specific resources, or HPC
- HTCondor can be a single frontend to have jobs run in many different ways on different systems
- Cgroups to ensure jobs can coexist without stepping on each others’ resources

# Out with the old…



```
🟩 1nd       aver:   4.40k   max:   8.05k   min:   2.45k   curr:   6.98k
🟦 1nh       aver:   2.98k   max:   5.57k   min:   0.05k   curr:   0.27k
⬜ 1nw       aver:   1.00k   max:   4.52k   min:   0.20k   curr:   1.25k
🟩 2nd       aver:   2.33k   max:   3.87k   min:   1.22k   curr:   2.45k
🟧 2nw       aver:   4.33k   max:   4.51k   min:   3.82k   curr:   4.48k
🟦 2nw4cores   aver:   2.72k   max:   3.60k   min:   1.97k   curr:   1.97k
🟦 8nh       aver:   3.16k   max:   9.88k   min:   0.27k   curr:   9.72k
🟩 8nm       aver:   0.02k   max:   0.20k   min:   0.00k   curr:   0.08k
⬜ ams1nd      aver:   1.15k   max:   1.80k   min:   0.71k   curr:   1.27k
```

# Benefits: community



Job Summary

| | min | max | avg | current |
|---|---|---|---|---|
| ALICE External_Cloud Running | 44 | 315 | 198 | 44 |
| ALICE Local_Condor Running | 5758 | 11369 | 9286 | 10818 |
| ATLAS External_Cloud Running | 2 | 63 | 31 | 2 |
| ATLAS Local_Condor Running | 3841 | 6607 | 5510 | 3841 |

Running Jobs by User

| | min | max | avg | current |
|---|---|---|---|---|
| alisgm76 | 5.96 K | 11.44 K | 9.49 K | 10.86 K |
| lhbplt01 | 3.52 K | 4.09 K | 3.78 K | 4.08 K |
| atlprd21 | 3.74 K | 6.50 K | 5.34 K | 3.74 K |
| cmsplt20 | 248 | 708 | 600 | 700 |

# Timescale

- Grid is prod since November
- Local required work with upstream for kerberos renewal, now no technical issues
- IBM support till end of 2017
- IT support for LSF till end of Run 2

# How can IT help?

- Some help available with migration
  - We can help advise on submission scripts etc
- Migration can be easy for most use cases
- Documentation and tutorial available at http://cern.ch/batchdocs
- batch-operations@cern.ch / SNOW to batch team / contact us directly

# Differences with LSF

- There are no queues
  - You just submit jobs – we do ask for time requirements (more later)
- Time is measured / limited / charged in Wall
  - No CPU time means no normalisation to consider
    - No more “1 normalised hour” (currently avg 20 minutes)
- Rather than queues, jobs submitted with a maxRuntime
  - Specified either with a +JobFlavour or +maxRuntime
  - More capacity for shorter jobs < 25h and less for v long < 1wk

# Memory limits

- Jobs are assigned slots with scaled 2gb / core
- CGroups enforce memory limits
  - Soft limit
  - Processes are swapped to disk if machine has memory pressure
  - If remaining process has RSS > RequestMemory, it is killed
- You can request > 2gb per job!
  - [but you will get > 1 core]

# Job Differences

- You need to write a submit file
  - They’re easy, reusable, and powerful
- Can’t submit a job from a job
  - Unless that first job is a DAG!
  - Complex workflows can be expressed using DAGs
- No array jobs
  - A submit file can submit multiple jobs
  - Many ways to control behaviour of multiple jobs

# Things that haven’t changed

- Shared filesystems
  - AFS, EOS, CVMFS available
  - AFS can be used for submission working dir as per LSF
  - EOS FUSE in future
- Jobs have access to Kerberos/AFS tokens
- Fairshare works in broadly same way
- Job writes to local scratch directory by default

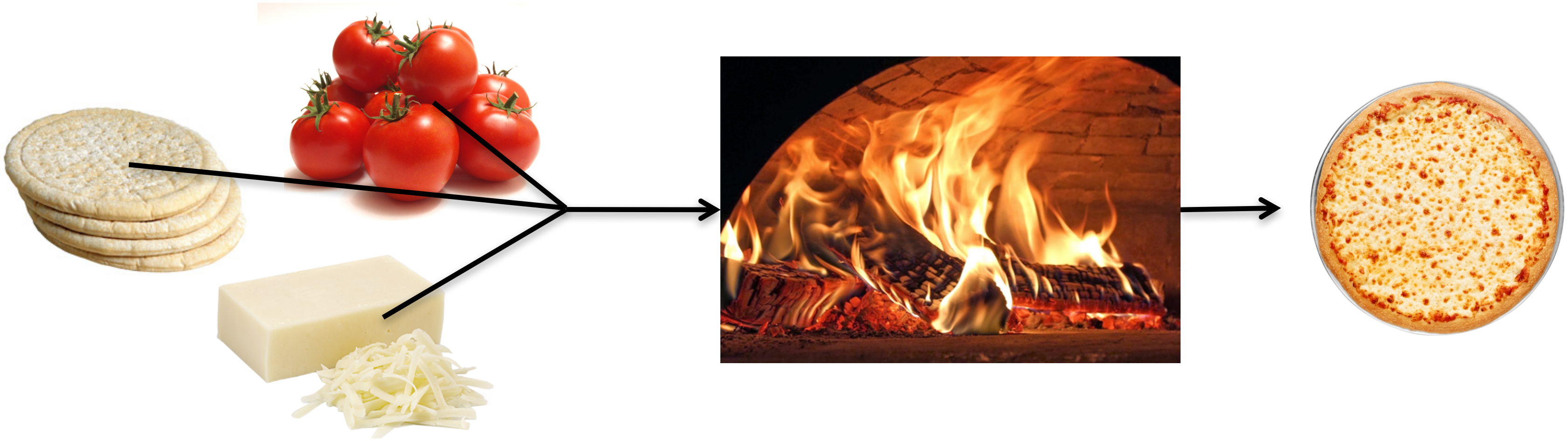
# Questions so far?

# Running a Job with HTCondor

[slides from CHTC at University of Wisconsin]

# Jobs

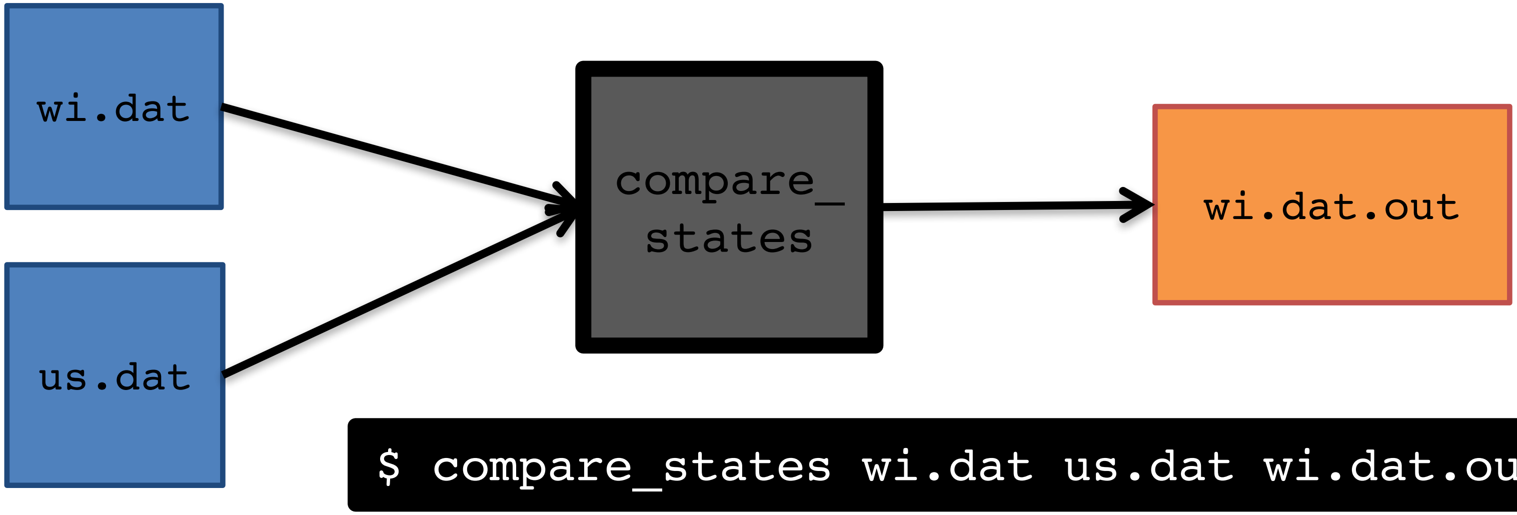
- A single computing task is called a “job”
- Three main pieces of a job are the input, executable (program) and output



- Executable must be runnable from the command line without any interactive input

