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

- 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...
Benefits: community
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 very 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_dir)/
input files
executable
```

```
(execute_dir)/
output files
```
Job Translation

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

```plaintext
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
```
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

```bash
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.

```bash
$ 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.
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/err or: 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
  ```

$$
\text{condor\_submit \textit{job\_submit}}
\text{condor\_q}
$$

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

```bash
$ 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>...

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.0</td>
<td>alice</td>
<td>5/9</td>
<td>11:09</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.0</td>
<td>alice</td>
<td>5/9   11:09</td>
<td>0+00:00:00</td>
<td>&lt;</td>
<td>0</td>
<td>0.0</td>
<td>compare_states wi.dat us.dat w</td>
</tr>
</tbody>
</table>

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
$ condor_q -nobatch

-- Schedd: submit-5.chtc.wisc.edu: <128.104.101.92:9618>...

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.0</td>
<td>alice</td>
<td>5/9 11:09</td>
<td>0+00:01:08</td>
<td>R</td>
<td>0</td>
<td>0.0 compare_states wi.dat us.dat</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>alice</td>
<td>5/9</td>
<td>11:09</td>
<td>0+00:02:02</td>
<td>&gt; 0</td>
<td>0.0</td>
<td>compare_states wi.dat us.dat</td>
</tr>
</tbody>
</table>

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
$ condor_q -nobatch

-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618>?...

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Submit Node

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

condor_submit

Idle (I) → Running (R) → Completed (C)

- Transfer executable and input to execute node
- Transfer output back to submit node

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

\[
\text{should\_transfer\_files} = \text{YES}
\]

  – not the case with a shared filesystem

\[
\text{should\_transfer\_files} = \text{NO}
\]
Job Matching and Class Ad Attributes
The Central Manager

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

submit

execute

central manager

execute

execute

execute
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

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
...

+ HTCondor configuration*

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

=
Computer “Machine” Class Ad

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.
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`

```bash
$ 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:

\[
\text{condor\_q \ [U/C/J] \ -af \ Attribute1 \ Attribute2 \ ...}
\]

```
$ 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)

```bash
$ 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”)

```sh
$ condor_status
Name          Activity     LoadAv Mem Activity OpSys Arch State
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+01
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+01
slot1_2@c002.chtc.wisc.edu       LINUX X86_64 Claimed Busy 1.000  2048  0+04
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

<table>
<thead>
<tr>
<th>Backfill</th>
<th>Drain</th>
</tr>
</thead>
<tbody>
<tr>
<td>X86_64/LINUX</td>
<td>10962</td>
</tr>
<tr>
<td>X86_64/WINDOWS</td>
<td>2</td>
</tr>
</tbody>
</table>

Total 10964 | 2 | 10340 | 613 | 0 | 0 | 0 | 9 |

HTCondor Manual: `condor_status`
Machine Attributes

- Use same options as `condor_q`:

```sh
$ condor_status -l Slot/Machine
condor_status [Machine] -af Attribute1 Attribute2 ...
```

```sh
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_q -compact
```

<table>
<thead>
<tr>
<th>Machine</th>
<th>Platform</th>
<th>Slots</th>
<th>Cpus</th>
<th>Gpus</th>
<th>TotalGb</th>
<th>FreCpu</th>
<th>FreeGb</th>
<th>CpuLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td>e007.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>8</td>
<td>8</td>
<td></td>
<td>23.46</td>
<td>0</td>
<td>0.00</td>
<td>1.24</td>
</tr>
<tr>
<td>Cb</td>
<td>x64/SL6</td>
<td>8</td>
<td>8</td>
<td></td>
<td>23.46</td>
<td>0</td>
<td>0.46</td>
<td>0.97</td>
</tr>
<tr>
<td>e008.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>11</td>
<td>16</td>
<td></td>
<td>23.46</td>
<td>5</td>
<td>0.00</td>
<td>0.81</td>
</tr>
<tr>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e009.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>8</td>
<td>8</td>
<td></td>
<td>23.46</td>
<td>0</td>
<td>4.46</td>
<td>0.76</td>
</tr>
<tr>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e010.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>1</td>
<td>12</td>
<td></td>
<td>23.45</td>
<td>11</td>
<td>13.45</td>
<td>0.00</td>
</tr>
<tr>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matlab-build-1.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>0</td>
<td>24</td>
<td></td>
<td>23.45</td>
<td>24</td>
<td>23.45</td>
<td>0.04</td>
</tr>
<tr>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matlab-build-5.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>24</td>
<td>80</td>
<td></td>
<td>1009.67</td>
<td>8</td>
<td>0.17</td>
<td>0.60</td>
</tr>
<tr>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- To summarize, use the “-compact” option

```
condor_status -compact
```

- To summarize, use the “-compact” option

```
```
(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

```plaintext
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
```
Multiple Jobs, No Variation

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

- 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

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

(submit_dir)/

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

job.submit
Using $(ProcId)$

```bash
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
Shared Files

• HTCondor can transfer an entire directory or all the contents of a directory
  – transfer whole directory
    \[\text{transfer_input_files} = \text{shared}\]
  – transfer contents only
    \[\text{transfer_input_files} = \text{shared/}\]

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

\[
\begin{align*}
\text{job.submit} \\
\text{shared/} \\
\text{reference.db} \\
\text{parse.py} \\
\text{analyze.py} \\
\text{cleanup.py} \\
\text{links.config}
\end{align*}
\]
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

```
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

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

Replacing single job inputs with a variable of choice

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

transfer_input_files = us.dat, wi.dat

queue 1
```

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

transfer_input_files = us.dat, $(infile)

queue ...
```
## Possible Queue Statements

| multiple “queue” statements | infile = wi.dat  
queue 1  
infile = ca.dat  
queue 1  
infile = ia.dat  
queue 1 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>matching ... pattern</td>
<td>queue infile matching *.dat</td>
</tr>
<tr>
<td>in ... list</td>
<td>queue infile in (wi.dat ca.dat ia.dat)</td>
</tr>
<tr>
<td>from ... file</td>
<td>queue infile from state_list.txt wi.dat ca.dat ia.dat state_list.txt</td>
</tr>
</tbody>
</table>
# Possible Queue Statements

| multiple "queue" statements | `infile = wi.dat`  
|                            | `queue 1`  
|                            | `infile = ca.dat`  
|                            | `queue 1`  
|                            | `infile = ia.dat`  
|                            | `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`  
|                            | `- wi.dat`  
|                            | `- ca.dat`  
|                            | `- ia.dat`  
|                            | `- state_list.txt` |

*Not Recommended*
## Queue Statement Comparison

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

- Both the “from” and “in” syntax support using multiple variables from a list.

```plaintext
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:

\[
\begin{align*}
\text{queue input matching} & \text{ files } *\.dat \\
\text{queue directory matching} & \text{ dirs job*}
\end{align*}
\]

• Submit multiple jobs with same input data

\[
\begin{align*}
\text{queue 10 input matching} & \text{ files } *\.dat \\
\text{arguments} & = -i $(\text{input}) -rep $(\text{Step}) \\
\text{queue 10 input matching} & \text{ files } *\.dat
\end{align*}
\]

– Use other automatic variables: \$(\text{Step})\]
Questions?