Day 10: More Condor

Suggested reading: Condor 7.7 Manual:
http://www.cs.wisc.edu/condor/manual/v7.7/

Chapter 2: Users’ Manual (at most, 2.1–2.7)
Chapter 9:
  condor_q, condor_status, condor_submit, condor_prio
Turn In Homework
Homework Review
More Condor Background
How Does Condor Work?
How Does Condor Work?

Function

- Track waiting/running jobs
- Track available machines
- Match jobs and machines
- Manage one machine
- Manage one job (on submitter)
- Manage one job (on machine)
## How Does Condor Work?

<table>
<thead>
<tr>
<th>Function</th>
<th>Condor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track waiting/running jobs</td>
<td>schedd (“sked-dee”)</td>
</tr>
<tr>
<td>Track available machines</td>
<td>collector</td>
</tr>
<tr>
<td>Match jobs and machines</td>
<td>negotiator</td>
</tr>
<tr>
<td>Manage one machine</td>
<td>startd (“start-dee”)</td>
</tr>
<tr>
<td>Manage one job (on submitter)</td>
<td>shadow</td>
</tr>
<tr>
<td>Manage one job (on machine)</td>
<td>starter</td>
</tr>
</tbody>
</table>
## How Does Condor Work?

<table>
<thead>
<tr>
<th>Function</th>
<th>Condor Name</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track waiting/running jobs</td>
<td>schedd (&quot;sked-dee&quot;)</td>
<td>1+</td>
</tr>
<tr>
<td>Track available machines</td>
<td>collector</td>
<td>1</td>
</tr>
<tr>
<td>Match jobs and machines</td>
<td>negotiator</td>
<td>1</td>
</tr>
<tr>
<td>Manage one machine</td>
<td>startd (&quot;start-dee&quot;)</td>
<td>per machine</td>
</tr>
<tr>
<td>Manage one job (on submitter)</td>
<td>shadow</td>
<td>per job running</td>
</tr>
<tr>
<td>Manage one job (on machine)</td>
<td>starter</td>
<td>per job running</td>
</tr>
</tbody>
</table>
The Life of a Job

Central Manager

- negotiator
- collector

Submit Machine

- schedd

Execute Machine

- startd
The Life of a Job

Central Manager

- negotiator
- collector

Submit Machine

- schedd

Execute Machine

- startd

send periodic updates
The Life of a Job

Central Manager

- negotiator
- collector

Submit Machine

- schedd

1. submit job

Execute Machine

- startd

send periodic updates
The Life of a Job

Central Manager

1. submit job

negotiator

2. request job details

collector

send periodic updates

schedd

startd

Submit Machine

Execute Machine
The Life of a Job

Central Manager

1. submit job
2. request job details
3. send jobs

Submit Machine

 negotiator

 collector

 Execute Machine

 startd

 schedd

send periodic updates

1. submit job
2. request job details
3. send jobs
The Life of a Job

Central Manager

- negotiator
- collector

Submit Machine

- schedd

Execute Machine

- startd

1. submit job
2. request job details
3. send jobs
4. notify of match

send periodic updates
The Life of a Job

Central Manager

1. submit job

2. request job details

3. send jobs

4. notify of match

send periodic updates

5. claim

Submit Machine

schedd

Execute Machine

startd

negotiator

collector
The Life of a Job

Central Manager

negotiator  collector

2. request job details  3. send jobs  4. notify of match

send periodic updates

scheddd  startd

5. claim  6. start

Submit Machine

1. submit job

Execute Machine

shadow  starter
The Life of a Job

**Central Manager**

- negotiator
- collector

**Submit Machine**

- schedd
- shadow

1. submit job
2. request job details
3. send jobs
4. notify of match
5. claim
6. start
7. transfer exec, input

**Execute Machine**

- startd
- starter
The Life of a Job

Central Manager

1. submit job
2. request job details
3. send jobs
4. notify of match
5. claim

Submit Machine

1. submit job
2. request job details
3. send jobs
4. notify of match
5. claim

Execute Machine

1. submit job
2. request job details
3. send jobs
4. notify of match
5. claim

Cartwright
2012 Spring
**The Life of a Job**

1. **Submit Machine**
   - **schedd**
     - 1. submit job
     - 6. start
     - 7. transfer exec, input
   - **shadow**
     - 6. start
     - 7. transfer exec, input
   - **starter**
     - 6. start
     - 8. start
   - **job**

2. **Central Manager**
   - **negotiator**
     - 2. request job details
     - 3. send jobs
     - 4. notify of match
   - **collector**
     - send periodic updates