# Job Example

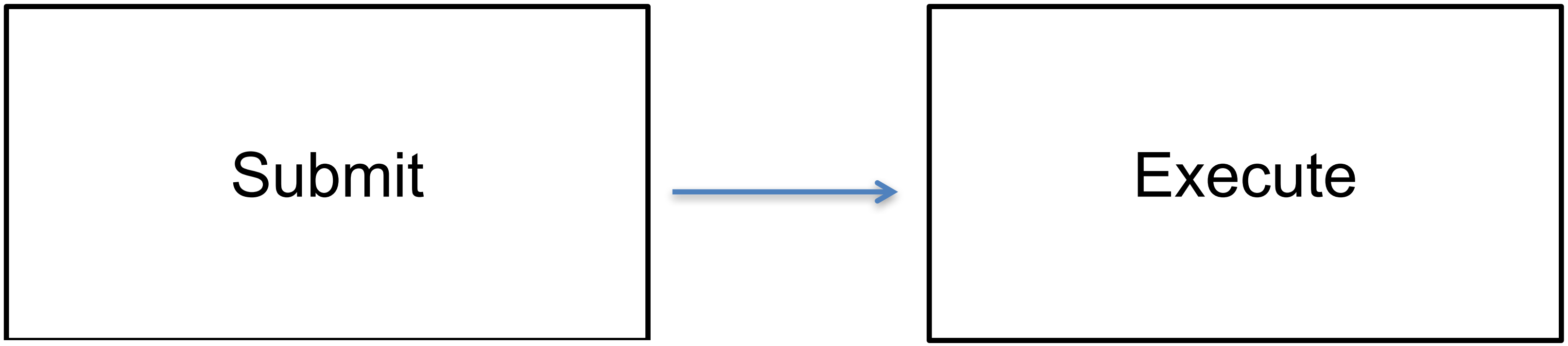
- For our example, we will be using an imaginary program called “compare_states”, which compares two data files and produces a single output file.



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

# File Transfer

- Our example will use HTCondor's file transfer option:



Submit

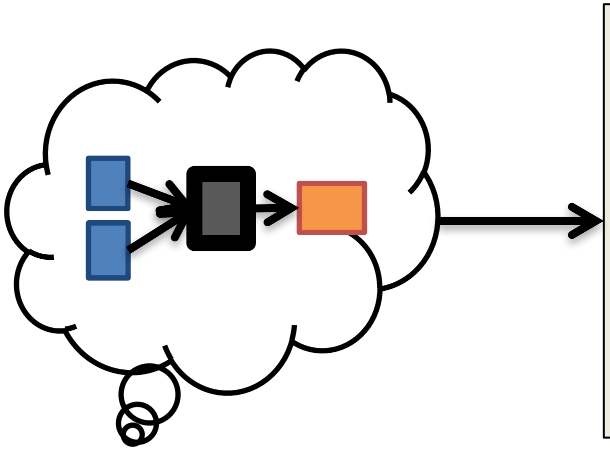
Execute

```
(submit_dir)/
   input files
   executable
```

```
(execute_dir)/
   output files
```

# Job Translation

- Submit file: communicates everything about your job(s) to HTCondor



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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

HTCondor

# Submit File

job.submit

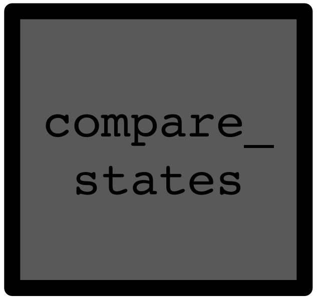
```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

# Submit File

job.submit

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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.

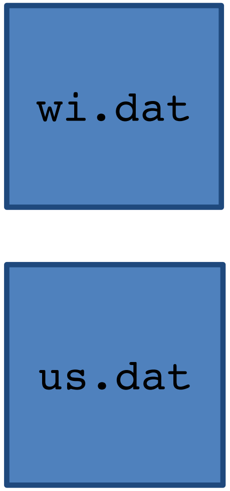
```
$ compare_states wi.dat us.dat wi.dat.out
```

# Submit File

job.submit

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- Indicate your input files.



wi.dat

us.dat

# Submit File

job.submit

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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 (usually output) from the job.

wi.dat.out

# Submit File

job.submit

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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
- `output`/`error`: captures stdout and stderr

# Submit File

job.submit

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- Request the appropriate resources for your job to run.
- `queue`: keyword indicating "create a job."

# Submitting and Monitoring

- To submit a job/jobs:

  **`condor_submit`** ***`submit_file_name`***

- To monitor submitted jobs, use:

  **`condor_q`**

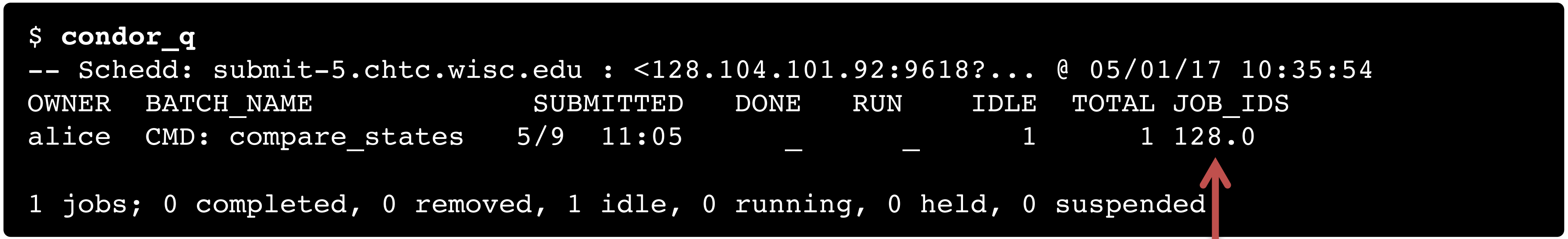
```
$ condor_submit job.submit
Submitting job(s).
1 job(s) submitted to cluster 128.
```

```
$ condor_q
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?... @ 05/01/17 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:
  - user’s job only (as of 8.6)
  - jobs summarized in “batches” (as of 8.6)
- Constrain with username, `ClusterId` or full `JobId`, which will be denoted `[U/C/J]` in the following slides



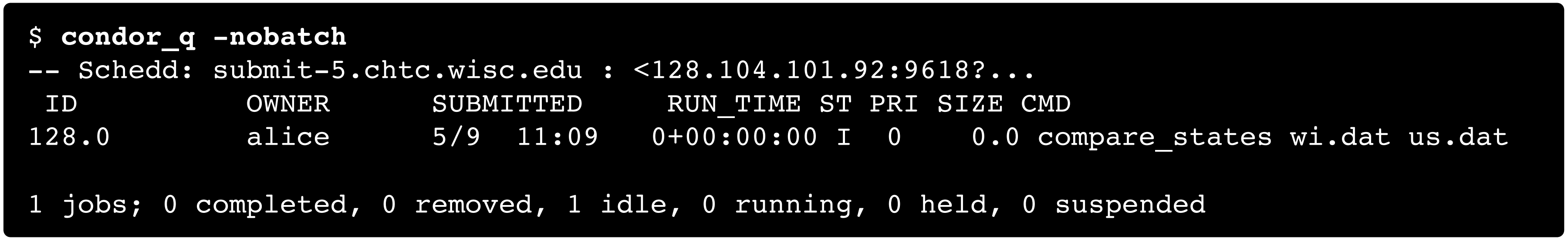
```
$ condor_q
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?... @ 05/01/17 10:35:54
OWNER  BATCH_NAME             SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
alice  CMD: compare_states   5/9  11:05      _      _      1      1 128.0

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

`JobId` = `ClusterId` .`ProcId`

# More about condor_q

- To see individual job information, use:

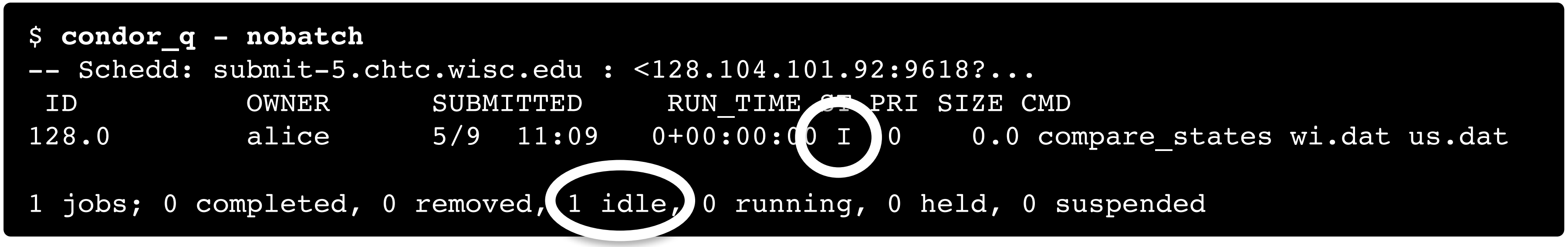
  **condor_q -nobatch**



```
$ 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 I  0    0.0 compare_states wi.dat us.dat

