Condor tunning

by Igor Sfiligoi (UCSD)

Regulating User Priorities

User priorities

- By default, the Negotiator treats all users equally
  - You get fair-share out of the box
    If you have two users, on average each gets ½ of slots
- If you have different needs, Negotiator supports
  - Priority factors
    - PF reduces user priority
    - If users X and Y have $PF_X=(N-1)*PF_Y$, on average user X gets $^1/_N$ of slots (with user Y the rest)
  - Accounting groups (see next slide)

http://research.cs.wisc.edu/condor/manual/v7.6/3_4User_Priorities.html

Accounting groups

- Users can be joined in accounting groups
  - The Negotiator defines the groups,
    but jobs specify which group they belong to
- Each group can be given a quota
  - Can be relative to the size of the pool
  - Sum of group jobs cannot exceed it
- If quotas >100%, can be used for relative prio
  - Here higher is better
  - Each group will be given,
    on average, $quota_G / sum(quotas)$ of slots

Jobs without any group may never get anything

Configuring AC

- To enable accounting groups, just define the quotas in the Negotiator config file
  - Which jobs go into which group not defined here

```
#condor_config.local

# 1.0 is 100%
GROUP_QUOTA_DYNAMIC_group_higgs = 3.2
GROUP_QUOTA_DYNAMIC_group_b =    2.2
GROUP_QUOTA_DYNAMIC_group_k =    6.5
GROUP_QUOTA_DYNAMIC_group_susy =  8.8
```

Using the AC

- Users must specify which group they belong to
  - No automatic mapping or validation in Condor
  - Based on trust
- Jobs must add to their submit file
  +AccountingGroup = "<group>.<user>"

```
Universe   = vanilla
Executable = cosmos
Arguments  = -k 1543.3
Output     = cosmos.out
Input      = cosmos.in
Log        = cosmos.log
+AccountingGroup = "group_higgs.frieda"
Queue 1
```

Configuring PF

- Priority factors a dynamic property
  - Set by cmdline tool `condor_userprio`

Anyone can read
Can only be set
by the Negotiator
administrator

```
$ condor_userprio -all -allusers |grep user1@node1
user1@node1        8016.22     8.02          10.00     0     15780.63 11/23/2011 05:59 12/30/2011 20:37
$ condor_userprio -setfactor user1@node1 1000
The priority factor of user1@node1 was set to 1000.000000
$ condor_userprio -all -allusers |grep user1@node1
user1@node1        8016.22     8.02        1000.00     0     15780.63 11/23/2011 05:59 12/30/2011 20:37
```

  - Although persistent between Negotiator restarts
  - May want to set higher default PF (e.g. 1000)
    - default of 1, and user PF cannot go below
    - Attribute regulated by **`DEFAULT_PRIO_FACTOR`**

http://research.cs.wisc.edu/condor/manual/v7.6/2_7Priorities_Preemption.html#sec:user-priority-explained

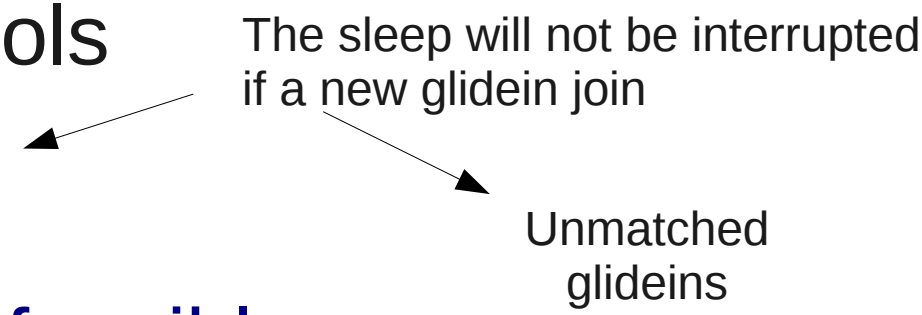
Negotiation cycle optimization

Negotiation internals

- The Negotiator does the matchmaking in loops
  - Get all Machine ClassAds (from Collector)
  - Get Job ClassAds (from Schedds)
    one by one, prioritized by user
    - Tell Schedd if a match has been found
  - Sleep
  - Repeat