3. **Execute Machine**
   - 5. claim
     - 6. start
     - 9. transfer output
Matchmaking Revisited

- Balances
  - Job (submitter)
  - Machine (owner)
  - Pool (administrator)

- Takes into account
  - Requirements
  - Preferences
  - Policy

- But how are they represented?
ClassAds

- For job, machine, etc.
- Loosely structured
- Few required parts
- Users can extend
- Can express:
  - Facts
  - Current state
  - Requirements
  - Preferences
  - Your shoe size
- attribute = expression

MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../homework_09.py"
Requirements =
  (Arch == "X86_64") &&
  (OpSys == "LINUX") &&
  ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../hw09.log"
Out = "hw09.out"
Err = "hw09.err"
NiceUser = false
ClassAds

• For job, machine, etc.
• Loosely structured
• Few required parts
• Users can extend
• Can express:
  – Facts
  – Current state
  – Requirements
  – Preferences
  – Your shoe size
• attribute = expression

MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../homework_09.py"
Requirements =
  (Arch == "X86_64") &&
  (OpSys == "LINUX") &&
  ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../hw09.log"
Out = "hw09.out"
Err = "hw09.err"
NiceUser = false
ClassAds

- For job, machine, etc.
- Loosely structured
- Few required parts
- Users can extend
- Can express:
  - Facts
  - Current state
  - Requirements
  - Preferences
  - Your shoe size
- attribute = expression

```
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../homework_09.py"
Requirements =
  (Arch == "X86_64") &&
  (OpSys == "LINUX") &&
  ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../hw09.log"
Out = "hw09.out"
Err = "hw09.err"
NiceUser = false
```
ClassAds

- For job, machine, etc.
- Loosely structured
- Few required parts
- Users can extend
- Can express:
  - Facts
  - Current state
  - Requirements
  - Preferences
  - Your shoe size
- attribute = expression

MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../homework_09.py"
Requirements =
  (Arch == "X86_64") &&
  (OpSys == "LINUX") &&
  ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../hw09.log"
Out = "hw09.out"
Err = "hw09.err"
NiceUser = false

boolean
ClassAds

- For job, machine, etc.
- Loosely structured
- Few required parts
- Users can extend
- Can express:
  - Facts
  - Current state
  - Requirements
  - Preferences
  - Your shoe size
- attribute = expression

```python
MyType = "Job"
TargetType = "Machine"
ClusterId = 14
Owner = "cat"
Cmd = "/.../homework_09.py"
Requirements =
  (Arch == "X86_64") &&
  (OpSys == "LINUX") &&
  ...
Rank = 0.0
In = "/dev/null"
UserLog = "/.../hw09.log"
Out = "hw09.out"
Err = "hw09.err"
NiceUser = false
```

operations/expressions
Priorities

- **Job priority**
  - Set by user (owner)
  - Is relative to *that user’s* other jobs
  - Higher number means run sooner

- **User priority**
  - Condor calculates this priority value based on past usage
  - Determines user’s potential share of machines
  - Lower number means run sooner (0.5 is minimum)
  - Results in “fair share” access to resources

- **Preemption**
  - Low priority jobs can be removed for high priority ones
  - Governed by fair-share algorithm and pool policy
What Makes a Good CHTC Job?

• Single-threaded, independent batch job

• Runs for about 10 minutes to 4 hours
  – Too short: Overhead costs predominate
  – Too long: Risk getting preempted (“bad-put”)  
  – CHTC removes any job after 24 hours of runtime

• Fits lots of machines — the more, the better!
  – Few requirements: low memory, low disk
  – Scripts! (few/no OS and architecture requirements)
Condor Commands
condor_q: Being More Selective

**condor_q username [...]**

- Lists jobs *only* owned by the user(s) (e.g., yourself)

**condor_q cluster [...]**

- Lists all jobs in the given cluster(s)

**condor_q cluster.process [...]**

- Lists only the given job(s)

```plaintext
-- Submitter: submit-368.chtc.wisc.edu : <...> : ...
ID  OWNER  SUBMITTED   RUN_TIME  ST  PRI  SIZE  CMD
 23.2  cat  11/13 15:21  0+00:00:00 I  0  0.0  explore.py
```
condor_q: ClassAd Output

condor_q -long cluster.process

• Displays complete ClassAd for each job (80+ lines)
• Great way to explore ClassAds for jobs
• Best to limit to a single job (cluster/process combo)!

