Job and Machine Policy
Configuration
HTCondor / ARC CE
Workshop
Barcelona 2016
Todd Tannenbaum
Quick Review
ClassAds: Example

```
[
    Type = "Apartment";
    SquareArea = 3500;
    RentOffer = 1000;
    HeatIncluded = False;
    OnBusLine = True;
    Rank = other.RentOffer + other.UnderGrad?0:10;
    Requirements =
        TARGET.RentOffer > (MY.RentOffer - 150);
]
```

```
[
    Type = "Renter";
    UnderGrad = False;
    RentOffer = 900;
    Rank = 1/(other.RentOffer + 100.0);
    Requirements =
        OnBusLine &&
        SquareArea > 1500;
]`
Policy Expressions

- Policy Expressions allow jobs and machines to restrict access, handle errors and retries, perform job steering, set limits, when/where jobs can start, etc.
Assume a simple setup

› Lets assume a pool with only one single user (me!).
  • no user/group scheduling concerns, we’ll get to that later…
We learned earlier...

- Job submit file can specify Requirements and Rank expressions to express constraints and preferences on a match

\[
\text{Requirements} = \text{OpSysAndVer}==\text{“RedHat6”}\\
\text{Rank} = \text{kflops}\\
\text{Executable} = \text{matlab}\\
\text{queue}
\]

- Another set of policy expressions control job status
Job Status Policy Expressions

- User can supply job policy expressions in the job submit file. See condor_submit man page.
- These expressions can reference any job ad attribute.

```plaintext
on_exit_remove = <expression>
on_exit_hold = <expression>
periodic_remove = <expression>
periodic_hold = <expression>
periodic_release = <expression>
```
Job Status State Diagram

Submit

Idle (I)

Running (R)

Completed (C)

Removed (X)

Held (H)
Job Policy Expressions

● Do not remove if exits with a signal:
  \[
  \text{on\_exit\_remove} = \text{ExitBySignal} == \text{False}
  \]

● Place on hold if exits with nonzero status or ran for less than an hour:
  \[
  \text{on\_exit\_hold} = \\
  (\text{ExitCode} != 0) \text{||} \\
  ((\text{time() - JobStartDate}) < 3600)
  \]

● Place on hold if job has spent more than 50% of its time suspended:
  \[
  \text{periodic\_hold} = \\
  (\text{CumulativeSuspensionTime} > \\
  (\text{RemoteWallClockTime} / 2.0))
  \]
Admins can also provide supply periodic job policy expressions in the condor_config file.

These expressions impact all jobs submitted to a specific schedd.

