Administrating HTCondor

Alan De Smet
Center for High Throughput Computing
adesmet@cs.wisc.edu
http://research.cs.wisc.edu/htcondor
The next 70 minutes…

› HTCondor Daemons & Job Startup
› Configuration Files
› Security, briefly
› Policy Expressions
  • Startd (Machine)
  • Negotiator
› Priorities
› Useful Tools
› Log Files
› Debugging Jobs
Daemons & Job Startup
Configuration Files
Configuration File

- CONDOR_CONFIG environment variable, 
  /etc/condor/condor_config, 
  ~condor/condor_config

- All settings can be in this one file

- Might want to share between all machines (NFS, automated copies, Wallaby, etc)
Other Configuration Files

LOCAL_CONFIG_FILE

- Comma separated, processed in order

```
LOCAL_CONFIG_FILE = \n/var/condor/config.local,\n/var/condor/policy.local,\n/shared/condor/config.$(HOSTNAME),\n/shared/condor/config.$(OPSYS)
```

LOCAL_CONFIG_DIR

```
LOCAL_CONFIG_DIR = \n/var/condor/config.d/,,\n/var/condor/$(OPSYS).d/
```
# I’m a comment!

CREATE_CORE_FILES=TRUE

MAX_JOBS_RUNNING = 50

# HTCondor ignores case:

log=/var/log/condor

# Long entries:

collector_host=condor.cs.wisc.edu, secondary.cs.cs.wisc.edu

Configuration File Macros

› You reference other macros (settings) with:
  • \texttt{A} = $(B)$
  • \texttt{SCHEDD} = $(SBIN)/condor\_schedd$

› Can create additional macros for organizational purposes
Configuration File Macros

› Can append to macros:
   \[ A=abc \]
   \[ A=$(A),def \]

› Don’t let macros recursively define each other!
   \[ A=$(B) \]
   \[ B=$(A) \]
Later macros in a file overwrite earlier ones

- B will evaluate to 2:
  \[
  \begin{align*}
  A &= 1 \\
  B &= \$(A) \\
  A &= 2
  \end{align*}
  \]
These are simple replacement macros

Put parentheses around expressions

\[ \text{TEN} = 5+5 \]

\[ \text{HUNDRED} = $(\text{TEN}) \ast $(\text{TEN}) \]
• HUNDRED becomes 5+5*5+5 or 35!

\[ \text{TEN} = (5+5) \]

\[ \text{HUNDRED} = ({$(\text{TEN}) \ast $(\text{TEN})}) \]
• \(((5+5)*(5+5)) = 100\]
Security, briefly
HTCondor Security

- Strong authentication of users and daemons
- Encryption over the network
- Integrity checking over the network
Minimal Security Settings

› You must set `ALLOW_WRITE`, or nothing works

› Simplest setting:

```
ALLOW_WRITE=*  
• Extremely insecure!
```

› A bit better:

```
ALLOW_WRITE=\  
*.cs.wisc.edu
```

"Bank Security Guard" by "Brad & Sabrina" © 2006
Licensed under the Creative Commons Attribution 2.0 license
More on Security

† Chapter 3.6, “Security,” in the HTCondor Manual
† htcondor-admin@cs.wisc.edu
Policy

“Don't even think about it” by Kat “tyger_lyllie” © 2005
Licensed under the Creative Commons Attribution 2.0 license
http://www.flickr.com/photos/tyger_lyllie/59207292/
http://www.webcitation.org/5XIh5mYGS
Policy

Who gets to run jobs, when?
Policy Expressions

› Specified in **condor_config**
  • Ends up slot ClassAd

› Policy evaluates both a slot ClassAd and a job ClassAd together
  • Policy can reference items in either ClassAd (See manual for list)

› Can reference **condor_config** macros: $(MACRONAME)
Slots vs Machines

› Machine – An individual computer, managed by one startd

› Slot – A place to run a job, managed by one starter.
  • A machine may have many slots
  • Partionable slots create more slots on the fly

