Grid Engine for users

Usage & productivity focus
Learning curve ahead...

- Grid Engine has a small number of command-line programs
- Most are very powerful and can be invoked in bewilderingly complex ways
  - Resource requests
  - Job arrays & dependencies
- Man pages and wikis.sun.com will be essential as you get up to speed
Exercise 01

- Goals
  - See Grid Engine in action
  - Run a few commands
Exercise 02

Goals

- Submitting, monitoring and naming a few batch jobs
- The most basic job script ('sleeper.sh')
Exercise 03 - qrsh

Goals

- Run real jobs
- “Instant” Execution via ‘qrsh’
qrsh/sqrsh - Reminder

- The Grid Engine 'qrsh' program will run your command or job ASAP on the least loaded node in the system. This is a quick and lightweight way to run short jobs or even test grid engine functionality.

- Note that 'qrsh' commands will fail on clusters where there are no free job slots. These and other error conditions need to be checked for whenever 'qrsh' is used as part of a script or automated workflow.

- Use with some caution - be careful of big or resource intensive jobs
Exercise 04

Goals

- First intro to job dependencies
- Trivial chaining
- Simple binary wrapping for trivial SGE integration
- Synchronous job submission
Summarizing basic usage
Summarizing basic usage
Most useful SGE commands

- qsub / qdel
  - Submit jobs & delete jobs
- qstat & qhost
  - Status info for queues, hosts and jobs
- qacct
  - Summary info and reports on completed job
- qrsh
  - Get an interactive shell on a cluster node
  - Quickly run a command on a remote host
- qmon
  - Launch the X11 GUI interface
SGE Commands: ‘qstat’

```
cat:~ administrator$ qstat

<table>
<thead>
<tr>
<th>job-ID</th>
<th>prior</th>
<th>name</th>
<th>user</th>
<th>state</th>
<th>submit/start at</th>
<th>queue</th>
<th>slots</th>
<th>ja-task-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 10:13:34</td>
<td>all.q@node002</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 10:04:01</td>
<td>all.q@node005</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.56000</td>
<td>hs</td>
<td>sga6043</td>
<td>r</td>
<td>12/07/2004 12:14:51</td>
<td>all.q@node007</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.56000</td>
<td>gr.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 09:47:40</td>
<td>all.q@node009</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 10:01:01</td>
<td>all.q@node010</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 10:11:49</td>
<td>all.q@node015</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m010798</td>
<td>r</td>
<td>12/07/2004 10:08:18</td>
<td>all.q@node016</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```
SGE Commands: ‘qstat -f’

cat:~ administrator$ qstat -f

<table>
<thead>
<tr>
<th>queue name</th>
<th>qtype</th>
<th>used/tot.</th>
<th>load_avg</th>
<th>arch</th>
<th>states</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:all.q@cat.tamu.edu">all.q@cat.tamu.edu</a></td>
<td>BIP</td>
<td>0/2</td>
<td>0.02</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:all.q@node001.cluster.private">all.q@node001.cluster.private</a></td>
<td>BIP</td>
<td>0/2</td>
<td>0.03</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:all.q@node002.cluster.private">all.q@node002.cluster.private</a></td>
<td>BIP</td>
<td>1/2</td>
<td>1.00</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 0.56000 first.sh m010798</td>
<td>r</td>
<td>12/07/2004 10:13:34</td>
<td>1</td>
</tr>
<tr>
<td><a href="mailto:all.q@node003.cluster.private">all.q@node003.cluster.private</a></td>
<td>BIP</td>
<td>0/2</td>
<td>0.00</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:all.q@node004.cluster.private">all.q@node004.cluster.private</a></td>
<td>BIP</td>
<td>0/2</td>
<td>0.01</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:all.q@node005.cluster.private">all.q@node005.cluster.private</a></td>
<td>BIP</td>
<td>1/2</td>
<td>1.00</td>
<td>darwin</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 0.56000 first.sh m010798</td>
<td>r</td>
<td>12/07/2004 10:04:01</td>
<td>1</td>
</tr>
</tbody>
</table>
SGE Commands: ‘qhost’

cat:~ administrator$ qhost

<table>
<thead>
<tr>
<th>HOSTNAME</th>
<th>ARCH</th>
<th>NCPU</th>
<th>LOAD</th>