```
system_periodic_remove = <expression>
system_periodic_hold = <expression>
system_periodic_release = <expression>
```

What is the period? Frequency of evaluation is configurable via a floor (1 minute), max (20 minutes), and schedd timeslice (1%).
How do you specify Requirements and Rank for machine slots?

- Specified in condor_config
- Machine slot policy (or ‘startd policy’) expressions can reference items in either the machine or candidate job ClassAd (See manual appendix for list)
Some Startd Expressions (when to start/stop jobs)

- START = <expr>
- RANK = <expr>
- SUSPEND = <expr>
- CONTINUE = <expr>
- PREEMPT = <expr>  \((\text{really means evict})\)
  - And the related WANT_VACATE = <expr>
Startd’s START

› START is the primary policy
› When FALSE the machine enters the Owner state and will not run jobs
› Acts as the Requirements expression for the machine, the job must satisfy START
  • Can reference job ClassAd values including Owner and ImageSize
Startd’s RANK

› Indicates which jobs a machine prefers

› Floating point number, just like job rank
  • 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 machines still advertise looking for higher ranked job to preempt the current job
  • LESSON: Startd Rank creates job preemption
Startd’s PREEMPT

› Really means vacate (I prefer nothing vs this job!)
› When PREEMPT becomes true, the job will be killed and go from Running to Idle
› Can “kill nicely”
  • WANT_VACATE = <expr>; if true then send a SIGTERM and follow-up with SIGKILL after MachineMaxVacateTime seconds.

Startd’s Suspend and Continue

› When True, send SIGSTOP or SIGCONT to all processes in the job
Default Startd Settings

› Always run jobs to completion

START = True
RANK = 0
PREEMPT = False
SUSPEND = False
CONTINUE = True

OR

use policy: always_run_jobs
Policy Configuration

› I am adding special new nodes, only for simulation jobs from Math. If none, simulations from Chemistry. If none, simulations from anyone.
Prefer Chemistry Jobs

\[ \text{START} = \text{KindOfJob} =?= \text{“Simulation”} \]

\[ \text{RANK} = \]

\[ 10 \times \text{Department} =?= \text{“Math”} + \]
\[ \text{Department} =?= \text{“Chemistry”} \]

\[ \text{SUSPEND} = \text{False} \]

\[ \text{PREEMPT} = \text{False} \]
Don’t let any job run longer than 24 hrs, except Chemistry jobs can run for 48 hrs.
Settings for showing runtime limits

\[
\text{START} = \text{True} \\
\text{RANK} = 0 \\
\text{PREEMPT} = \text{TotalJobRunTime} > \text{ifThenElse(Department=?="Chemistry", 48 \times (60 \times 60), 24 \times (60 \times 60))}
\]

Note: this will result in the job going back to Idle in the queue to be rescheduled.
Runtime limits with a chance to checkpoint

\[
\text{START} = \text{True} \\
\text{RANK} = 0 \\
\text{PREEMPT} = \text{TotalJobRunTime} > \\
\quad \text{ifThenElse(} \text{Department=?="Chemistry"}, \\
\quad \quad 48 \times (60 \times 60), \\
\quad \quad 24 \times (60 \times 60)) \\
\text{WANT_VACATE} = \text{True} \\
\text{MachineMaxVacateTime} = 300
\]

Wonder if the user will have any idea why their jobs was evicted....
Runtime limits with job hold

START = True
RANK = 0

TIME_EXCEEDED = TotalJobRunTime >
 ifThenElse (Department=?="Chemistry", 
 48 * (60 * 60),
 24 * (60 * 60) )

PREEMPT = $(TIME_EXCEEDED)
WANT_HOLD = $(TIME_EXCEEDED)
WANT_HOLD_REASON =
 ifThenElse( Department=?="Chemistry", 
 "Chem job failed to complete in 48 hrs", 
 "Job failed to complete in 24 hrs" )
C:\temp>condor_q

-- Submitter: ToddsThinkpad : <127.0.0.1:49748> : ToddsThinkpad

  ID    OWNER  SUBMITTED  RUN_TIME ST  PRI  SIZE  CMD
  1.0  tannenba  12/5 17:29 0+24:00:03 H 0 0.0  myjob.exe

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

C:\temp>condor_q -hold

-- Submitter: ToddsThinkpad : <127.0.0.1:49748> : ToddsThinkpad

  ID    OWNER  HELD_SINCE  HOLD_REASON
  1.0  tannenba  12/6 17:29  Job failed to complete in 24 hrs

1 jobs; 0 completed, 0 removed, 0 idle, 0 running, 1 held, 0 suspended
Could we implement via job policy instead of startd policy?

Yes. Put in condor_config on submit host:

```python
SYSTEM_PERIODIC_HOLD =
(time() - JobStartTime) > (24*60*60))
SYSTEM_PERIODIC_HOLD_REASON =
"Job failed to complete in 24 hrs"
```

Which to use?

- You may only have control of one or the other
- Startd policy evaluated more often (every 5 secs)
- Consider if policy is associated with the job or with the machine – keep responsibilities of both in mind
Custom Attributes in Slot Ads

› Several ways to add custom attributes into your slot ads
  • From the config file(s) (for static attributes)
  • From a script (for dynamic attributes)
  • From the job ad of the job running on that slot
  • From other slots on the machine

› Can add a custom attribute to all slots on a machine, or only specific slots
Custom Attributes from config file (static attributes)

- Define your own slot attributes that jobs can match against.
- If you want the slot to have HasMatlab = true
  
  Define value in config, and then add name to STARTD_EXPRS (or _ATTRS), like this

  \[
  \text{STARTD_EXPRS} = \$(\text{STARTD_EXPRS}) \text{ HasMatlab HasMatlab} = \text{true} \\
  \text{Or SLOTX_HasMatlab} = \text{true}
  \]

- Note: Also SUBMIT_EXPRS, SCHEDD_EXPRS, MASTER_EXPRS, …
Custom Attributes from a script (for dynamic attrs)

› Run a script/program to define attributes
  • Script returns a ClassAd
  • Attributes are merged into Slot ClassAds
    STARTD_CRON_JOB_LIST = tag
    STARTD_CRON_tag_EXECUTABLE = detect.sh

› Run once or periodically

› Control which slots get the attributes with SlotMergeConstraint or SlotId
Custom attributes from job classad running on the slot

- Can take attribute X from job ad and publish it into slot ad when job is running on this slot.

- condor_config:
  ```
  STARTD_JOB_EXPRS = $(STARTD_JOB_EXPRS) CanCheckPoint
  ```

- Example submit file:
  ```
  executable = foo
  +CanCheckPoint = True
  queue
  ```

- Now can reference attributes of the job in machine policy and PREEMPTION_REQUIREMENTS, e.g.
  ```
  PREEMPT = CanCheckPoint =?= True
  ```
Cross publishing Slot Attributes

- Policy expressions sometimes need to refer to attributes from other slots
- Cross-publish with `STARTD_SLOT_ATTRS`
  
  ```
  STARTD_SLOT_ATTRS = State, Activity
  ```

  Now all slots can see Slot1_State, Slot2_state,…

- Each slot’s attrs published in ALL slot ads with SlotN_X, so you can do this:
  
  ```
  START = $(START) && (SlotId==2 && Slot1_State != "Claimed") && ...
  ```
Default Behavior: One static slot per core

- One static execution slot per CPU core, such that each slot is single core slot
- Other resources (Disk, Memory, etc) are divided evenly among the slots

- How can I customize this?
It is easy to Lie!

- Set Arbitrary values for CPUs, Memory
- HTCondor will allocate resources to slots as if the values are correct
  
  ```
  NUM_CPUS = 99
  MEMORY = \n  $(DETECTED_MEMORY) * 99
  ```

- Default values:
  
  ```
  NUM_CPUS = $(DETECTED_CPUS)
  MEMORY = $(DETECTED_MEMORY)
  ```
Control how resources are allocated to slots

Define up to 10 slot ‘types’

For each slot type you can define

• A name prefix for the slot (defaults to “slot”)
• The number of slots of that type to create
• How many machine resources each slot should contain
• Policy knobs (START, PREEMPT, etc) per slot type
Why?

- Perhaps to match your job load

Examples

• Each slot has two cores

```
NUM_SLOTS_TYPE_1 = $(DETECTED_CPUS)/2
SLOT_TYPE_1 = Cpus=2
```

• Non-uniform static slots: Make one “big” slot, the rest in single core slots