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

# Job Idle

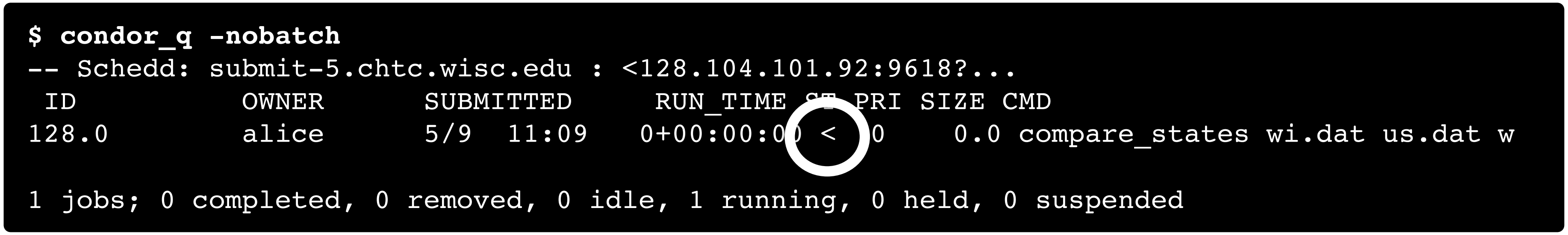


```
$ condor_q - nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 I  0    0.0 compare_states wi.dat us.dat

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

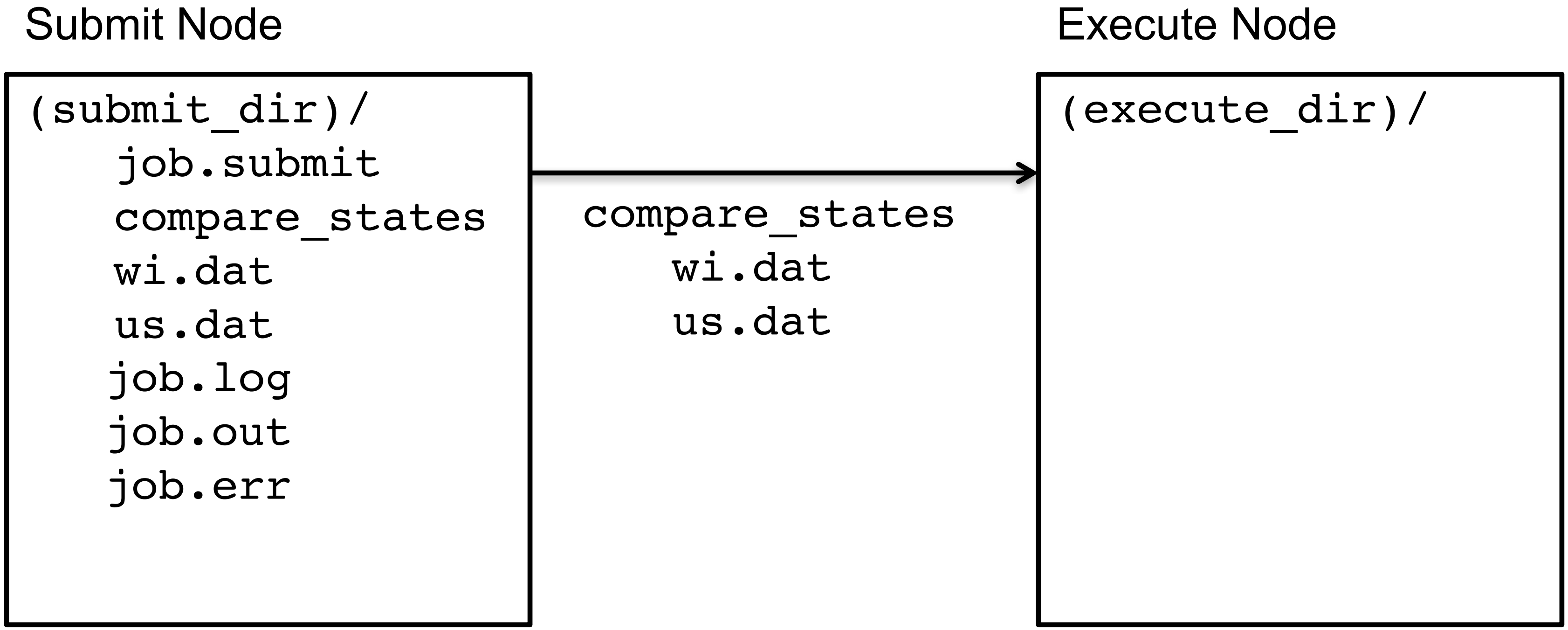
Submit Node

```
(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 w

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



Submit Node

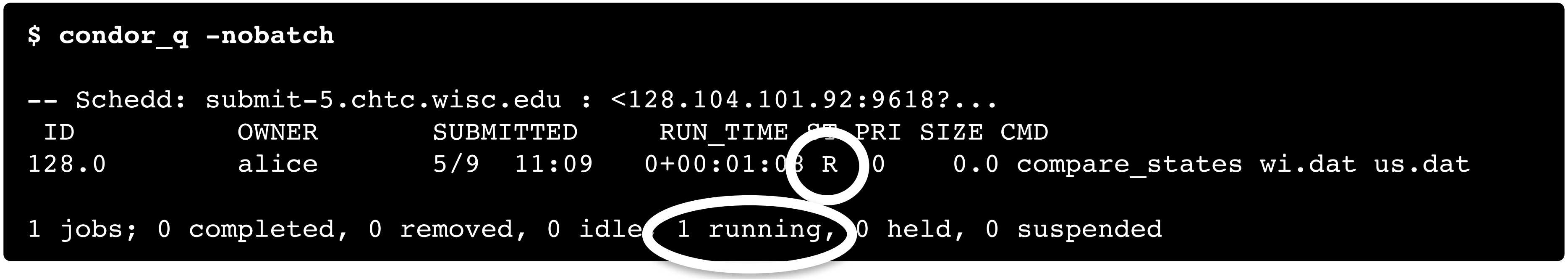
```
(submit_dir)/
    job.submit
    compare_states
    wi.dat
    us.dat
    job.log
    job.out
    job.err
```

```
compare_states
    wi.dat
    us.dat
```

Execute Node

```
(execute_dir)/
```

# Job Running



```
$ 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+00:01:03 R  0    0.0 compare_states wi.dat us.dat

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

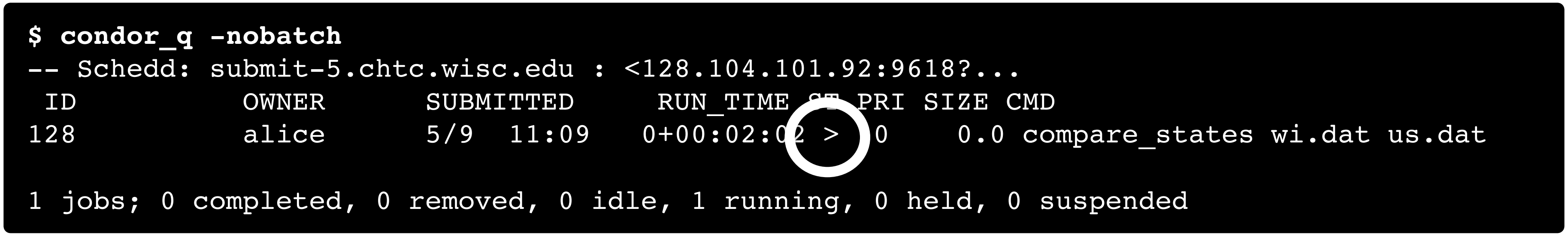
Submit Node

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

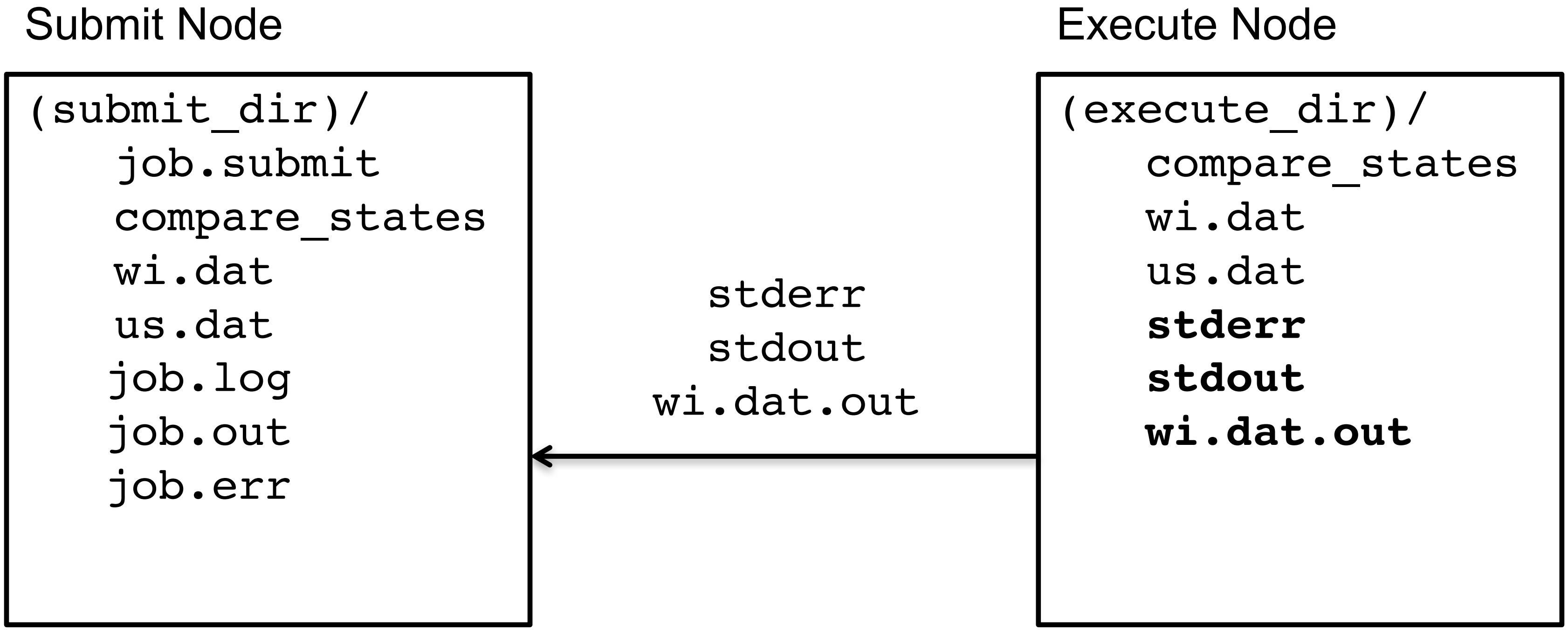
Execute Node

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

# Job Completes