<th>MEMTOT</th>
<th>MEMUSE</th>
<th>SWAPTO</th>
<th>SWAPUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cat</td>
<td>darwin</td>
<td>2</td>
<td>0.03</td>
<td>2.0G</td>
<td>1.4G</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node001</td>
<td>darwin</td>
<td>2</td>
<td>0.02</td>
<td>2.0G</td>
<td>227.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node002</td>
<td>darwin</td>
<td>2</td>
<td>1.00</td>
<td>2.0G</td>
<td>274.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node003</td>
<td>darwin</td>
<td>2</td>
<td>0.00</td>
<td>2.0G</td>
<td>273.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node004</td>
<td>darwin</td>
<td>2</td>
<td>0.02</td>
<td>2.0G</td>
<td>275.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node005</td>
<td>darwin</td>
<td>2</td>
<td>1.00</td>
<td>2.0G</td>
<td>274.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node006</td>
<td>darwin</td>
<td>2</td>
<td>0.02</td>
<td>2.0G</td>
<td>274.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node007</td>
<td>darwin</td>
<td>2</td>
<td>1.00</td>
<td>2.0G</td>
<td>326.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node008</td>
<td>darwin</td>
<td>2</td>
<td>0.00</td>
<td>2.0G</td>
<td>271.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node009</td>
<td>darwin</td>
<td>2</td>
<td>1.00</td>
<td>2.0G</td>
<td>274.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>node010</td>
<td>darwin</td>
<td>2</td>
<td>1.05</td>
<td>2.0G</td>
<td>275.0M</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
SGE Commands: qsub

- Used to submit job scripts to Grid Engine
  - Usage: qsub [options] [scriptfile] [script args]
- qsub finally accepts binaries
  - qsub -b y /bin/hostname
- Powerful
  - `man qsub` is your friend
  - qsub usage can be as simple or as complicated as you need

- Example options
  - -A account_string; used to group accounting info
  - -hold_jid job_id; holds submitted job on job_id
  - -l resource=value; requests a specific resource
  - -t n[-m[:s]]; array job
  - -v, -V; export some or all of your ENV variables
qsub

General format:

$ qsub <qsub options> program <prog. options>

The simplest possible SGE submit syntax would be of this form:

$ qsub ./myjob.sh
Example: sleeper.sh

#!/bin/sh
#
# Usage: sleeper.sh [time]
# default for time is 60 seconds

# -- our name ---
#$ -N Sleeper
#$ -S /bin/sh

/bin/echo I am running on host `hostname`.
/bin/echo Sleeping now at: `date`

time=60
if [ $# -ge 1 ]; then
time=$1
fi
sleep $time

echo Now it is: `date`
#! /bin/sh
#
# Usage: sleeper.sh [time]
# default for time is 60 seconds

# -- SGE ARGUMENTS --
#$ -N Sleeper
#$ -S /bin/sh

/bin/echo I am running on host `hostname`.
/bin/echo Sleeping now at: `date`

time=60
if [ $# -ge 1 ]; then
time=$1
fi
sleep $time

echo Now it is: `date`
Real world example

#!/bin/sh

# Batch-submission script for SGE (Sun GridEngine) system

# Do we need to re-source our grid engine environment?
source /common/sge/default/common/settings.sh

## -- Chris Dagdigian; BioTeam Inc.
## -- Embedded grid engine directives follow
#$ -N %NAME%
#$ -o %DIR%/.%JOBID%.qlog.out
#$ -e %DIR%/.%JOBID%.qlog.err
#$ -P glide
#$ -hard -l glideL-impact-main=1
#$ -hard -l glideL-impact-glide=4

## -- ok back to work (Glide stuff below) ...
More useful ‘qsub’ arguments

- All of these can be embedded in scripts, passed via the command-line or passed via the GUI job submission tool

- `-A [string]`
  - Pass a string that will end up in accounting log. Useful for post processing or grouping jobs for grouping and reporting

- `-m b` or `-m e`
  - Mail submitter when job begins/ends

- `-m a` or `-m s`
  - Mail submitter when aborted or suspended

- `-m n`
  - Override all other mail options; Don’t send email for any reason
Jobs: Binaries vs. Scripts

- SGE 6 at the CLI assumes scripts
  - “qsub -b y ...” to override
- SGE 6 DRMAA assumes binaries
- 2 main differences in handling
  - For scripts, SGE transfers entire file
  - For binaries, SGE just sends the path
Using Resources

- Resources can be collected together using arithmetic and Boolean operators to form very complex resource requirement strings.

- qsub -hard -l \ arch=solaris64,h_mem_free=800M,swap_free=50M ./myJob.sh