```
NUM_SLOTS_TYPE_1 = 1
SLOT_TYPE_1 = Cpus=4, Memory=50%
NUM_SLOTS_TYPE_2 = $(DETECTED_CPUS)-4
SLOT_TYPE_2 = Cpus=1
```
How to steer single core jobs away from the big multi-core slots

- Via job ad (if you control submit machine...)
  
  \[ \text{DEFAULT\_RANK} = \text{RequestCPUs} - \text{CPUs} \]

- Via \text{condor\_negotiator} on central manager
  
  \[ \text{NEGOTIATOR\_PRE\_JOB\_RANK} = \backslash \text{RequestCPUs} - \text{CPUs} \]
Another why – Special Purpose Slots

Slot for a special purpose, e.g. data movement, maintenance, interactive jobs, etc…

# Lie about the number of CPUs
NUM_CPUS = $(DETECTED_CPUS)+1

# Define standard static slots
NUM_SLOTS_TYPE_1 = $(DETECTED_CPUS)

# Define a maintenance slot
NUM_SLOTS_TYPE_2 = 1
SLOT_TYPE_2 = cpus=1, memory=1000
SLOT_TYPE_2_NAME_PREFIX = maint
SLOT_TYPE_2_START = owner=="tannenba"
SLOT_TYPE_2_PREEMPT = false
C:\home\tannenba>condor_status

<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>maint5@ToddsThinkp</td>
<td>WINDOWS</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>0.000</td>
<td>1000</td>
<td>0+00:00:08</td>
<td></td>
</tr>
<tr>
<td>slot1@ToddsThinkpa</td>
<td>WINDOWS</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>0.000</td>
<td>1256</td>
<td>0+00:00:04</td>
<td></td>
</tr>
<tr>
<td>slot2@ToddsThinkpa</td>
<td>WINDOWS</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>0.110</td>
<td>1256</td>
<td>0+00:00:05</td>
<td></td>
</tr>
<tr>
<td>slot3@ToddsThinkpa</td>
<td>WINDOWS</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>0.000</td>
<td>1256</td>
<td>0+00:00:06</td>
<td></td>
</tr>
<tr>
<td>slot4@ToddsThinkpa</td>
<td>WINDOWS</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>0.000</td>
<td>1256</td>
<td>0+00:00:07</td>
<td></td>
</tr>
</tbody>
</table>

Total Owner Claimed Unclaimed Matched Preempting Backfill

| X86_64/WINDOWS | 5 | 0 | 0 | 5 | 0 | 0 | 0 |

Total 5 0 0 5 0 0 0
Define a custom resource

Define a custom STARTD resource

• `MACHINERESOURCE_<tag>`

Can come from a script (if you want dynamic discovery)

• `MACHINERESOURCE_INVENTORY_<tag>`
Fungible resources or "Unnamed" resources

› For OS resources you don't need to know a name to use
  • Cpu cores, Memory, Disk
› For intangible resources
  • Bandwidth
  • Licenses?
› Works with Static and Partitionable slots
Unnamed custom resource example : bandwidth (1)

> condor_config_val -dump Bandwidth
MACHINE_RESOURCE_Bandwidth = 1000

> grep -i bandwidth userjob.submit
REQUEST_Bandwidth = 200
Unnamed custom resource example: bandwidth (2)

Assuming 4 static slots

```bash
> condor_status -long | grep -i bandwidth
Bandwidth = 250
DetectedBandwidth = 1000
TotalBandwidth = 1000
TotalSlotBandwidth = 250
```
Non-fungible resources or Named resources

› For resources not assigned by OS, and thus need to be assigned via name
  • GPUs, Instruments, Directories

› Configure by listing resource ids
  • Quantity is inferred

› Specific id(s) are assigned to slots

› Works with Static and Partitionable slots
Named custom resource example: GPUs (1)

```bash
> condor_config_val -dump gpus
MACHINE_RESOURCE_GPUs = CUDA0, CUDA1
ENVIRONMENT_FOR_AssignedGPUs = GPU_NAME GPU_ID=/CUDA/
ENVIRONMENT_VALUE_FOR_UnAssignedGPUs = 10000

> grep -i gpus userjob.submit
REQUEST_GPUs = 1
```

Or

use feature: GPUs
Named custom resource example : GPUs (2)

> condor_status -long slot1 | grep -i gpus
AssignedGpus = "CUDA0"
DetectedGPUs = 2
GPUs = 1
TotalSlotGPUs = 1
TotalGPUs = 2
Named custom resource example:

: GPUs (3)

❯ Environment of a job running on that slot