Sleep time

- The Negotiator sleeps between cycles to save CPU
  - But will be interrupted if new jobs are submitted
- Makes sense for static pools
  - But can be a problem for dynamic ones like glideins
- Keep the sleep as low as feasible
  `NEGOTIATOR_INTERVAL`
  - Installer sets it to 60s




```
NEGOTIATOR_INTERVAL = 60
```

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:NegotiatorInterval

Preemption

- By default, Condor will preempt a user as soon as jobs from a higher priority users are available
  - Preemption == wasted CPU ← Preemption==killing jobs
- The glideinWMS installer disables preemption
  - Once a glidein is given to a user, he can keep it until the glidein dies

```
# Prevent preemption
PREEMPTION_REQUIREMENTS = False
# Speed up Matchmaking
NEGOTIATOR_CONSIDER_PREEMPTION = False
```

Glidein lifetime is limited, so fair share still enforced long term

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:NegotiatorConsiderPreemption

Protecting against dead glideins

- Glideins can die
  - Just a fact of life on the Grid
- But their ClassAds will stay in the Collector for about 20 mins
  - If they were Idle at the time, will match jobs!
- Make sure the lowest priority user get stuck
  - Order Machine ClassAds by freshness

```
NEGOTIATOR_POST_JOB_RANK = MY.LastHeardFrom
```

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:NegotiatorPostJobRank

Protecting against slow schedds

- If one schedd gets stuck, you want to limit the damage, and match jobs from other schedds
  - Negotiator can limit the time spent on each schedd

Was really important before 7.6.X... less of an issue now

- The glideinWMS installer sets these limits
  - See Condor manual for details

```
NEGOTIATOR_MAX_TIME_PER_SUBMITTER=60
NEGOTIATOR_MAX_TIME_PER_PIESPIN=20
```

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:NegotiatorMaxTimePerSubmitter

One way matchmaking

- The Negotiator normally sends the match both the Schedd and the Startd
  - Only Schedd strictly needed
- Given that the glideins can be behind a firewall, it is a good idea to disable this behavior

```
NEGOTIATOR_INFORM_STARTD = False
```

Collector tunning

Collector tunning

- The major tunning is the Collector tree
  - Already discussed in the Installation section
- Three more optional tunnings
  - Security session lifetime
  - File descriptor limits
  - Client handling parameters

Security session lifetime

- All communication to the Collector (leafs) is GSI based
- But GSI handshakes are expensive!
  - Condor thus uses it just to exchange a shared secret (i.e. password) with the remote glidein
  - Will use this passwd for all further communications
  - But the passwd also has a limited lifetime
- Default lifetime quite long
  - The glideinWMS installer limits it to 12h

To limit damage in case it is stolen

Was months in old versions of Condor!

```
SEC_DAEMON_SESSION_DURATION = 50000
```

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:SecDefaultSessionDuration
http://iopscience.iop.org/1742-6596/219/4/042017

File descriptor limit

- The Collector (leafs) also function as CCB
  - Will use 5+ FDs for every glidein they serve!
- By default, OS only gives 1024 FDs to each process
  - But Condor will limit itself to ~80% of that
  - So it has a few FDs left for internal needs (e.g. logs)
- You **may** want to increase this limit
  - Must be done as root
    (and Collector usually not root)
- And don't forget the system-wide limits
  `/proc/sys/fs/file-max`

Not done by glideinWMS installer

The alternative is to just run more secondary collectors **(recommended)**

Client handling limits

- Every time someone runs `condor_status`
  the Collector must act on it
  - A query is also done implicitly when you run `condor_q -name <schedd_name>`

Used by the Frontend processes

- Since the Collector is a busy process, the Condor team added the option to fork on query
  - This will use up RAM!
  - So there is a limit **`COLLECTOR_QUERY_WORKERS`**

