[Grid Engine Config Tips](http://idolinux.blogspot.com/2011/12/grid-engine-config-tips.html)

Here are some of the Grid Engine configuration steps we should take on a new install. I recommend doing all of these from the very beginning, to prevent changes that may confuse or break user workflow.

There is one thing we must always do with a new compute cluster, and that is enable hard memory limits. Users are usually not too keen on any kind of limit, because jobs will eventually run into them. Once the realization is made that limits ensures node stability and uptime, users will demand them. Without limits, one bad job can crash a node and bring down many other jobs.

To enable hard memory limits, we modify the complex configuration to make h\_vmem requestable.

# qconf -mc

h\_vmem h\_vmem MEMORY <= YES YES 1g 0

Once this complex is set, it is a good idea to define a default option for qsub in the*$SGE\_ROOT/default/common/sge\_request* file. When enabling h\_vmem, we should also set a default value for h\_stack. h\_vmem sets a limit on virtual memory, while h\_stack sets a limit on stack space for binary execution. Without a sufficient value for h\_stack, programs like Python, Matlab or IDL will fail to start. Here, we are also binding each job to a single core.

-binding linear:1

-q all.q

-l h\_vmem=1g

-l h\_stack=128m

If we want to manually set values for each individual node, like slots and memory, a for-loop is very helpful.

# qconf -rattr exechost complex\_values slots=8,num\_proc=8,h\_vmem=8g node01

# for ((I=1; I <= 16 ; I++)); do

> NODE=`printf "node%02d\n" $I`

> MEM=`ssh $NODE 'free -b |grep Mem |cut -d" " -f 5'`

> SWAP=`ssh $NODE 'free -b |grep Swap |cut -d" " -f 4'`

> VMEM=`echo $MEM+$SWAP|bc`

> qconf -rattr exechost complex\_values slots=8,num\_proc=8,h\_vmem=$VMEM $NODE

> done

To submit a job with a 4 gig limit, use the -l command line option.

$ qsub -l h\_vmem=4g -l h\_stack=256m myjob.sh

To see available memory, use qstat.

$ qstat -F h\_vmem

It is also a good idea to place limits on the amount of memory any single process on the login node may allocate, in the */etc/security/limits.conf* file. This example will limit any user in the *clusterusers* group to 4 gigs per process. Anything larger should be ran via qlogin. When adding new users, make sure to add them to this now default group.

# limit any process to 4GB = 1024\*1024\*4KB = 4194304

@clusterusers hard rss 4194304

@clusterusers hard as 4194304

There should also be a limit on how many jobs a single user can queue at once. If a user must submit over 2000 jobs simultaneously, they may want to consider a more manageable workflow utilizing array jobs.

# qconf -mconf

max\_u\_jobs 2000

If we want to limit the number of jobs a single user can have in the running state simultaneously:

# qconf -msconf

max\_reservation 128

maxujobs 128

If the queue will be accepting multi-slot parallel jobs, slot reservation should be enabled to prevent starvation. Otherwise, single-slot jobs will constantly fill in space ahead of the big job. This can be done by submitting multi-slot jobs with the “-R y" option.

To enable a simple fairshare policy between all users, there are only three options to check:

# qconf -mconf

enforce\_user auto

auto\_user\_fshare 100

# qconf -msconf

weight\_tickets\_functional 10000

To be a bit more verbose, we should collect some job scheduler info.

# man sched\_conf

# qconf -msconf

schedd\_job\_info true

Now we can see why or why not a job is scheduled.

$ qstat -j 427997

$ qacct -j 427997

If we plan to allow graphical GUI programs in the queue, we must setup a qlogin wrapper script with proper X11 forwarding.

# vim /usr/global/sge/qlogin\_wrapper

# chmod +x /usr/global/sge/qlogin\_wrapper

qlogin\_wrapper:

#!/bin/sh

HOST=$1

PORT=$2

shift

shift

echo /usr/bin/ssh -Y -p $PORT $HOST

/usr/bin/ssh -Y -p $PORT $HOST

Set the qlogin wrapper and ssh shell:

# qconf -mconf

qlogin\_command /usr/global/sge/qlogin\_wrapper

qlogin\_daemon /usr/sbin/sshd -i

If we have a floating license server with a limited number of seats, we will want to configure a consumable complex resource. When a user submits a job, the qsub option '-l idl=1' must be used. In this example, the number of jobs that specify idl will be limited to 15 at any one time.

# qconf -mc

matlab ml INT <= YES YES 0 0

idl idl INT <= YES YES 0 0

# qconf -me global

complex\_values matlab=10,idl=15

If we want to have multiple queues across the same hosts, we can define a policy so that nodes do not become oversubscribed.

# qconf -arqs

{

name limit\_slots\_to\_cores\_rqs

description Prevents core oversubscription across queues.

enabled TRUE

limit hosts {\*} to slots=$num\_proc

}

Here are some handy aliases I find useful in my ~/.bashrc file:

alias qconf-all='qconf -mq all.q'

alias qconf-offline='echo "all.q@nodeXX" ; \qconf -rattr queue slots 0' # all.q@node23

alias qconf-online='echo "all.q@nodeXX" ; \qconf -rattr queue slots 8' # all.q@node23

alias qstat-errors='\qstat -f -explain E'

alias qstat-summary='\qstat -g c'

alias qstat-mem='qstat -F h\_vmem'

alias qstat-ext='qstat -ext'

alias qstat-io='qstat -ext | awk '"'"'{print $11 " " $5 " " $1}'"'"' | grep -v "\-\-" | sort -n'

alias qmod-clear='\qmod -c "\*"'

And done. If you have some must-do configuration steps, please post them in the comments bellow.