  - Job must run on a 64 bit Solaris box with at least 800 MB of free memory and 50 MB of available swap space

- Remember:
  - You can embed these requests in your scripts so they don’t have to be typed all the time
  - Can also define “default request” files on a per-user or global level
Default / Preference Files

- Default (qsub) submission settings
  - `$SGE_ROOT/$SGE_CELL/common/sge_request`
  - `$HOME/.sge_request`
  - `$PWD/.sge_request`

- Default (qstat) monitoring settings
  - `$SGE_ROOT/$SGE_CELL/common/sge_qstat`
  - `$HOME/.sge_qstat`

- Overridden by runtime arguments
  - Explicit: “qsub -clear … “
Monitoring our job with ‘qstat’

cat:- administrator$ qstat

<table>
<thead>
<tr>
<th>job-ID</th>
<th>prior</th>
<th>name</th>
<th>user</th>
<th>state</th>
<th>submit/start at</th>
<th>queue</th>
<th>slots</th>
<th>ja-task-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:13:34</td>
<td>all.q@node002</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:04:01</td>
<td>all.q@node005</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.56000</td>
<td>hs</td>
<td>sga6043</td>
<td>r</td>
<td>12/07/2004 12:14:51</td>
<td>all.q@node007</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.56000</td>
<td>gr.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 09:47:40</td>
<td>all.q@node009</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:01:01</td>
<td>all.q@node010</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:11:49</td>
<td>all.q@node015</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.56000</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:08:18</td>
<td>all.q@node016</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Accounting data with ‘qacct’

cat:~/sge-test administrator$ qacct -j 30

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>qname</td>
<td>all.q</td>
</tr>
<tr>
<td>hostname</td>
<td>node003.cluster.private</td>
</tr>
<tr>
<td>group</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>owner</td>
<td>administrator</td>
</tr>
<tr>
<td>project</td>
<td>NONE</td>
</tr>
<tr>
<td>department</td>
<td>defaultdepartment</td>
</tr>
<tr>
<td>jobname</td>
<td>hostname</td>
</tr>
<tr>
<td>jobnumber</td>
<td>30</td>
</tr>
<tr>
<td>taskid</td>
<td>undefined</td>
</tr>
<tr>
<td>account</td>
<td>sge</td>
</tr>
<tr>
<td>priority</td>
<td>0</td>
</tr>
<tr>
<td>qsub_time</td>
<td>Wed Dec 8 09:33:20 2004</td>
</tr>
<tr>
<td>start_time</td>
<td>Wed Dec 8 09:42:05 2004</td>
</tr>
<tr>
<td>end_time</td>
<td>Wed Dec 8 09:42:05 2004</td>
</tr>
<tr>
<td>granted_pe</td>
<td>NONE</td>
</tr>
<tr>
<td>slots</td>
<td>1</td>
</tr>
<tr>
<td>failed</td>
<td>0</td>
</tr>
<tr>
<td>exit_status</td>
<td>0</td>
</tr>
<tr>
<td>ru_wallclock</td>
<td>0</td>
</tr>
</tbody>
</table>

chris@bioteam.net
Via the ‘qmon’ GUI
Submitting via the GUI
Submitting jobs

- Jobs are submitted via the ‘qsub’ command
- Many factors affect how/when a job gets dispatched for execution
  - Job resource requirements
  - Availability of eligible execution hosts
  - Various job slot limits
  - Job dependency conditions
  - Fairshare or priority constraints
  - Load conditions
Submitting Jobs

- Important to note that jobs are not necessarily dispatched in the order received
Checking running or pending jobs

We use ‘qstat’

cat:/home/chris$ qsub simple.sh
Your job 31 ("simple.sh") has been submitted.

cat:/home/chris$ qstat

<table>
<thead>
<tr>
<th>job-ID</th>
<th>prior</th>
<th>name</th>
<th>user</th>
<th>state</th>
<th>submit/start at</th>
<th>queue</th>
<th>slots</th>
<th>ja-task-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.56</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:13:34</td>
<td>all.q@node002</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.56</td>
<td>first.sh</td>
<td>m0l0798</td>
<td>r</td>
<td>12/07/2004 10:04:01</td>
<td>all.q@node005</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.56</td>
<td>hs</td>
<td>sga6043</td>
<td>r</td>
<td>12/07/2004 12:14:51</td>
<td>all.q@node007</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0.00</td>
<td>simple.sh</td>
<td>adminstr</td>
<td>qw</td>
<td>12/08/2004 10:01:24</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Checking completed jobs