› The start advertises each slot
  • The ClassAd is a “Machine” ad for historical reasons
Slot Policy Expressions

- START
- RANK
- SUSPEND
- CONTINUE
- PREEMPT
- KILL
START

- START is the primary policy
- When FALSE the slot enters the Owner state and will not run jobs
- Acts as the Requirements expression for the slot, the job must satisfy START
  - Can reference job ClassAd values including Owner and ImageSize
RANK

› Indicates which jobs a slot prefers
  • Jobs can also specify a rank

› Floating point number
  • Larger numbers are higher ranked
  • Typically evaluate attributes in the Job ClassAd
  • Typically use + instead of &&
Often used to give priority to owner of a particular group of machines

Claimed slots still advertise looking for higher ranked job to preempt the current job

- *RANK causes preemption!*
SUSPEND and CONTINUE

› When SUSPEND becomes true, the job is suspended

› When CONTINUE becomes true a suspended job is released
When PREEMPT becomes true, the job will be politely shut down
- Vanilla universe jobs get SIGTERM
  - Or user requested signal
- Standard universe jobs checkpoint

When KILL becomes true, the job is SIGKILLed
- Checkpointing is aborted if started
Minimal / Default Settings

- Always runs jobs
  - START = True
  - RANK = 0
  - SUSPEND = False
  - CONTINUE = True
  - PREEMPT = False
  - KILL = False

“Lonely at the top” by Guyon Moree (“gumuz”) © 2005
Licensed under the Creative Commons Attribution 2.0 license
http://www.flickr.com/photos/gumuz/7340411/ http://www.webcitation.org/5XIh8s0kI
I am adding nodes to the Cluster… but the Chemistry Department has priority on these nodes.
New Settings for the Chemistry nodes

› Prefer Chemistry jobs

   START = True
   RANK = Department == "Chemistry"
   SUSPEND = False
   CONTINUE = True
   PREEMPT = False
   KILL = False
Submit file with Custom Attribute

- Prefix an entry with “+” to add to job ClassAd
  
  Executable = charm-run
  Universe = standard
  +Department = "Chemistry"
  queue
What if “Department” not specified?

START = True
RANK = Department == "Chemistry"
SUSPEND = False
CONTINUE = True
PREEMPT = False
KILL = False
Give the machine’s owners (adesmet and roy) highest priority, followed by the Chemistry department, followed by the Physics department, followed by everyone else.

- Can use automatic **Owner** attribute in job attribute to identify adesmet and roy
More Complex RANK

\[
\text{IsOwner} = (\text{Owner} == "adesmet" \lor \text{Owner} == "roy")
\]

\[
\text{IsChem} = (\text{Department} == "Chemistry")
\]

\[
\text{IsPhys} = (\text{Department} == "Physics")
\]

\[
\text{RANK} = \$(\text{IsOwner}) \times 20 + \$(\text{IsChem}) \times 10 \lor + \$(\text{IsPhys})
\]
I have an unhealthy fixation with PBS so… *kill jobs after 12 hours, except Physics jobs get 24 hours.*
Useful Attributes

› **CurrentTime**
  • Current time, in Unix epoch time (seconds since midnight Jan 1, 1970)

› **EnteredCurrentActivity**
  • When did HTCondor enter the current activity, in Unix epoch time
Configuration

ActivityTimer = \( (\text{CurrentTime} - \text{EnteredCurrentActivity}) \)

\text{HOUR} = (60*60)

\text{HALFDAY} = (\text{HOUR})*12)

\text{FULLDAY} = (\text{HOUR})*24)

\text{PREEMPT} = \$

\text{PREEMPT} = (\text{IsPhys}) \&\& (\text{ActivityTimer} > \text{FULLDAY})) \$

|| \$

(\text{IsPhys}) \&\& (\text{ActivityTimer} > \text{HALFDAY})) \$

\text{KILL} = \$\text{(PREEMPT)}$
The cluster is okay, but...