It only uses RAM if the parent memory pages change
But not unusual on a busy Collector

Defaults to 16

You may want to either lower or increase it

Firewall tunning

- If you decided you want to keep the firewall you may need to tune it
  - Open ports for the Collector tree
  - And possibly the Negotiator (if Schedds outside the perimeter)
- You may also need to tune the firewall scalability knobs

Negotiator port usually dynamic... will need to fix it with
NEGOTIATOR_ARGS = -f -p 9617

Statefull firewalls will have to track 5+ TCP connections per glidein

```
# for iptables
sysctl -w net.ipv4.netfilter.ip_conntrack_max=385552;
sysctl -p;
echo 48194 > /sys/module/ip_conntrack/parameters/hashsize
```

Schedd tunning

Schedd tunning

- The following knobs may need to be tuned
  - Job limits
  - Startup/Termination limits
  - Client handling limits
  - Match attributes
  - Requirements defaults
  - Periodic expressions
- You also may need to tune the system settings

Job limits

- As you may remember, each running job uses a non-negligible amount of resources
  - Thus the Schedd has a limit on them **MAX_JOB_RUNNING**
- Set it to a value that is right for your HW

You may waste some CPU on glideins, but better than crashing the submit node

```
# glideinWMS installer provided value
MAX_JOBS_RUNNING = 6000
```

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:MaxJobRunning

Startup limits

- Each time a job is started
  - The Schedd must create a shadow
  - The Shadow must load the input files from disk
  - The Shadow must transfer the files over the network
- If too many start at once, may overload the node
- Two sets of limits:
  - **JOB_START_DELAY/JOB_START_COUNT**
  - **MAX_CONCURRENT_UPLOADS**

Limit at the Schedd

Limit on the network

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:JobStartCount
http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:MaxConcurrentUploads

Termination limits

- Termination can be problematic as well
  - Output files must be transferred back
    - And you do not know how big will it be!
  - If using glexec, an authentication call will be made on each glidein
- Similar limits:
  - **JOB_STOP_DELAY/JOB_STOP_COUNT**
  - **MAX_CONCURRENT_DOWNLOADS**

Can easily overload the submit node.

Each calling the site GUMS. Imagine O(1k) within a second!

Limit at the Schedd

Limit on the network

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:JobStopCount
http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:MaxConcurrentDownloads

Client handling limits

- Every time someone runs `condor_q`
  the Schedd must act on it

Used by the Frontend processes

- Since the Schedd is a busy process, the Condor team added the option to fork on query
  - This will use up RAM!
  - So there is a limit **`SCHEDD_QUERY_WORKERS`**

It only uses RAM if the parent memory pages change
But not unusual on a busy Schedd

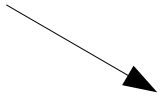
Defaults to 3

You will want to increase it

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:ScheddQueryWorkers

Match attributes for history

- It is often useful to know where were the jobs running
  - By default, you get just the host name **`LastRemoteHost`**
  - May be useful to know which glidein factory it came from, too (etc.)
- But you can get match info easily if you ...
  - Add to the job submit files **job_attribute = $$(glidein attribute)**



After matching, Schedd adds in the ClassAd **`MATCH_EXP_job_attribute`**

System level addition

- You can add them in the system config

```
# condor_config.local
JOB_GLIDEIN_Factory = "$$(GLIDEIN_Factory:Unknown)"
JOB_GLIDEIN_Name = "$$(GLIDEIN_Name:Unknown)"
JOB_GLIDEIN_Entry_Name = "$$(GLIDEIN_Entry_Name:Unknown)"
JOB_GLIDEIN_ClusterId = "$$(GLIDEIN_ClusterId:Unknown)"
JOB_GLIDEIN_ProcId = "$$(GLIDEIN_ProcId:Unknown)"
JOB_GLIDECLIENT_Name = "$$(GLIDECLIENT_Name:Unknown)"
JOB_GLIDECLIENT_Group = "$$((GLIDECLIENT_Group:Unknown)"
JOB_Site = "$$(GLIDEIN_Site:Unknown)"

SUBMIT_EXPRS = $(SUBMIT_EXPRS) \
   JOB_GLIDEIN_Factory JOB_GLIDEIN_Name \
   JOB_GLIDEIN_Entry_Name \
   JOB_GLIDEIN_ClusterId JOB_GLIDEIN_ProcId \
   JOB_GLIDECLIENT_Name JOB_GLIDECLIENT_Group \
   JOB_Site
```