```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 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
```

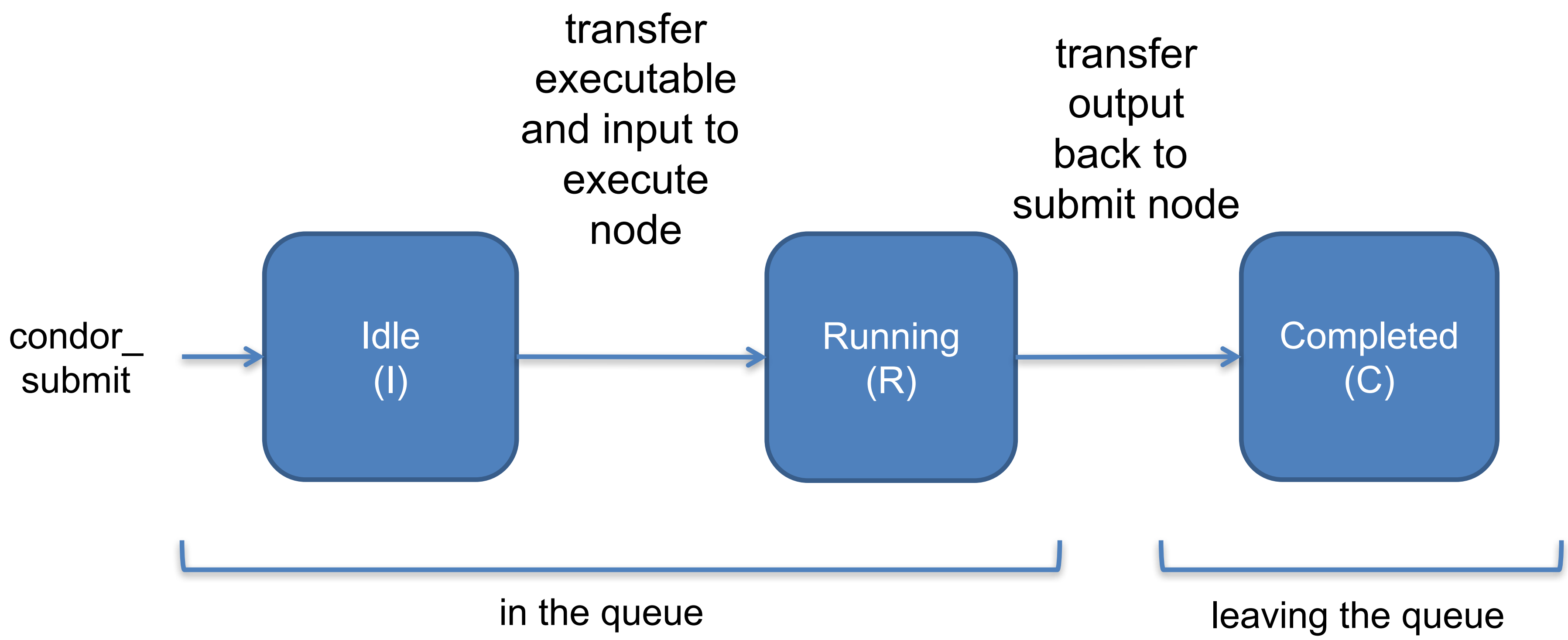
Submit Node

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

# 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
```

# Job States



condor_ submit

Idle (I)

transfer executable and input to execute node

Running (R)

transfer output back to submit node

Completed (C)

in the queue

leaving the queue

# Assumptions

- Aspects of your submit file may be dictated by infrastructure and configuration
- For example: file transfer
  - previous example assumed files would need to be transferred between submit/execute

    ```
    should_transfer_files = YES
    ```

  - not the case with a shared filesystem

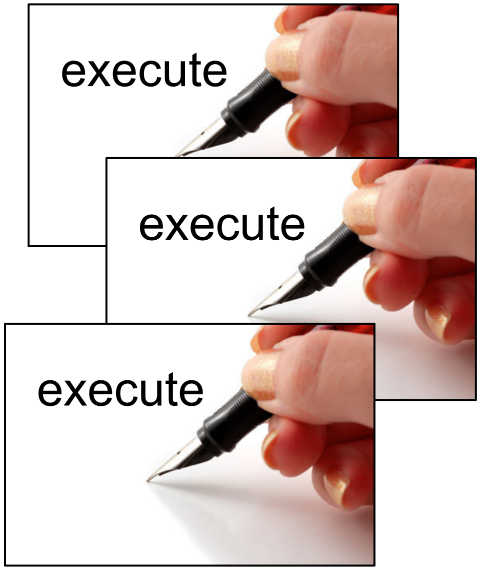
    ```
    should_transfer_files = NO
    ```

# Job Matching and Class Ad Attributes

# The Central Manager

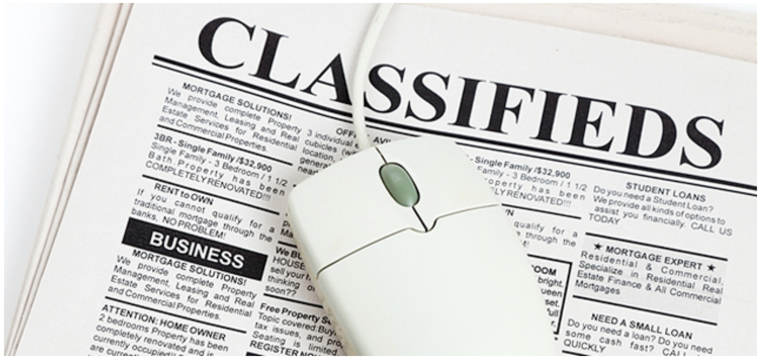
- HTCondor matches jobs with computers via a “central manager”.

central manager



# Class Ads

- HTCondor stores a list of information about each job and each computer.
- This information is stored as a “Class Ad”



- Class Ads have the format:

```
AttributeName = value
```

can be a boolean, number, or string

HTCondor Manual: Appendix A: Class Ad Attributes

# Job Class Ad

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

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

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

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

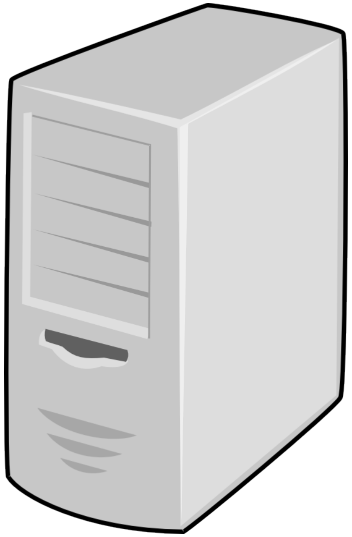
+

HTCondor configuration*

=

```
RequestCpus = 1
Err = "job.err"
WhenToTransferOutput = "ON_EXIT"
TargetType = "Machine"
Cmd =
"/home/alice/tests/htcondor_week/compar
e_states"
JobUniverse = 5
Iwd = "/home/alice/tests/htcondor_week"
RequestDisk = 20480
NumJobStarts = 0
WantRemoteIO = true
OnExitRemove = true
TransferInput = "us.dat,wi.dat"
MyType = "Job"
Out = "job.out"
UserLog =
"/home/alice/tests/htcondor_week/job.lo
g"
RequestMemory = 20
...

...
```