HTCondor can only use the desktops when they would otherwise be idle.
Defining Idle

› One possible definition:
  • No keyboard or mouse activity for 5 minutes
  • Load average below 0.3
Desktops should

- **START** jobs when the machine becomes idle
- **SUSPEND** jobs as soon as activity is detected
- **PREEMPT** jobs if the activity continues for 5 minutes or more
- **KILL** jobs if they take more than 5 minutes to preempt
Useful Attributes

› **LoadAvg**
  • Current load average

› **CondorLoadAvg**
  • Current load average generated by HTCondor

› **KeyboardIdle**
  • Seconds since last keyboard or mouse activity
NonCondorLoadAvg = (LoadAvg - CondorLoadAvg)
BgndLoad = 0.3
CPU_Busy = ($(NonCondorLoadAvg) >= $(BgndLoad))
CPU_Idle = (!$(CPU_Busy))
KeyboardBusy = (KeyboardIdle < 10)
KeyboardIsIdle = (KeyboardIdle > 300)
MachineBusy = ($(CPU_Busy) || $(KeyboardBusy))
Desktop Machine Policy

\[
\begin{align*}
\text{START} &= $(CPU\_Idle) \land $(KeyboardIsIdle) \\
\text{SUSPEND} &= $(MachineBusy) \\
\text{CONTINUE} &= $(CPU\_Idle) \land \text{KeyboardIdle} > 120 \\
\text{PREEMPT} &= (\text{Activity} == "Suspended") \land $(ActivityTimer) > 300 \\
\text{KILL} &= $(ActivityTimer) > 300
\end{align*}
\]
Mission Accomplished
Slot States
Slot Activities

Section 3.5: Policy Configuration for the condor_startd

- Preempting
  - Vacating
  - Killing
- Want vacate?
- Claimed
  - Want Suspend?
  - Busy
  - Idle
  - Matched
- Owner
- Drained
  - Idle
  - Retiring
- Unclaimed
  - Benchmarking
- Idle
- Busy
- Killing
- Suspended
- ID
Custom Slot Attributes

› Can add attributes to a slot’s ClassAd, typically done in the local configuration file

\[
\begin{align*}
\text{INSTRUCTIONAL} &= \text{TRUE} \\
\text{NETWORK\_SPEED} &= 1000 \\
\text{STARTD\_EXPRS} &= \text{INSTRUCTIONAL, NETWORK\_SPEED}
\end{align*}
\]
Jobs can now specify Rank and Requirements using new attributes:

Requirements = INSTRUCTIONAL=!*=TRUE
Rank = NETWORK_SPEED

Dynamic attributes are available; see STARTD_CRON_* in the manual
Further Machine Policy Information

- For further information, see section 3.5 “Policy Configuration for the `condor_startd`” in the HTCondor manual
- htcondor-users mailing list
  http://research.cs.wisc.edu/htcondor/mail-lists/
- htcondor-admin@cs.wisc.edu
Job Priority

- Set with `condor_prio`
- Users can set priority of their own jobs
- Integers, larger numbers are higher priority
- Only impacts order between jobs for a single user on a single schedd
- A tool for users to sort their own jobs
User Priority

- Determines allocation of machines to waiting users
- View with `condor_userprio`
- Inversely related to machines allocated (lower is better priority)
  - A user with priority of 10 will be able to claim twice as many machines as a user with priority 20
User Priority

Effective User Priority is determined by multiplying two components

- Real Priority
- Priority Factor
Real Priority

› Based on actual usage
› Defaults to 0.5
› Approaches actual number of machines used over time
  • Configuration setting `PRIORITY_HALFLIFE`
Priority Factor

› Assigned by administrator
  • Set with `condor_userprio`
› Defaults to 1 (`DEFAULT_PRIO_FACTOR`)
Negotiator Policy Expressions

- `PREEMPTION_REQUIREMENTS` and `PREEMPTION_RANK`
- Evaluated when `condor_negotiator` considers replacing a lower priority job with a higher priority job.
- Completely unrelated to the `PREEMPT` expression.
If false will not preempt machine