-- Submitter: submit-368.chtc.wisc.edu : <...> : ...
PeriodicRemove = false
CommittedSlotTime = 0
Out = "explore.out.24.1"
ImageSize_RAW = 1
NumCkpts_RAW = 0
EnteredCurrentStatus = 1321219554
CommittedSuspensionTime = 0
WhenToTransferOutput = "ON_EXIT"
NumSystemHolds = 0
StreamOut = false
...
condor_q: Why Isn’t My Job Running?

• Tries to figure out if your job can run
• Often helpful – occasionally not – good starting pt.

condor_q -analyze cluster.process

026.000: Run analysis summary. Of 2072 machines,
2072 are rejected by your job's requirements
0 reject your job because of their own requirements

... No successful match recorded.
Last failed match: Sun Nov 13 15:33:29 2011
Reason for last match failure: no match found

WARNING: Be advised:
No resources matched request's constraints

The Requirements expression for your job is:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Machines Matched</th>
<th>Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( target.Memory &gt;= 9999999 )</td>
<td>0</td>
<td>MODIFY TO 212001</td>
</tr>
<tr>
<td>2 ( TARGET.Arch == &quot;X86_64&quot; )</td>
<td>2020</td>
<td></td>
</tr>
<tr>
<td>3 ( TARGET.OpSys == &quot;LINUX&quot; )</td>
<td>2020</td>
<td></td>
</tr>
</tbody>
</table>
condor_status: Classes of Machines

condor_status -avail

- Lists slots that are available

condor_status -constraint ClassAdExpr

- Lists slots that match constraint(s)

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
<th>ActvtyTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:slot10@c011.chtc.w">slot10@c011.chtc.w</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>6.690</td>
<td>12017</td>
<td>0+14:41:56</td>
</tr>
<tr>
<td><a href="mailto:slot10@c013.chtc.w">slot10@c013.chtc.w</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>7.980</td>
<td>12017</td>
<td>0+14:50:57</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:slot25@opt-a012.ch">slot25@opt-a012.ch</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.000</td>
<td>99111</td>
<td>0+21:01:43</td>
</tr>
</tbody>
</table>

Total Owner Claimed Unclaimed Matched Preempting Backfill

<table>
<thead>
<tr>
<th>X86_64/LINUX</th>
<th>66</th>
<th>2</th>
<th>55</th>
<th>9</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>66</td>
<td>2</td>
<td>55</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
condor_status: Being More Selective

condor_status *hostname* [....]

• Lists slots with the given hostname(s)

condor_status *slot@hostname* [....]

• Lists the given slot(s)

```
% condor_status c040.chtc.wisc.edu

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
<th>ActvtyTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:slot10@c040.chtc.wi">slot10@c040.chtc.wi</a> LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>7.990</td>
<td>12017</td>
<td>0+19:36:09</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:slot1@c040.chtc.wi">slot1@c040.chtc.wi</a> LINUX</td>
<td>X86_64</td>
<td>Owner</td>
<td>Idle</td>
<td>0.000</td>
<td>4599</td>
<td>0+19:36:03</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:slot9@c040.chtc.wi">slot9@c040.chtc.wi</a> LINUX</td>
<td>X86_64</td>
<td>Owner</td>
<td>Idle</td>
<td>0.020</td>
<td>250</td>
<td>47+05:24:44</td>
<td></td>
</tr>
</tbody>
</table>

Total Owner Claimed Unclaimed Matched Preempting Backfill

X86_64/LINUX  10  9  1  0  0  0
Total          10  9  1  0  0  0
```
condor_status: ClassAd Output

condor_status -long slot@hostname

- Displays complete ClassAd for each slot (120+ lines)
- Great way to understand ClassAds for machines
- Best to limit to a single slot!

Machine = "opt-a001.chtc.wisc.edu"
DCSignalRuntime = 247.566893
EnteredCurrentState = 1321222293
JavaVersion = "1.6.0_20"
DetectedMemory = 258331
OpSysAndVer = "LINUX"
HasMPI = true
CpuIsBusy = false
LastBenchmark = 1321228954
HasVM = false
JavaVendor = "Sun Microsystems Inc."
...
condor_prio

Condor_prio -p value cluster[.process] […]

- Sets the job priority to the given value
- Identify job(s) with 1+ user(s), cluster(s), process(es)

Condor_prio +value cluster[.process] […]
Condor_prio -value cluster[.process] […]