```bash
cat:~/sge-test administrator$ qacct -j 31
==================================================
qname        all.q
hostname      node006.cluster.private
group         UNKNOWN
owner         administrator
project       NONE
department    defaultdepartment
jobname       simple.sh
jobnumber     30
taskid        undefined
account       sge
priority      0
qsub_time     Wed Dec  8 09:33:20 2004
start_time    Wed Dec  8 09:42:05 2004
end_time      Wed Dec  8 09:42:05 2004
granted_pe    NONE
slots          1
failed         0
exit_status    0
ru_wallclock   0
```
Job Status Checking: Summary

- For running or pending jobs:
  - Use the ‘qstat’ command

- For completed jobs:
  - Use the ‘qacct’ command
About System & Cluster Status

- `qstat -f`
  - Look for queues in alarm (‘a’) or (‘au’) state
  - Look for load averages of 99.99 percent
- `qhost`
Pending Jobs

A Jobs initial state when it is submitted to SGE is PENDING.

- Reported by ‘qstat’ as state (‘qw’)
  - (queued waiting)

Reasons for job remaining in a pending state.

- No free job slots
- All queues have hit suspend or load thresholds
- You have requested an impossible resource
qstat simple usage

- `qstat -help`
  - More usage info
- `qstat`
  - Displays current jobs in the system
- `qstat -j [job ID or joblist]`
  - Shows config and scheduler info for job
- `qstat -l [resource string]`
  - Shows jobs/queues that provide/need the resource
- `qstat -u <user>`
  - Show only jobs from that user
- `qstat -t`
  - Information on array jobs
qstat simple usage continued

- `qstat -q [queue]`
  - Show jobs running in queue

- `qstat -explain`
  - More info about the reason queue(s) in alarm state

- `qstat -f`
  - Full queue summary

- `qstat -f -ne`
  - Queue summary with empty queues ignored
Possible job states reported by `qstat`

- ‘t’ -- Transferring
- ‘r’ -- Running
- ‘R’ -- Restarted
- ‘s’ -- Suspended
- ‘S’ -- Suspended by the queue
- ‘T’ -- Suspend queue threshold reached
- ‘w’ -- Waiting
- ‘h’ -- Hold
- ‘e’ -- Error
Possible queue states reported by qstat

- ‘u’ -- Unknown (sge_execd or server down?)
- ‘a’ -- Alarm (load threshold reached)
- ‘A’ -- Alarm (suspend threshold reached)
- ‘s’ -- Suspended (by user or admin)
- ‘d’ -- Disabled (by user or admin)
- ‘C’ -- Suspended (by calendar)
- ‘D’ -- Disabled (by calendar)
- ‘S’ -- Suspended (by subordination)
- ‘E’ -- Error (sge_execd can’t reach shepherd)
Demo Time (#5, #6)
Array Jobs
Simple Workflow
05 - Array Jobs Demo

Goal

- See Array Jobs in action
- Understand them (!)
  - A huge benefit for some types of workflows
Goal
- See a simple workflow script that uses arrays and job dependencies to perform a powerful multi-step task

Contrived Use case:
- A 10 element array job representing real scientific "work"
- A post processing job that is dependent on completion of the "work"
- A cleanup script that is dependent on the postprocessing step
Questions?
Debugging SGE problems
Debugging SGE Problems

When you:
- Can’t run SGE commands
  - Command not found
  - System not responding
  - Remote operation permission denied

Try:
- qhost and ‘qstat -f’
Debugging SGE Problems (cont.)

- Job level problems
  - Run:
    - qsub -w v <full job request>
  - This will tell you if the job can run if
    - All slots on all queues were empty
    - All load values were ignored
  - Good source of info on ‘why can’t my job be scheduled’ problems
Debugging SGE Problems (cont.)

- Job level problems with pending jobs
  - Run:
    - `qstat -j <job_id>`
  - This will tell you why the job is pending and if there are any reasons why queues cannot accept the job
Debugging SGE Problems (cont.)

- Many times the problems are not SGE related
  - Permission, path or ENV problems

- Best thing to do is watch your STDERR and STDOUT
  - Use the qsub ‘-e’ and ‘-o’ switches to send output to a file that you can read
  - Use qsub ‘-eo’ to send STDOUT and STDERR to the same file (useful for debugging)
Debugging SGE Problems (cont.)

- To get email listing why a job aborted
  - Use: ‘qsub -m a user@host [rest of command] ’
Debugging SGE Problems (cont.)

- Checking exit status and seeing if jobs ran to completion without error
  - Use: ‘qacct -j <job_id>’ to query the accounting data
  - Will also tell you if the job had to be requeued onto a different queue or exechost
Final word on debugging …

- SGE Admins have many more tools
  - Scheduler trace/profile/monitoring
  - Jobdir “keep_active=true”
  - SGE debug ENV variables

- Tip:
  - Ask for help if you get stuck
More Grid Engine Usage
Jobs pending on resources...