- Typically used to avoid pool thrashing
- Typically use:
  - `RemoteUserPrio` – Priority of user of currently running job (higher is worse)
  - `SubmittorPrio` – Priority of user of higher priority idle job (higher is worse)

> `PREEMPTION_REQUIREMENTS = FALSE`
PREEMPTION_REQUIREMENTS

› Only replace jobs running for at least one hour and 20% lower priority

\[
\text{StateTimer} = (\text{CurrentTime} - \text{EnteredCurrentState}) \\
\text{HOUR} = (60*60) \\
\text{PREEMPTION_REQUIREMENTS} = \ \ \\
$(\text{StateTimer}) > (1 * $(\text{HOUR})) \ \ \\
& & \text{RemoteUserPrio > SubmittorPrio} \times 1.2 \]
Picks which already claimed machine to reclaim

Strongly prefer preempting jobs with a large (bad) priority and a small image size

\[
\text{PREEMPTION\_RANK} = \setminus (\text{RemoteUserPrio} \times 1000000) \setminus - \text{ImageSize}
\]
Accounting Groups

› Manage priorities across groups of users and jobs
› Can guarantee minimum numbers of computers for groups (quotas)
› Supports hierarchies
› Anyone can join any group
Find current configuration values

% condor_config_val MASTER_LOG
/var/condor/logs/MasterLog

% cd `condor_config_val LOG`
condor_config_val -v

› Can identify source

% condor_config_val -v CONDOR_HOST
CONDOR_HOST: condor.cs.wisc.edu

Defined in
‘/etc/condor_config_config.hosts’, line 6
What configuration files are being used?

```
$ condor_config_val -config
Config source:
   /var/home/condor/condor_config
Local config sources:
   /unsup/condor/etc/condor_config.hosts
   /unsup/condor/etc/condor_config.global
   /unsup/condor/etc/condor_config.policy
   /unsup/condor-test/etc/hosts/puffin.local
```
condor_fetchlog

› Retrieve logs remotely
condor_fetchlog beak.cs.wisc.edu
Master
Checking the current status

› `condor_status`
› `condor_q`
› Greg's “How High Throughput was My Cluster?” this afternoon
Queries the collector for information about daemons in your pool

Defaults to finding `condor_startds`

`condor_status -schedd` summarizes all job queues

`condor_status -master` returns list of all `condor_masters`
condor_status

- `long` displays the full ClassAd
- Optionally specify a machine name to limit results to a single host

```bash
condor_status -l node4.cs.wisc.edu
```
- Do not use in scripts/programs
condor_status -constraint

 › Only return ClassAds that match an expression you specify
 › Show me idle slots with 1GB or more memory
   
   • condor_status -constraint
     'Memory >= 1024 && Activity == "Idle"'
```bash
$ condor_status -af Activity OpSys Arch | sort | uniq -c

<table>
<thead>
<tr>
<th>State</th>
<th>Count</th>
<th>Os</th>
<th>Arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>56</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
<tr>
<td>Idle</td>
<td>35</td>
<td>LINUX</td>
<td>INTEL</td>
</tr>
<tr>
<td>Idle</td>
<td>1515</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
<tr>
<td>Idle</td>
<td>369</td>
<td>WINDOWS</td>
<td>X86_64</td>
</tr>
<tr>
<td>Retiring</td>
<td>31</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
</tbody>
</table>
```

- Report only fields you request
- Census of systems in your pool:

> condor_status -af Activity OpSys Arch | sort | uniq -c
condor_status -autoformat

- Separate by tabs, commas, spaces, newlines
- Label each field by name
- Escape as a ClassAd value
- Add headers
- Several easy to parse options
condor_status -format

