Scheduling Policy

John (TJ) Knoeller
Condor Week 2017

Overview

- Policy options in the SCHEDD
  - Limits
  - Job policy
  - Mutating jobs
  - Preventing changes

Limits

- Max jobs running
- Max jobs per submission
- Max jobs per Owner (8.6)
- Max running DAGs per Owner (8.6)
- Max materialized jobs per cluster (8.7.1)
- Max active input transfers
- Max active output transfers

User vs Owner vs Submitter

- Owner attribute of job is OS ‘user’
  - Shadow impersonates Owner for file i/o
  - Set by SCHEDD based on submit identity
  - Immutable
- Accounting ‘user’ a.k.a. Submitter
  - Who’s quota/priority is checked/docked
  - (Owner + Nice) + Domain + AccountingGroup
  - User can change at will

Most limits are Submitter limits

- “Fair” share is by submitter
  - Negotiator only knows about submitters
  - Priority / Quota
  - Transfer queue
- A few per-owner limits
  - Max jobs per owner (8.6)
  - Max running DAGs per owner (8.6)

Monitoring the limits

- Todd has a talk on this
- Schedd Stats
  - condor_status –schedd –direct -long
- Per submitter stats
  - condor_status –submit –long
  - condor_sos condor_q –tot –long
- Show jobs doing file transfer
  - condor_sos condor_q –io

Job policy

- You want to have a policy about what jobs are allowed, or require certain attributes?
  - Submit requirements
  - Submit attributes
  - Job transforms
  - system_periodic_remove/hold/release
    - covered in "Job and Machine Policy" talk

Example job policy

- All jobs must have "Experiment" attribute
  - Reject jobs that don't conform to the policy

```
SUBMIT_REQUIREMENT_NAMES = $(SUBMIT_REQUIREMENT_NAMES) CheckExp
SUBMIT_REQUIREMENT_CheckExp = \
    JobUniverse == 7 || Experiment isnt undefined
SUBMIT_REQUIREMENT_CheckExp_REASON = \
   "submissions must have +Experiment"


# JobUniverse 7 is Scheduler universe, i.e. DAGMAN.
# JobUniverse 12 is Local universe, maybe except this also?
```

Defaulting job attributes

- Configure SUBMIT_ATTRS to add attributes to jobs.

```
SUBMIT_ATTRS = $(SUBMIT_ATTRS) Experiment
Experiment = "CHTC"
```

- Job ad starts with **`Experiment="CHTC"`** before the submit file is processed

SUBMIT_ATTRS

- Good for setting defaults
- Work happens outside of the SCHEDD
- User can override or un-configure
- Unconditional
- May not happen with remote submit
  (Depends on who owns the config)

Mutating jobs using job transforms (new in 8.6)

› Configure JOB_TRANSFORM_*

```
JOB_TRANSFORM_NAMES = $(JOB_TRANSFORM_NAMES) SetExp
JOB_TRANSFORM_SetExp = [ set_Experiment = "CHTC"; ]
```

› **`Experiment="CHTC"`** written into each job ad as it is submitted.
probably not a good thing in this case

Transforming only some jobs

```
JOB_TRANSFORM_NAMES = $(JOB_TRANSFORM_NAMES) SetExp
JOB_TRANSFORM_SetExp @=end
[
  Requirements = JobUniverse != 7 && Experiment is undefined
  set_Experiment = "CHTC";
]
@end
```

› Adds **`Experiment="CHTC"`** to each job that doesn't already have that attribute

About job transforms

- Converted to native syntax on startup
- Job router syntax is loosely ordered
  - copy > delete > set > eval_set
- Native syntax is
  - Confusing (and might be changing)
  - Top to bottom
  - Has temporary variables
  - Has Conditionals

Job transform native syntax

```
# Use job transform to add pool constraint to vanilla jobs
# based on whether the job needs GPUs or not
#
JOB_TRANSFORM_GPUS @=end
  REQUIREMENTS JobUniverse == 5
  tmp.NeedsGpus = $(MY.RequestGPUs:0) > 0
  if $INT(tmp.NeedsGpus)
    SET Requirements $(MY.Requirements) && (Pool == "ICECUBE")
  else
    SET Requirements $(MY.Requirements) && (Pool == "CHTC")
  endif
@end
```

Preventing change

- IMMUTABLE_JOB_ATTRS
  - Cannot be changed once set
- PROTECTED_JOB_ATTRS
  - Cannot be changed by the user
- SECURE_JOB_ATTRS
  - Like protected, but have security implications

```
IMMUTABLE_JOB_ATTRS=$(IMMUTABLE_JOB_ATTRS) Experiment
```

The motivating case for all this

- How do I assign jobs to accounting groups automatically, while preventing users from cheating?
  - Job transforms + Immutable attributes
- But doing this in classad language is *painful*

```
eval_set_AcctGroup=\
  IfThenElse(Owner=="Bob","CHTC",
    IfThenElse(Owner=="Alice","Math",
      IfThenElse(Owner=="Al","Physics","Unknown")
  ))
```

Introducing Map files

› Map file is text, with 3 fields per line

› * <key_or_regex> <result_list>

```
* Bob       CHTC, Security
* Alice     CHTC, Math, Physics
* /.*Hat/i Problem
* /.*/      CHTC
```

› Yes, the first field must be *

Defining a map

```
SCHEDD_CLASSAD_USER_MAP_NAMES = MyMap

CLASSAD_USER_MAPFILE_MyMap = /path/to/mapfile
            <or>
SCHEDD_CLASSAD_USER_MAPDATA_MyMap @=end
 * Bob CHTC,Security
 * Alice CHTC,Math,Physics
 * /.*Hat/i Problem
 * /.*/ CHTC
@end
```

Can now use the userMap("MyMap") function in Classad expressions in the SCHEDD.

The Classad userMap function

```
result = userMap(mname, input)
```

- map input to first result

```
result = userMap(mname, input, preferred)
```

- map input to preferred result

```
result = userMap(mname, input, pref, def)
```

- map input to preferred or default result

Putting it all together

```
SCHEDD_CLASSAD_USER_MAP_NAMES = $(SCHEDD_CLASSAD_USER_MAP_NAMES) Groups
CLASSAD_USER_MAPFILE_Groups = /path/to/mapfile

# Assign groups automatically
JOB_TRANSFORM_NAMES = AssignGroup
JOB_TRANSFORM_AssignGroup @=end
 [
  copy_Owner="AcctGroupUser";
  copy_AcctGroup="RequestedAcctGroup";
  eval_set_AcctGroup=usermap("AssignGroup",AcctGroupUser,AcctGroup);
]
@end

# Prevent Cheating
IMMUTABLE_JOB_ATTRS = $(IMMUTABLE_JOB_ATTRS) AcctGroup AcctGroupUser
SUBMIT_REQUIREMENT_NAMES = $(SUBMIT_REQUIREMENT_NAMES) CheckGroup
SUBMIT_REQUIREMENT_CheckGroup = AcctGroup isnt undefined
SUBMIT_REQUIREMENT_CheckGroup_REASON = strcat("Could not map '", Owner, "' to a group")
```

Or, to put it another way

```
use FEATURE:AssignAccountingGroup(/path/map)
```

You can run

```
 condor_config_val use feature:AssignAccountingGroup
```

to see what this metaknob expands to

CENTER FOR
HIGH THROUGHPUT
COMPUTING

HTCondor

Any Questions?