```
[sgeadmin@portal examples]$ qstat

<table>
<thead>
<tr>
<th>job-ID</th>
<th>prior</th>
<th>name</th>
<th>user</th>
<th>state</th>
<th>submit/start at</th>
<th>queue</th>
<th>master</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa1.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>71</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa10.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>67</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa2.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>73</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa3.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa4.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>69</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa5.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>66</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa6.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>68</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:38</td>
<td>cfa7.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>65</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:22</td>
<td>cfa8.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>r</td>
<td>12/20/2002 01:00:22</td>
<td>cfa9.q</td>
<td>MASTER</td>
</tr>
<tr>
<td>74</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>0</td>
<td>Sleeper</td>
<td>sgeadmin</td>
<td>qw</td>
<td>12/20/2002 01:00:31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Jobs pending on resources...

```bash
chrisdag:tmp dag$ qstat -j 46
===============================================
job_number: 46
exec_file: job_scripts/46
submission_time: Wed Mar 26 10:03:33 2008
owner: dag
uid: 501
group: dag
gid: 501
sgo_o_home: /Users/dag
sgo_o_log_name: dag
gso_o_path: /opt/sge/bin/darwin-
x86:/usr/local/bin:/usr/local/sbin:/opt/bin:/opt/sbin:/opt/mysql/bin:\
/sw/bin:/sw/sbin:/usr/bin:/bin:/usr/sbin:
sgo_o_shell: /bin/bash
sgo_o_workdir: /private/tmp
sgo_o_host: chrisdag-aliased
account: sge
cwd: /private/tmp
path_aliases: /tmp_mnt/ * * /
hard_resource_list: arch=solaris64
mail_list: dag@chrisdag-aliased
notify: FALSE
job_name: Sleeper
jobshare: 0
shell_list: /bin/sh
env_list: 
script_file: ./sleeper.sh
 scheduling info: queue instance "test.q@chrisdag-aliased" dropped because it is disabled
 (-l arch=solaris64) cannot run at host "chrisdag-aliased" because it offers only
hl:arch=darwin-x86
```
Using Resources

- Resources can be collected together using arithmetic and Boolean operators to form very complex resource requirement strings.

```bash
qsub -hard -l \ arch=solaris64,h_mem_free=800M,swap_free=50M ./myJob.sh
```
- Job must run on a 64 bit Solaris box with at least 800 MB of free memory and 50 MB of available swap space

Remember:
- You can embed these requests in your scripts so they don’t have to be typed all the time
- Can also define “default request” files on a per-user or global level
Available default resources

Serial Job
Hard Resources

Available Resources

Soft Resources

Available default resources

Serial Job
Hard Resources

Available Resources

Soft Resources
Queue instance resources

cat:~:/sge-test administrator$ qconf -se node001.cluster.private
hostname node001.cluster.private
load_scaling NONE
complex_values NONE
load_values np_load_long=0.000000,load_short=0.006836, \
load_medium=0.019043,load_long=0.000000,arch=darwin, \
num_proc=2,mem_free=1820.000000M,swap_free=0.000000M, \
virtual_free=1820.000000M,mem_total=2048.000000M, \
swap_total=0.000000M,virtual_total=2048.000000M, \
mem_used=228.000000M,swap_used=0.000000M, \
virtual_used=228.000000M,cpu=0.600000, \
np_load_avg=0.009522,np_load_short=0.003418, \
np_load_medium=0.009522,load_avg=0.019043
processors 2
user_lists NONE
xuser_lists NONE
projects NONE
xprojects NONE
usage_scaling NONE
report_variables NONE
Resources you may care about