# Computer "Machine" Class Ad



+

HTCondor configuration

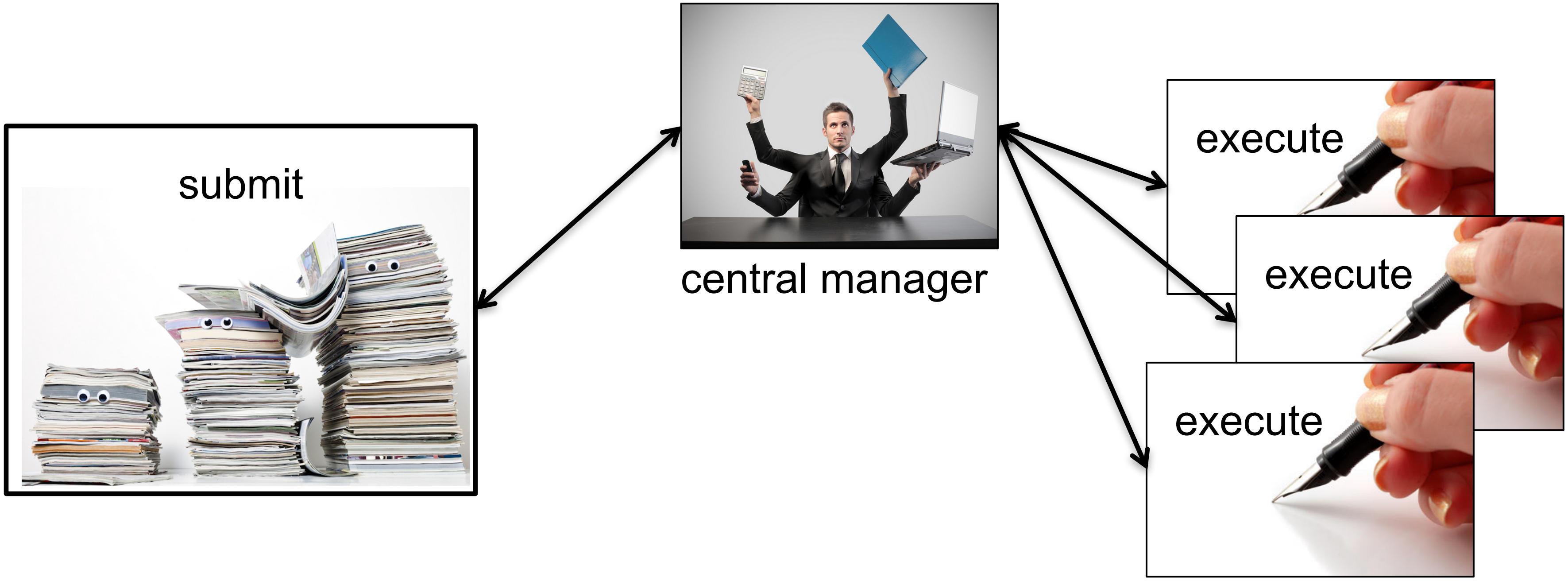
=

```
HasFileTransfer = true
DynamicSlot = true
TotalSlotDisk = 4300218.0
TargetType = "Job"
TotalSlotMemory = 2048
Mips = 17902
Memory = 2048
UtsnameSysname = "Linux"
MAX_PREEMPT = ( 3600 * 72 )
Requirements = ( START ) && (
IsValidCheckpointPlatform ) && (
WithinResourceLimits )
OpSysMajorVer = 6
TotalMemory = 9889
HasGluster = true
OpSysName = "SL"
HasDocker = true

...
```

# Job Matching

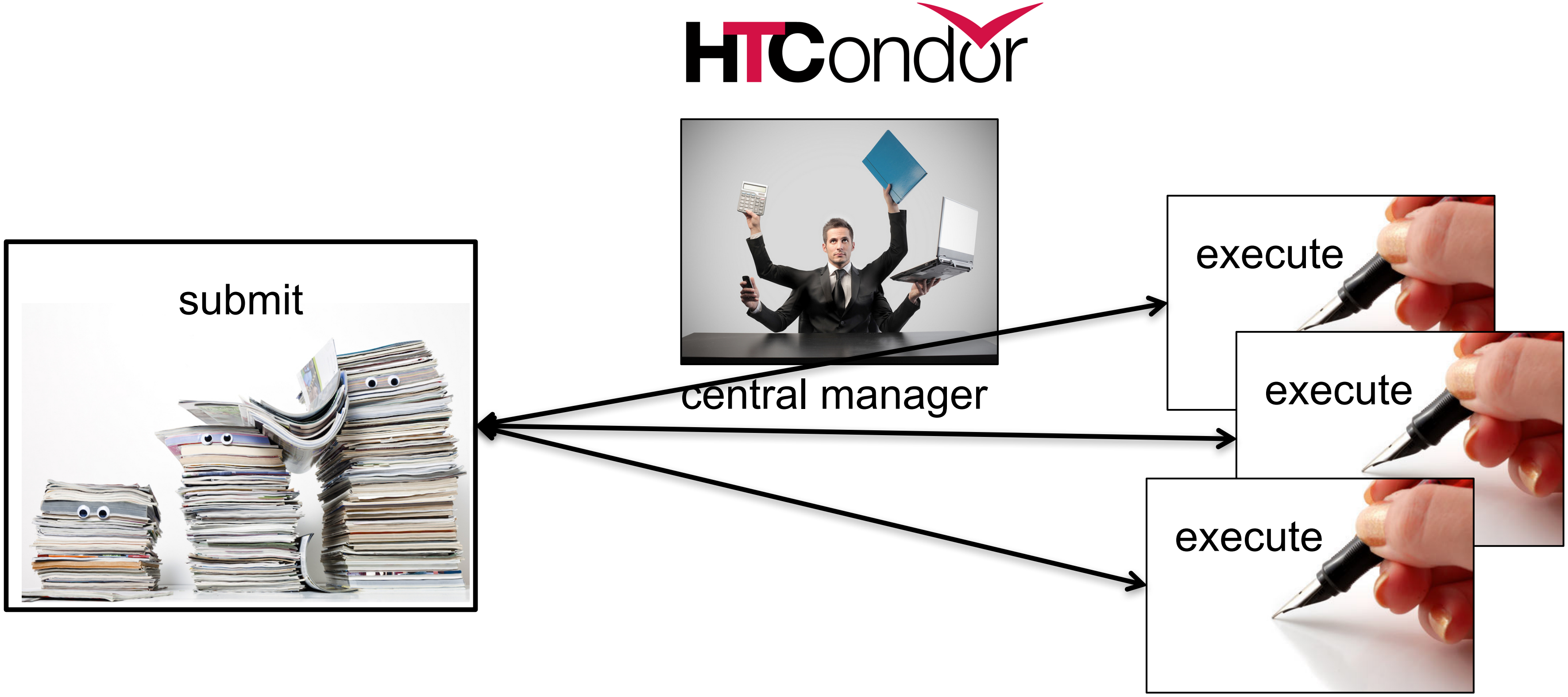
- On a regular basis, the central manager reviews Job and Machine Class Ads and matches jobs to computers.

HTCondor



# Job Execution

- (Then the submit and execute points communicate directly.)



# Class Ads for People

- Class Ads also provide lots of useful information about jobs and computers to HTCondor users and administrators

# Finding Job Attributes

- Use the “long” option for condor_q
  **condor_q *-l JobId***

```
$ condor_q -l 128.0
WhenToTransferOutput = "ON_EXIT"
TargetType = "Machine"
Cmd = "/home/alice/tests/htcondor_week/compare_states"
JobUniverse = 5
Iwd = "/home/alice/tests/htcondor_week"
RequestDisk = 20480
NumJobStarts = 0
WantRemoteIO = true
OnExitRemove = true
TransferInput = "us.dat,wi.dat"
MyType = "Job”
UserLog = "/home/alice/tests/htcondor_week/job.log"
RequestMemory = 20
...
```

# Useful Job Attributes

- `UserLog`: location of job log
- `Iwd`: Initial Working Directory (i.e. submission directory) on submit node
- `MemoryUsage`: maximum memory the job has used
- `RemoteHost`: where the job is running
- `BatchName`: attribute to label job batches
- ...and more

# Displaying Job Attributes

- Use the “auto-format” option:

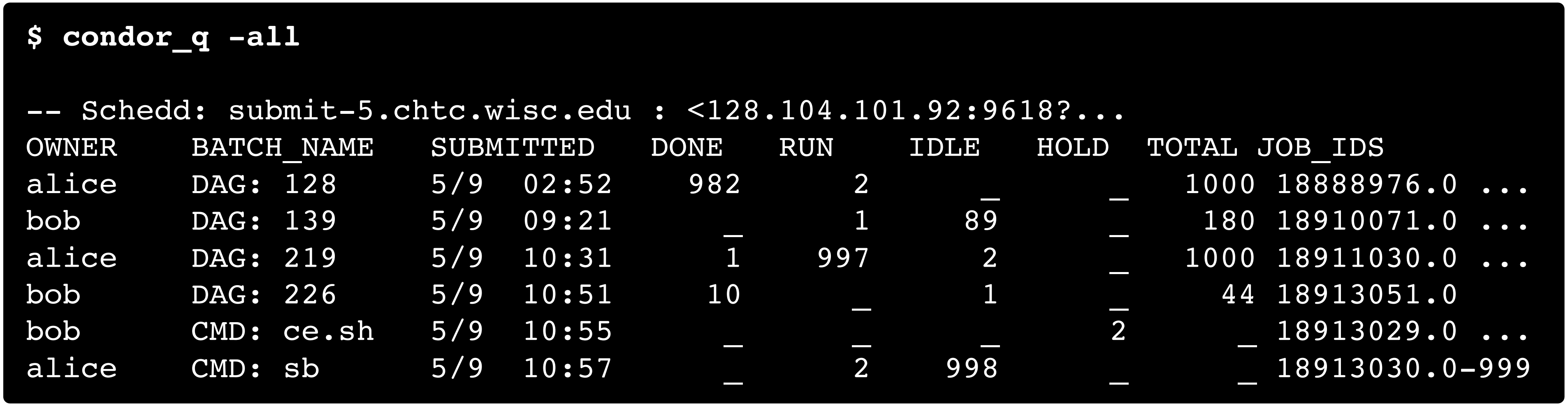
```
condor_q [U/C/J] -af Attribute1 Attribute2 ...
```

```
$ condor_q -af ClusterId ProcId RemoteHost MemoryUsage

17315225 116 slot1_1@e092.chtc.wisc.edu 1709
17315225 118 slot1_2@e093.chtc.wisc.edu 1709
17315225 137 slot1_8@e125.chtc.wisc.edu 1709
17315225 139 slot1_7@e121.chtc.wisc.edu 1709
18050961 0 slot1_5@c025.chtc.wisc.edu 196
18050963 0 slot1_3@atlas10.chtc.wisc.edu 269
18050964 0 slot1_25@e348.chtc.wisc.edu 245
18050965 0 slot1_23@e305.chtc.wisc.edu 196
18050971 0 slot1_6@e176.chtc.wisc.edu 220
```

# Other Displays

- See the whole queue (all users, all jobs)

  **`condor_q -all`**



```
$ condor_q -all

-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
OWNER    BATCH_NAME   SUBMITTED   DONE   RUN    IDLE   HOLD  TOTAL JOB_IDS
alice    DAG: 128     5/9  02:52    982      2      _      _   1000 18888976.0 ...
bob      DAG: 139     5/9  09:21      _      1     89      _    180 18910071.0 ...
alice    DAG: 219     5/9  10:31      1    997      2      _   1000 18911030.0 ...
bob      DAG: 226     5/9  10:51     10      _      1      _     44 18913051.0
bob      CMD: ce.sh   5/9  10:55      _      _      _      2      _ 18913029.0 ...
alice    CMD: sb      5/9  10:57      _      2    998      _      _ 18913030.0-999
```

# `condor_q` Reminder

- Default output is batched jobs
  - Batches can be grouped manually using the `JobBatchName` attribute in a submit file:

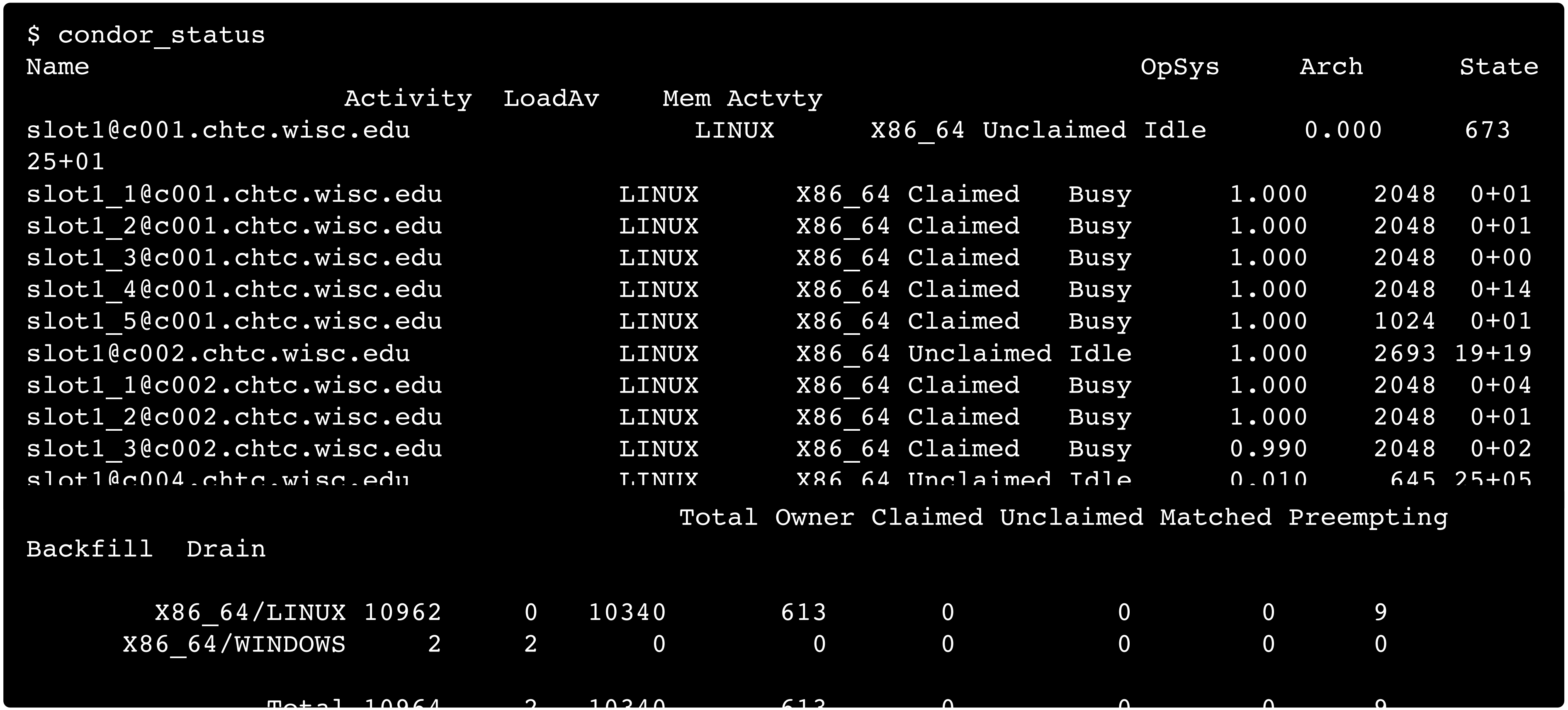
```
+JobBatchName = “CoolJobs”
```

  - Otherwise HTCondor groups jobs automatically
- To see individual jobs, use:

```
condor_q -nobatch
```

# Class Ads for Computers

as **condor_q** is to jobs, **condor_status** is to computers (or “machines”)



```
$ condor_status
Name                                                       OpSys      Arch       State
                    Activity  LoadAv    Mem Actvty
slot1@c001.chtc.wisc.edu                    LINUX      X86_64 Unclaimed Idle      0.000     673
25+01
slot1_1@c001.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+01
slot1_2@c001.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+01
slot1_3@c001.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+00
slot1_4@c001.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+14
slot1_5@c001.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    1024  0+01
slot1@c002.chtc.wisc.edu               LINUX      X86_64 Unclaimed Idle      1.000    2693 19+19
slot1_1@c002.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+04
slot1_2@c002.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      1.000    2048  0+01
slot1_3@c002.chtc.wisc.edu             LINUX      X86_64 Claimed   Busy      0.990    2048  0+02
slot1@c004.chtc.wisc.edu               LINUX      X86_64 Unclaimed Idle      0.010     645 25+05

                                          Total Owner Claimed Unclaimed Matched Preempting
Backfill  Drain

        X86_64/LINUX 10962     0    10340       613       0          0        0       9
      X86_64/WINDOWS     2     2        0         0       0          0        0       0

               Total 10964     2    10340       613       0          0        0       9
```

HTCondor Manual: condor_status

# Machine Attributes

- Use same options as **condor_q**:

**condor_status** -l *Slot/Machine*

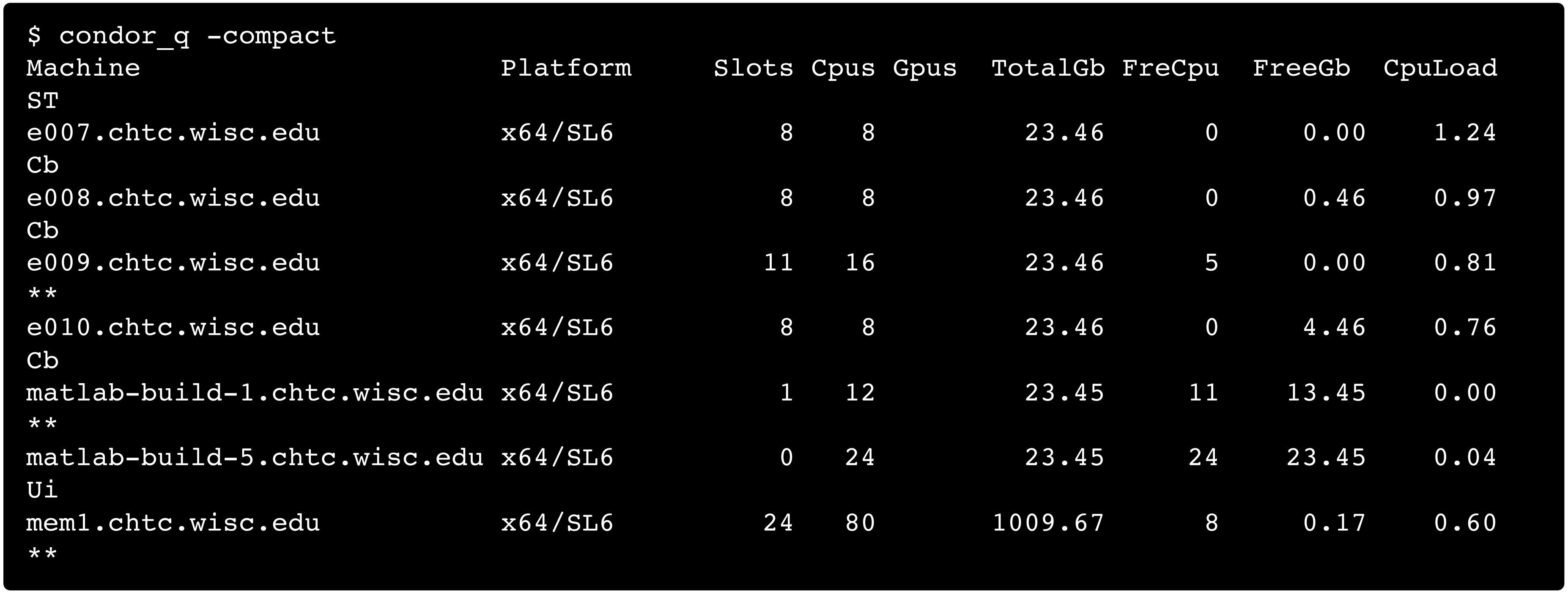
**condor_status** [Machine] -af *Attribute1 Attribute2* ...

```
$ condor_status -l slot1_1@c001.chtc.wisc.edu
HasFileTransfer = true
COLLECTOR_HOST_STRING = "cm.chtc.wisc.edu”
TargetType = "Job”
TotalTimeClaimedBusy = 43334c001.chtc.wisc.edu
UtsnameNodename = ""
Mips = 17902
MAX_PREEMPT = ( 3600 * ( 72 - 68 * ( WantGlidein =?= true ) ) )
Requirements = ( START ) && ( IsValidCheckpointPlatform ) && (
WithinResourceLimits )
State = "Claimed"
OpSysMajorVer = 6
OpSysName = "SL”
...
```

# Machine Attributes

- To summarize, use the “-compact” option

**condor_status -compact**



```
$ condor_q -compact
Machine                      Platform     Slots Cpus Gpus  TotalGb FreCpu  FreeGb  CpuLoad
ST
e007.chtc.wisc.edu           x64/SL6          8    8         23.46      0     0.00    1.24
Cb
e008.chtc.wisc.edu           x64/SL6          8    8         23.46      0     0.46    0.97
Cb
e009.chtc.wisc.edu           x64/SL6         11   16         23.46      5     0.00    0.81
**
e010.chtc.wisc.edu           x64/SL6          8    8         23.46      0     4.46    0.76
Cb
matlab-build-1.chtc.wisc.edu x64/SL6          1   12         23.45     11    13.45    0.00
**
matlab-build-5.chtc.wisc.edu x64/SL6          0   24         23.45     24    23.45    0.04
Ui
mem1.chtc.wisc.edu           x64/SL6         24   80       1009.67      8     0.17    0.60
**
```

# (60 SECOND) PAUSE

Questions so far?

# Submitting Multiple Jobs with HTCondor

# Many Jobs, One Submit File

- HTCondor has built-in ways to submit multiple independent jobs with one submit file

# Advantages

- Run many independent jobs...
  - analyze multiple data files
  - test parameter or input combinations
  - and more!
- ...without having to:
  - start each job individually
  - create separate submit files for each job

# Multiple, Numbered, Input Files

job.submit

```
executable = analyze.exe
arguments = file.in file.out
transfer_input_files = file.in

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

queue
```

(submit_dir)/

```
analyze.exe
file0.in
file1.in
file2.in

job.submit
```

- Goal: create 3 jobs that each analyze a different input file.

# Multiple Jobs, No Variation

job.submit

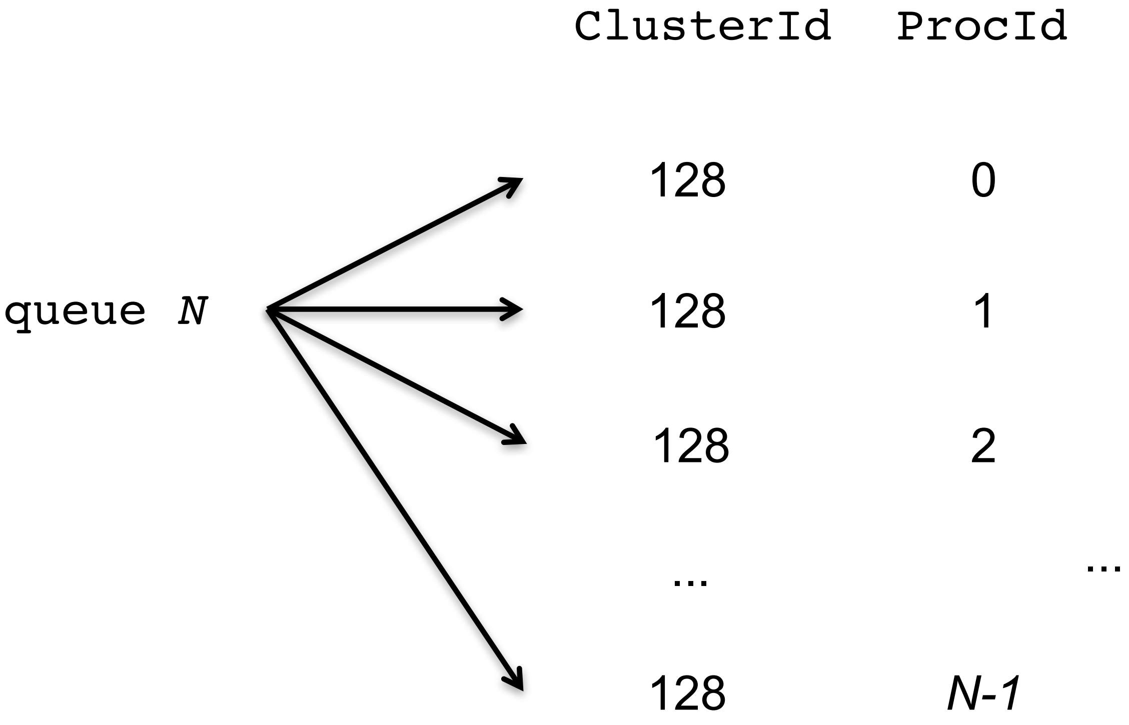
```
executable = analyze.exe
arguments = file0.in file0.out
transfer_input_files = file.in

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

queue 3
```

(submit_dir)/

```
analyze.exe
file0.in
file1.in
file2.in

job.submit
```

- This file generates 3 jobs, but doesn’t use multiple inputs and will overwrite outputs

# Automatic Variables



`queue N`

| `ClusterId` | `ProcId` |
|---|---|
| 128 | 0 |
| 128 | 1 |
| 128 | 2 |
| ... | ... |
| 128 | *N-1* |

- Each job's `ClusterId` and `ProcId` numbers are saved as job attributes
- They can be accessed inside the submit file using:
  - `$(ClusterId)`
  - `$(ProcId)`

# Job Variation

job.submit

```
executable = analyze.exe
arguments = file0.in file0.out
transfer_input_files = file0.in

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

queue
```

(submit_dir)/

```
analyze.exe
file0.in
file1.in
file2.in

job.submit
```

- How to uniquely identify each job (filenames, log/out/err names)?

# Using $(ProcId)

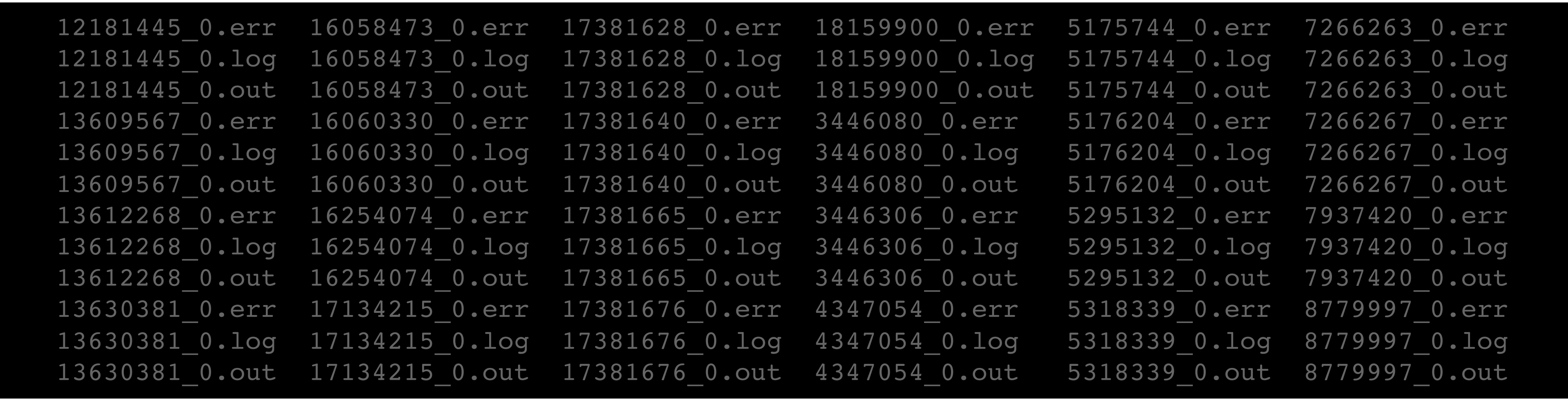
job.submit

```
executable = analyze.exe
arguments = file$(ProcId).in file$(ProcId).out
should_transfer_files = YES
transfer_input_files = file$(ProcId).in
when_to_transfer_output = ON_EXIT

log = job_$(ClusterId).log
output = job_$(ClusterId)_$(ProcId).out
error = job_$(ClusterId)_$(ProcId).err

queue 3
```

