Job Matching, Handling, and Other HTCondor Features

Monday, Lecture 3

Lauren Michael
Questions so far?
Goals for this Session

- Understand HTCondor mechanisms more deeply
- Automation, additional use cases and features
How is HTC Optimized?

• System must track jobs, machines, policy, …
• System must recover gracefully from failures
• Try to use all available resources, all the time
• Lots of variety in users, machines, networks, …
• Sharing is hard (e.g. policy, security)
HTCONDOR MATCHMAKING
Roles in an HTCondor System

- **Users**
  - Define jobs, their requirements, and preferences
  - Submit and cancel jobs
  - Check on the status of jobs

- **Administrators**
  - Configure and control the HTCondor system
  - Implement policies
  - Check on the status of machines

- **HTCondor Software**
  - Track and manage machines
  - Track and run jobs
  - Match jobs to machines (enforcing all policies)
• On a regular basis, the central manager reviews job and machine attributes, and pool policies, and matches jobs to slots.
Single Computer
Terminology: Matchmaking

two-way process of finding a slot for a job

• **Jobs** have requirements and preferences
  – e.g.: I need one CPU core, 100 GB of disk space, and 10 GB of memory

• **Machines** have requirements and preferences
  – E.g.: I run jobs only from users in the Comp. Sci. dept., and prefer to run ones that ask for a lot of memory

• **Important jobs may run first or replace less important ones**
HTCondor Priorities

• **User priority**
  – Computed based on past usage
  – Determines user’s “fair share” percentage of slots
  – Lower number means run sooner (0.5 is minimum)

• **Job priority**
  – Set per job by the user (owner)
  – Relative to that user’s other jobs
  – Set in submit file or changed later with `condor_prio`
  – Higher number means run sooner

• **Preemption**
  – Low priority jobs stopped for high priority ones (stopped jobs go back into the regular queue)
  – Governed by fair-share algorithm and pool policy
  – Not enabled on all pools
Class Ads

- HTCondor stores a list of information about each job and each machine of potential slots.
- This information is stored for each job and each machine as its “Class Ad”

- Class Ads have the format:
  
  AttributeName = value

  can be a boolean (T/F), number, or string

HTCondor Manual: Appendix A: Class Ad Attributes
Submit file

```
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
```

Default HTCondor configuration

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

Default HTCondor configuration

HasFileTransfer = true
DynamicSlot = true
TotalSlotDisk = 4300218.0
TargetType = "Job"
TotalSlotMemory = 2048
Mips = 17902
Memory = 2048
UtsnameSysname = "Linux"
MAX_PREEMPT = ( 3600 * ( 72 - 68 * 
( WantGlidein =?= true ) ) )
Requirements = ( START ) && ( 
IsValidCheckpointPlatform ) && ( 
WithinResourceLimits )
OpSysMajorVer = 6
TotalMemory = 9889
HasGluster = true
OpSysName = "SL"
HasDocker = true
...

...
On a regular basis, the central manager reviews Job and Machine ClassAds and matches jobs to slots.
Job Execution

- (Then the submit and execute points communicate directly.)
USING CLASSADS
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 12008.0
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
- **JobBatchName**: user-labeled job batches
- ...and more
Displaying Job Attributes

• View only specific attributes (\texttt{-af} for ‘autoformat’)

\texttt{condor_q [U/C/J] -af Attribute1 Attribute2 ...}

$\texttt{condor_q -af ClusterId ProcId RemoteHost MemoryUsage}$

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
condor_q Reminder

• Default output is batched jobs
  – Batches can be grouped by the user with the JobBatchName attribute in a submit file:
    
    ```
    JobBatchName = CoolJobs
    ```
  – Otherwise HTCondor groups jobs, automatically, by same executable

• To see individual jobs, use:
  ```condor_q -nobatch```
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_1@c002.chtc.wisc.edu LINUX    X86_64  Unclaimed Idle 1.000  2693 19+19
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 1.000  2048 0+01
slot1_3@c002.chtc.wisc.edu LINUX    X86_64  Claimed Busy 0.990  2048 0+02

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX 10962 0 10340 613 0 0 0 0 9
X86_64/WINDOWS 2 2 0 0 0 0 0 0 0
Total 10964 2 10340 613 0 0 0 0 9

HTCondor Manual: condor_status
```
Machine Attributes