- Example: MatLab licenses
- Handled via Grid Engine “System Complex” or “Load Sensor” mechanisms
- A FlexLM license is a special type of “user requestable, consumable resource”
Example: Licensed MatLab Jobs

- You must request the MatLab “resource”
- Assume cluster currently has 3 floating licenses:
- Usage would be:
  - qsub -hard -l matlab=1 ./matlab-script.sh
  - Or embedded inside a script:
  - #$ -hard -l matlab=1
Exercise: Array Job Example

- Array Jobs are extremely powerful
- Very efficiently handle the problem:
  - “how do I run application X many, many times with only minor changes in the command line arguments?
Exercise: Array Job example

- Why this matters
  - Grid Engine can probably handle a few tens of thousands of standalone jobs at any one time.
  - Grid Engine 6 has a design goal of handling 500,000 element job arrays
Exercise: Array Job example

- Experiment with the array job example script and input data
Lab Time (07_greedyJobs)

"How do I guarantee my job will get sole access to a compute node so it does not have to compete with another running job for resources?"
A few words on Resource Quotas
Resource Quotas

- The main enhancement to SGE 6.1
- Will likely have a significant impact
- Solves multiple issues that have been bothering SGE admins for years:
  - max_u_jobs on a per-host basis
  - Max jobs per user on a per-queue basis
  - Per user slot limits on parallel environments
Why quotas matter to users

- **Good & Bad**
  - Just another way for management to slow you down right?
  - Well …
    - Much potential for serious good
    - Very flexible and powerful capabilities
    - Removes the need for nasty hacks and global limits that SGE admins have had to invent over time
Why quotas matter to users

Key message

- Another subsystem you should be aware of
  - Like tickets & policies
- … so you know what is going on with your jobs and workflow
- … and so you can better communicate with the admins regarding your needs
Resource Quotas

- Syntax similar to firewall rules
- Simple Example
  - “limit slot access to user1 and user2 on every host in the @LinuxHosts hostgroup (except for host penguin03)”

```json
{
    name: example_resource_quota_set,
    enabled: true,
    limit: {
        users: ['user1', 'user2'],
        hosts: ['@LinuxHosts', '!penguin03'],
        slots: 1
    }
}
```
Resource Quotas

- Syntax
  - Multiple rule sets contain one or more rules
  - First matching rule from each set wins
  - Strictest rule set wins
- Rules can contain
  - Wildcard (*)
  - Logical not operator (!)
  - Brackets ({}): Means “treat this rule as per-member” instead of as a group
Quota Command Line

For Admins
- qconf -[AaMmds]rqs
  - The usual “Add, modify, delete, show” arg modifiers apply
- Wizard methods work
  - qconf -mattr resource_quota enabled false rule_1

For Users & Admins
- New binary “qquota” in 6.1
  - Also honors a “.sge_qquota” preference file
  - $SGE_ROOT/$CELL/common/sge_qquota
  - $HOME/.sge_qquota
Resource Quota Example 1

“The total number of running jobs from project “killerApp” should not exceed 40”

```json
{
    name: project_limit,
    description: "Throttle killerApp projects to 40 concurrent",
    enabled: true,
    limit: "project killerApp to slots=40"
}
```
Resource Quota Example 2

- “No power user should have more than 10 running jobs”

```json
{
    name: power_limit,
    description: Limit all power users
    enabled: true
    limit: users {@power} to slots=10
}
```
Resource Quota Example 3

- “Total number of running jobs from power users should not exceed 40, everyone else is limited to max 5 running jobs each”

```json
{
 name: "power_limit",
 description: "Limit all power users",
 enabled: true,
 limit: {
 users: @power to slots=40,
 users: * to slots=5
 }
}
```
“The total number of jobs without projects must not exceed 10”

```json
{
    name: "nonproject_limit",
    description: "Limit jobs without project affiliation",
    enabled: true,
    limit: "projects !* to slots=10"
}
```
Quota checking for users

- New program ‘qquota’
- Man page has best usage
- By default:
  - Shows you all in-play quota rules that apply to the calling user
END;

- Thanks!