- Raise or lower the job priority by the given amount
Submit Files
Setting Priority (Again)

priority = \textit{integer}

- Sets job priority right in submit file
- Default is 0
- Only affects relative priority of your jobs
- Can override using \texttt{condor\_prio}
Notifications by Email

notification = Always | Complete | Error | Never

- When to send email
  - **Always**: job checkpoints or completes
  - **Complete**: job completes *(default)*
  - **Error**: job completes with error
  - **Never**: do not send email

notify_user = email

- Where to send email
- Defaults to *job-owner@submit-machine*
Input Files From the Internet

\[\text{transfer\_input\_files} = \text{URL}[, \ldots]\]

- Grab input files from any available URL
- **BUT:** If the download fails, your job goes on hold
  - You don’t know when your job will run
  - Maybe that will be during server maintenance, etc.
- So, great idea, but maybe wait for retries…
  - Can always pre-fetch file yourself
  - Or, job itself can download files, and do it robustly
Arbitrary Attributes

+AttributeName = value

- Adds arbitrary attribute(s) to job ClassAd
- Useful in (at least) two cases:
  - Find jobs using attribute: `condor_q -constraint`
  - Attribute has special policy meaning in pool
- As it happens, we have a special policy...

+WantRHEL6Job = true
rank = (IsRHEL6 == True)
Requirements

expression = \textit{ClassAdExpression}

- Expression must evaluate to true to run on machine
- Condor adds defaults! View with \texttt{condor\_q -long}
- See Condor Manual (esp. 2.5.2 & 4.1) for details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpSys</td>
<td>operating system</td>
</tr>
<tr>
<td>Arch</td>
<td>architecture</td>
</tr>
<tr>
<td>Memory</td>
<td>memory, in MB</td>
</tr>
<tr>
<td>HasJava</td>
<td>True/False</td>
</tr>
<tr>
<td>IsRHEL6</td>
<td>True/False</td>
</tr>
<tr>
<td>ShoeSize</td>
<td>(if defined in pool)</td>
</tr>
</tbody>
</table>

requirements =
  (OpSys == "LINUX") &&
  ((Arch == "X86\_64") ||
   (Arch == "INTEL")) &&
  (Memory >= 64) &&
  (IsRHEL6 == true)
Preferences (Rank)

\[
\text{rank} = \text{ClassAdExpression}
\]

- Ranks *matching* machines in order by preference
- Must evaluate to a FP number, greater is preferred
  - False becomes 0.0, True becomes 1.0
  - Undefined or error values become 0.0
- Writing rank expressions is an art form

\[
\text{rank} = \text{Memory}
\]

\[
\text{rank} = (\text{IsRHEL6} \ == \ True)
\]

\[
\text{rank} = ((\text{substr(Machine, 0, 1)} \ == \ 'c') \ * \ 2) \ + \\
(\text{substr(Machine, 0, 1)} \ == \ 'e'))
\]
One Submit, Many Jobs: I

- Can use `queue` statement many times
- Make changes between `queue` statements
  - Change `arguments`, `output`, `priority`, ...
  - Whatever you do not explicitly change stays the same

```python
executable = test.py
...  
log    = test.log

output  = test-1.out
arguments = "test-input.txt 42"
queue

output  = test-2.out
arguments = "test-input.txt 43"
queue
```
One Submit, Many Jobs: II

- Submits \( N \) copies of the job
  - One cluster number for all copies, just as before
  - Process numbers go from 0 – (\( N \)–1)

- What good is having \( N \) copies of the same thing?
  - Randomized processes (cf. homework #8)
  - Job fetches work description from somewhere?
  - But what about overwriting output files, etc.?

- Wouldn’t it be nice to have different files and/or arguments automatically applied to each job?
Separating Files by Run

output = program.out.$(Cluster).$(Process)

- Can use either/both of these variables anywhere
  - Often used in output, error, and log files
- Maybe use $(Process) in arguments?
  - No math on values; your program must handle as is

... 
output = test.$(Cluster)_$(Process).out
log = test.$(Cluster)_$(Process).log
arguments = "test-input.txt $(Process)"
queue 10
Separating Directories by Run

initialdir = *path*

• Use *path* (instead of submit dir.) to locate files
  – I.e., `output`, `error`, `log`, `transfer_input_files`
  – *Not executable*; always relative to submit directory
• Mix with `$\text{(Process)}$` and separate all I/O by job

```
initialdir = run-$(Process)
transfer_input_files = input-$(Process).txt
output = test.$(Cluster)-$(Process).out
log = test.$(Cluster)-$(Process).log
arguments = "input-$(Process).txt $(Process)"
queue 10
```
Homework
Homework

• Write a little bit of Python code, lest you forget!
• Run lots of jobs from a single submit file
• Play with condor_q, condor_status, & condor_prio