Factory identification

Glidein identification
(to give the Factory admins)

Frontend identification

Any other attribute

This is how you push them in the user job ClassAds

Additional job defaults

- With glideins, we often want empty user job requirements
  - But Condor does not allow it
- It will forcibly add something about
  - Arch ← e.g. (Arch == "INTEL")
  - Disk ← e.g. (Disk>DiskUsage)
  - Memory ← e.g. (Memory>ImageSize)
- Unless they are already there... so lets add them

```
# condor_config.local
APPEND_REQ_VANILLA = (Memory>=1)&&(Disk>=1)&&(Arch=!="fake")
```

Just to make sure it always evaluates to True

Periodic expressions

- Unless you have very well behaved users, some of the jobs will be pathologically broken
  - You want to detect them and remove the from the system

The will be wasting CPU cycles at best

- Condor allows for periodic expressions

SYSTEM_PERIODIC_REMOVE — Just remove

SYSTEM_PERIODIC_HOLD — Leave in the queue, but do not match

  - An arbitrary boolean expression
  - What to look for comes with experience

Example periodic expression

```
# condor_config.local
SYSTEM_PERIODIC_HOLD = \
  ((ImageSize>2000000) || \
   (DiskSize>1000000) || \
   (( JobStatus == 2 ) &&  \
    ((CurrentTime-JobCurrentStartDate ) > 100000)) || \
   (JobRunCount>5))

SYSTEM_PERIODIC_HOLD_REASON = \
   ifThenElse((ImageSize>2000000), \
                  “Used too much memory”, \
    … \
    ifThenElse((JobRunCount>5), \
                   “Tried too many times”, \
                   “Reason unknown”)...)
```

Too much memory

Too much disk space

Took too long

Tried too many times

In Condor 7.7.X
you can also provide
a human readable reason string

http://www.cs.wisc.edu/condor/manual/v7.6/3_3Configuration.html#param:SystemPeriodicHold
http://www.cs.wisc.edu/condor/manual/v7.7/3_3Configuration.html#param:SystemPeriodicHoldReason

File descriptors and process IDs

- Each Shadow keeps open O(10) FDs
  - And we have a shadow per running job
  - Make sure you have enough system-wide file descriptors
- Make also sure the system can support all the shadow PIDs

```
~$ echo 4849118 > /proc/sys/fs/file-max
~$ cat /proc/sys/fs/file-nr
59160  0   4849118
~$ echo 128000 > /proc/sys/kernel/pid_max
```

http://www.cs.wisc.edu/condor/condorg/linux_scalability.html

Firewall tunning

- If you decided you want to keep the firewall you may need to tune it
  - At minimum, open the port of the Shared Port daemon
- You may also need to tune the firewall scalability knobs

Statefull firewalls will have to track 2 connections per running job

```
# for iptables
sysctl -w net.ipv4.netfilter.ip_conntrack_max=385552;
sysctl -p;
echo 48194 > /sys/module/ip_conntrack/parameters/hashsize
```

The End

Pointers

- The official project Web page is http://tinyurl.com/glideinWMS
- glideinWMS development team is reachable at glideinwms-support@fnal.gov
- Condor Home Page http://www.cs.wisc.edu/condor/
- Condor support condor-user@cs.wisc.edu condor-admin@cs.wisc.edu

Acknowledgments

- The glideinWMS is a CMS-led project developed mostly at FNAL, with contributions from UCSD and ISI
- The glideinWMS factory operations at UCSD is sponsored by OSG
- The funding comes from NSF, DOE and the UC system