- Use the `$(ClusterId)`, `$(ProcId)` variables to provide unique values to jobs.*

* May also see `$(Cluster)`, `$(Process)` in documentation

# Organizing Jobs



```
12181445_0.err  16058473_0.err  17381628_0.err  18159900_0.err  5175744_0.err  7266263_0.err
12181445_0.log  16058473_0.log  17381628_0.log  18159900_0.log  5175744_0.log  7266263_0.log
12181445_0.out  16058473_0.out  17381628_0.out  18159900_0.out  5175744_0.out  7266263_0.out
13609567_0.err  16060330_0.err  17381640_0.err  3446080_0.err   5176204_0.err  7266267_0.err
13609567_0.log  16060330_0.log  17381640_0.log  3446080_0.log   5176204_0.log  7266267_0.log
13609567_0.out  16060330_0.out  17381640_0.out  3446080_0.out   5176204_0.out  7266267_0.out
13612268_0.err  16254074_0.err  17381665_0.err  3446306_0.err   5295132_0.err  7937420_0.err
13612268_0.log  16254074_0.log  17381665_0.log  3446306_0.log   5295132_0.log  7937420_0.log
13612268_0.out  16254074_0.out  17381665_0.out  3446306_0.out   5295132_0.out  7937420_0.out
13630381_0.err  17134215_0.err  17381676_0.err  4347054_0.err   5318339_0.err  8779997_0.err
13630381_0.log  17134215_0.log  17381676_0.log  4347054_0.log   5318339_0.log  8779997_0.log
13630381_0.out  17134215_0.out  17381676_0.out  4347054_0.out   5318339_0.out  8779997_0.out
```

# Shared Files

- HTCondor can transfer an entire directory or all the contents of a directory
  - transfer whole directory

```
transfer_input_files = shared
```

  - transfer contents only

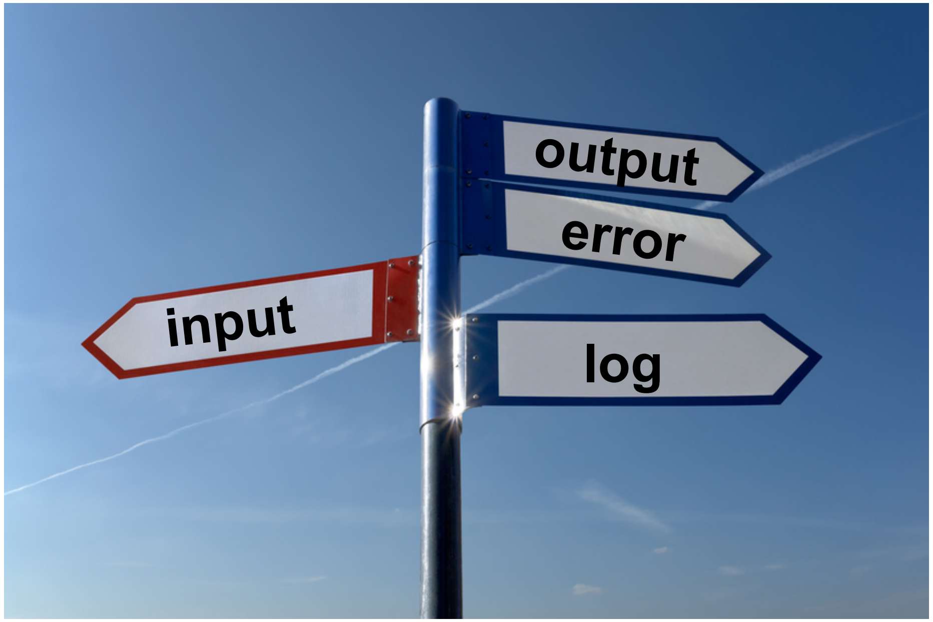
```
transfer_input_files = shared/
```

```
(submit_dir)/
job.submit
shared/
    reference.db
    parse.py
    analyze.py
    cleanup.py
    links.config
```

- Useful for jobs with many shared files; transfer a directory of files instead of listing files individually

# Organize Files in Sub-Directories

- Create sub-directories* and use paths in the submit file to separate input, error, log, and output files.



* must be created before the job is submitted

# Use Paths for File Type

(submit_dir)/

```
job.submit     file0.out    input/        log/         err/
analyze.exe    file1.out      file0.in      job0.log     job0.err
               file2.out      file1.in      job1.log     job1.err
                              file2.in      job2.log     job2.err
```

job.submit

```
executable = analyze.exe
arguments = file$(Process).in file$(ProcId).out
transfer_input_files = input/file$(ProcId).in

log = log/job$(ProcId).log
error = err/job$(ProcId).err

queue 3
```

# InitialDir

- Change the submission directory for each job using `initialdir`
- Allows the user to organize job files into separate directories.
- Use the same name for all input/output files
- Useful for jobs with lots of output files

# Separate Jobs with InitialDir

(submit_dir)/

```
job.submit      job0/         job1/         job2/
analyze.exe       file.in       file.in       file.in
                  job.log       job.log       job.log
                  job.err       job.err       job.err
                  file.out      file.out      file.out
```

job.submit

```
executable = analyze.exe
initialdir = job$(ProcId)
arguments = file.in file.out
transfer_input_files = file.in

log = job.log
error = job.err

queue 3
```

Executable should be in the directory with the submit file, *not* in the individual job directories

# Other Submission Methods

- What if your input files/directories aren't numbered from 0 - (N-1)?
- There are other ways to submit many jobs!

# Submitting Multiple Jobs

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

transfer_input_files = us.dat, wi.dat

queue 1
```

Replacing single job inputs

```
executable = compare_states
arguments = $(infile) us.dat $(infile).out

transfer_input_files = us.dat, $(infile)

queue ...
```

with a variable of choice

# Possible Queue Statements

| | |
|---|---|
| multiple “queue” statements | `infile = wi.dat`<br>`queue 1`<br>`infile = ca.dat`<br>`queue 1`<br>`infile = ia.dat`<br>`queue 1` |
| matching ... pattern | `queue infile matching *.dat` |
| in ... list | `queue infile in (wi.dat ca.dat ia.dat)` |
| from ... file | `queue infile from state_list.txt`<br>`wi.dat`<br>`ca.dat`<br>`ia.dat`<br>`state_list.txt` |

# Possible Queue Statements

| | |
|---|---|
| multiple “queue” statements | ~~`infile = wi.dat`~~<br>~~`queue 1`~~<br>~~`infile = ca.dat`~~<br>~~`queue 1`~~<br>~~`infile = ia.dat`~~<br>~~`queue 1`~~<br>Not Recommended |
| matching ... pattern | `queue infile matching *.dat` |
| in ... list | `queue infile in (wi.dat ca.dat ia.dat)` |
| from ... file | `queue infile from state_list.txt`<br>`wi.dat`<br>`ca.dat`<br>`ia.dat`<br>`state_list.txt` |

# Queue Statement Comparison

| | |
|---|---|
| multiple queue statements | Not recommended.  Can be useful when submitting job batches where a single (non-file/argument) characteristic is changing |
| matching .. pattern | Natural nested looping, minimal programming, use optional “files” and “dirs” keywords to only match files or directories<br>Requires good naming conventions, |
| in .. list | Supports multiple variables, all information contained in a single file, reproducible<br>Harder to automate submit file creation |
| from .. file | Supports multiple variables, highly modular (easy to use one submit file for many job batches), reproducible<br>Additional file needed |

# Using Multiple Variables

- Both the "`from`" and "`in`" syntax support using multiple variables from a list.

job.submit

```
executable = compare_states
arguments = -y $(option) -i $(file)

should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = $(file)

queue file,option from job_list.txt
```

job_list.txt

```
wi.dat, 2010
wi.dat, 2015
ca.dat, 2010
ca.dat, 2015
ia.dat, 2010
ia.dat, 2015
```

HTCondor Manual: submit file options

# Other Features

- Match only files or directories:

```
queue input matching files *.dat
```

```
queue directory matching dirs job*
```

- Submit multiple jobs with same input data

```
queue 10 input matching files *.dat
```

  – Use other automatic variables: `$(Step)`

```
arguments = -i $(input) -rep $(Step)
queue 10 input matching files *.dat
```

# Questions?