• Use same ClassAd options as `condor_q`:

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

```bash
$ 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:
  \texttt{condor\_status -compact}

<table>
<thead>
<tr>
<th>Machine</th>
<th>Platform</th>
<th>Slots</th>
<th>Cpus</th>
<th>Gpus</th>
<th>TotalGb</th>
<th>PreCpu</th>
<th>FreeGb</th>
<th>CpuLoad</th>
<th>ST</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</td>
<td>0.00</td>
<td>1.24</td>
</tr>
<tr>
<td>e008.chtc.wisc.edu</td>
<td>x64/SL6</td>
<td>8</td>
<td>8</td>
<td></td>
<td>23.46</td>
<td>0</td>
<td>0.46</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>e009.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>
<td></td>
</tr>
<tr>
<td>e010.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>
<td></td>
</tr>
<tr>
<td>matlab-build-1.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>
<td>0.00</td>
</tr>
<tr>
<td>matlab-build-5.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>
<td>0.04</td>
</tr>
<tr>
<td>meml.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>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Platform</th>
<th>Owner</th>
<th>Claimed</th>
<th>Unclaimed</th>
<th>Matched</th>
<th>Preempting</th>
<th>Backfill</th>
<th>Drain</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x64/SL6</td>
<td>10416</td>
<td>0</td>
<td>9984</td>
<td>427</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>x64/WinVista</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>10418</td>
<td>2</td>
<td>9984</td>
<td>427</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
AUTOMATION AND OTHER FEATURES
Retries

- Problem: a small number of jobs fail with a known error code; if they run again, they complete successfully.

- Solution: If the job exits with an error code, leave it in the queue to run again. This is done via the automatic option `max_retries`.

`max_retries` = 5
More automation

• Check out the Intro to HTCondor talk from HTCondor Week 2019 for more on:
  – self-checkpointing
  – automatic hold/release (e.g. if job running too long)
  – auto-increasing memory request (e.g. if memory usage varies a lot across jobs)
“Live” Troubleshooting

• To log in to a job where it is running, use:

  condor_ssh_to_job JobId

$ condor_ssh_to_job 128.0
Welcome to slot1_31@e395.chtc.wisc.edu!
Your condor job is running with pid(s) 3954839.
Interactive Jobs

- An interactive job proceeds like a normal batch job, but opens a bash session into the job’s execution directory instead of running an executable.

```
condor_submit -i submit_file
```

```
$ condor_submit -i interactive.submit
Submitting job(s).
1 job(s) submitted to cluster 18980881.
Waiting for job to start...
Welcome to slot1_9@e184.chtc.wisc.edu!
```

- Useful for testing and troubleshooting
Job Universes

• HTCondor has different “universes” for running specialized job types
  
  [HTCondor Manual: Choosing an HTCondor Universe]

• Vanilla (default)
  – good for most software
  
  [HTCondor Manual: Vanilla Universe]

• Set in the submit file using: `universe = vanilla`
Other Universes

- **Standard**
  - Built for code (C, fortran) that can be statically compiled with `condor_compile`
  
  [HTCondor Manual: Standard Universe](#)

- **Java**
  - Built-in Java support

  [HTCondor Manual: Java Applications](#)

- **Local**
  - Run jobs on the submit node

  [HTCondor Manual: Local Universe](#)
Other Universes (cont.)

- **Docker**
  - Run jobs inside a Docker container
    
    HTCondor Manual: Docker Universe Applications

- **VM**
  - Run jobs inside a virtual machine
    
    HTCondor Manual: Virtual Machine Applications

- **Scheduler**
  - Runs DAG workflows (Thursday)
    
    HTCondor Manual: Parallel Applications
Multi-CPU and GPU Computing

- Jobs that use multiple cores on a single computer can use the vanilla universe (parallel universe for multi-server MPI, where supported):

  ```
  request_cpus = 16
  ```

- If there are computers with GPUs, request them with:

  ```
  request_gpus = 1
  ```
Want More HTCondor Features?

- See the “Introduction to Using HTCondor” talk from HTCondor Week 2019!!
  
YOUR TURN!
Exercises!

- Ask questions!
- Lots of instructors around

• Coming up:
  - Now-3:00 Hands-on Exercises
  - 3:00 – 3:15 Break
  - 3:15 – 5:00 Intro to DHTC, OSG