› Like autoformat, but with manual formatting

› Useful for writing simple reports

› Uses C printf style formats
  • One field per argument

“slanting” by Stefano Mortellaro (“fazen”) © 2005
Licensed under the Creative Commons Attribution 2.0 license
http://www.flickr.com/photos/fazen/17200735/
http://www.webcitation.org/5XihNWC7Y
```
% condor_status -format '%-10s '
Activity -format '%-7s ' OpSys -format '%s\n' Arch | sort | uniq -c

<table>
<thead>
<tr>
<th>Activity</th>
<th>OpSys</th>
<th>Arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busy</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
<tr>
<td>Idle</td>
<td>LINUX</td>
<td>INTEL</td>
</tr>
<tr>
<td>Idle</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
<tr>
<td>Idle</td>
<td>WINDOWS</td>
<td>X86_64</td>
</tr>
<tr>
<td>Retiring</td>
<td>LINUX</td>
<td>X86_64</td>
</tr>
</tbody>
</table>
```
Examining Queues condor_q

› View the job queue
› The \texttt{-long} option is useful to see the entire ClassAd for a given job
› supports \texttt{-constraint}, \texttt{-autoformat}, and \texttt{-format}
› Can view job queues on remote machines with the \texttt{-name} option
condor_q -analyze and -better-analyze

› Why isn't this job running? default
› On this machine? -machine
› What does this machine hate my job? -better-analyse:reverse
› General reports -analyze:sum -analyze:sum,rev
HTCondor’s Log Files

› HTCondor maintains one log file per daemon

› Can increase verbosity of logs on a per daemon basis
  • SHADOW_DEBUG, SCHEDD_DEBUG, and others
  • Space separated list
Useful Debug Levels

- **DFULLDEBUG** dramatically increases information logged
  - Does not include other debug levels!
- **D_COMMAND** adds information about commands received

```
SHADOW_DEBUG = D_FULLDEBUG D_COMMAND
```
Log files are automatically rolled over when a size limit is reached
• Only one old version is kept
• Defaults to 10 megabytes
• Rolls over quickly with `D_FULLDEBUG`
• `MAX_DEFAULT_LOG`
• Also per daemon settings
  • `MAX_SHADOW_LOG`, `MAX_SCHEDD_LOG`, and others
Many log files entries primarily useful to HTCondor developers

- Especially if D_FULLDEBUG is on
- Minor errors are often logged but corrected
- Take them with a grain of salt
- htcondor-admin@cs.wisc.edu
Debugging Jobs: condor_q

 › Examine the job with condor_q
   • especially the very powerful –analyze and –better-analyze
Debugging Jobs: User Log

› Examine the job’s user log
  • Can find with:
    condor_q -af UserLog 17.0
  • Set with “log” in the submit file
  • You can set EVENT_LOG to get a unified log for all jobs under a schedd

› Contains the life history of the job
› Often contains details on problems
Debugging Jobs: ShadowLog

- Examine *ShadowLog* on the submit machine
  - Note any machines the job tried to execute on
  - There is often an “ERROR” entry that can give a good indication of what failed
Debugging Jobs: Matching Problems

- No ShadowLog entries? Possible problem matching the job.
  - Examine ScheddLog on the submit machine
  - Examine NegotiatorLog on the central manager
Debugging Jobs: Remote Problems

› ShadowLog entries suggest an error but aren’t specific?
  • Examine StartLog and StarterLog on the execute machine
Debugging Jobs: Reading Log Files

› HTCondor logs will note the job ID each entry is for
  • Useful if multiple jobs are being processed simultaneously
  • grepping for the job ID will make it easy to find relevant entries

› Occasionally HTCondor doesn't know yet…
Debugging Jobs: What Next?

› If necessary add “D_FULLDEBUG D_COMMAND” to DEBUG_DAEMONNAME setting for additional log information

› Increase MAX_DAEMONNAME_LOG if logs are rolling over too quickly

› If all else fails, email us
  • htcondor-admin@cs.wisc.edu
More Information

› Staff here at HTCondor Week
› HTCondor Manual
› htcondor-users mailing list
   http://research.cs.wisc.edu/
       htcondor/mail-lists/
› htcondor-admin
   htcondor-admin@cs.wisc.edu
Thank You!

Any questions?

“My mouse” by “MysterFaery” © 2006 Licensed under the Creative Commons Attribution 2.0 license