```
> env | grep -I CUDA
_CONDOR_AssignedGPUs = CUDA0
GPU_NAME = CUDA0
GPU_ID = 0
```
Non-uniform static slots help, but not perfect…

8 Gb machine partitioned into 5 static slots

4 Gb Slot 1Gb 1Gb 1Gb 1Gb

4 Gb Job 1Gb 1Gb 1Gb 1Gb
8 Gb machine partitioned into 5 static slots

- 4Gb Slot
- 1Gb
- 1Gb
- 1Gb
- 1Gb

- 1Gb
- 1Gb
- 1Gb
- 1Gb
- 1Gb
8 Gb machine partitioned into 5 static slots

4Gb Slot 1Gb 1Gb 1Gb 1Gb

1Gb

7 Gb free, but idle job

4 Gb Job
Partitionable Slots: The big idea

- One parent “partitionable” slot
- From which child “dynamic” slots are made at claim time
- When dynamic slots are released, their resources are merged back into the partitionable parent slot
Partionable slots split on
• Cpu
• Disk
• Memory
• (plus any custom startd resources you defined)

When you are out of CPU or Memory, you’re out of slots
3 types of slots

› Static (e.g. the usual kind)
› Partitionable (e.g. unclaimed resources)
› Dynamic (claimed slots carved off a partitionable slot parent)
  • Dynamically created at claim time
  • But once created, act as static
  • When unclaimed, resources go back to partitionable parent
8 Gb Partitionable slot
7 Gb Partitionable slot

1Gb
6 Gb Partitionable slot

1Gb  1Gb
5 Gb Partitionable slot

1Gb 1Gb 1Gb
0 Gb Partitionable slot

1Gb 1Gb 1Gb 1Gb 1Gb 1Gb 1Gb 1Gb
4 Gb Partitionable slot

1Gb  1Gb  1Gb  1Gb
1 Gb Partitionable slot

1Gb  1Gb  1Gb  4Gb
How to configure

NUM_SLOTS = 1
NUM_SLOTS_TYPE_1 = 1
SLOT_TYPE_1 = cpus=100%
SLOT_TYPE_1_PARTITIONABLE = true
Looks like

```
$ condor_status

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot1@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.110</td>
<td>8192</td>
</tr>
</tbody>
</table>

Total          | Owner | Claimed | Unclaimed | Matched |
X86_64/LINUX   | 1     | 0       | 0         | 1        |
Total          | 1     | 0       | 0         | 1        |
```
When running

```bash
$ condor_status
```

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot1@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.110</td>
<td>4096</td>
</tr>
<tr>
<td>slot1_1@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>1024</td>
</tr>
<tr>
<td>slot1_2@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>2048</td>
</tr>
<tr>
<td>slot1_3@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>1024</td>
</tr>
</tbody>
</table>
Can specify default Request values

JOB_DEFAULT_REQUEST_CPUS
JOB_DEFAULT_REQUEST_MEMORY
JOB_DEFAULT_REQUEST_DISK
## Fragmentation

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot1@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.110</td>
<td>4096</td>
</tr>
<tr>
<td>slot1_1@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>2048</td>
</tr>
<tr>
<td>slot1_2@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>1024</td>
</tr>
<tr>
<td>slot1_3@c</td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.000</td>
<td>1024</td>
</tr>
</tbody>
</table>

Now I submit a job that needs 8G – what happens?
Solution: Draining

- `condor_drain`
- `condor_defrag`
  
  - One daemon defrags whole pool
  - Central manager good place to run
  - Scan pool, try to fully defrag some startds by invoking `condor_drain` to drain machine
  - Only looks at partitionable machines
  - Admin picks some % of pool that can be “whole”

Note: some heuristics can reduce need to defrag, such as packing large jobs on high number nodes and low jobs on low number
Oh, we got knobs…

DEFRAG_DRAINING_MACHINES_PER_HOUR

DEFRAG_MAX_WHOLE_MACHINES

DEFRAG_SCHEDULE

• graceful (obey MaxJobRetirementTime, default)
• quick (obey MachineMaxVacateTime)
• fast (hard-killed immediately)
Match Only Multicore Jobs to recently drained machines

- Since the purpose of the defrag daemon is to drain jobs on a p-slot so multi-core jobs can begin to match, it would be best to implement a policy where recently drained p-slots can insist on matching only multicore jobs for a period of time.

- See https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=HowToMatchMulticoreAfterDrain

- Many other good HOWTO recipes on homepage
Future work on Partitionable Slots

› See my talk Wednesday!
Further Information

› For further information, see section 3.5 “Policy Configuration for the condor_startd” in the Condor manual

› HTCondor HOWTOs recipes at http://htcondor.org.

› htcondor-users mailing list
  • http://htcondor.org/mail-lists/