SGE 6.0 configuration guide, version 1.1

Juha Jäykkä
juolja@utu.fi

Department of Physics
Laboratory of Theoretical Physics
University of Turku

18.03.2005

First, some notes

- This needs to be revised to include GRID queues.
- Also, some LAM/MPICH issues need to be discussed and decided upon.

Installing SGE 6.0u3

- As per O-P's instructions. When we got the SGE 6 packages and O-P's installation instructions, I went on to install it on topaasi.
- With some real-time assistance from O-P, all went exactly according to his instructions. What you might want to check is that the nodes really have yum installed before you start. We had two nodes, installed in December and January, that did not have. There had been some rpm's missing. O-P fixed that in topaasi and I think in all other clusters, too, but please check. (Nothing will crash if you don't - those nodes just won't have SGE 6 on them.)

Why?

Because backfilling - the whole reason to upgrade to 6.0 in the first place - is broken in u3.

How?

a) grab from CVS the version with V60_u4_TAG or download the source, compile and install

b) wait until CSC packages it, then install as per CSC's instructions

Upgrading continued, method a)

- compile as per Sun's instructions
- stop SGE processes on front end
  - jobs need to be stopped/suspended/killed - they will run happily without the sge queue master
- install as per Sun's instructions
- do not restart sge yet
- tar up /opt/gridengine, copy tar to nodes
- make note of sge processes running on nodes, kill them
- untar the tarball, restart sge processes on nodes and front end

Upgrade notes

- Note that upgrading and restarting SGE6u3 to u4 keeps the queues intact, if done correctly. I cannot guarantee that my method is correct, so please consult Sun's docs first. The above method a is what I did with topaasi and it worked there.
- Note also that method a does not survive a node reinstall, since reinstall will install CSC's u3 version until CSC updates it to u4.

I will skip configuring u3, since it is the same except that you get no advantage of it.

Configuring SGE 6.0u4

- default config is a FIFO, except multi-cpu jobs are overridden by serial jobs
- proper configuration requires some background info
- grain of salt - I may not understand it all

SGE 5 vs. 6

- queues have been renamed to queue instances (still on-node)
- a new beast called cluster queue, which is what we've been waiting for
- cluster queue consists of queue instances and spans the whole cluster

CSC default install

- one cluster queue, called all.q
- serial jobs override parallel ones $\Rightarrow$ we need to do something
  - either split the cluster in two: parallel nodes and serial node
  - will cause lots of idle time
  - not useful for small sites
- another solution uses a thing called backfilling

Resource reservation

- jobs can request RR to reserve their resources even when they are still in the queue
- gives them chance to run even with serial jobs in the queue
- RR will happily let all CPUs idle waiting for a really big job
- to reduce number of idle CPUs, backfilling is used
- nicely, backfilling and RR are both turned off by default

Configuring RR and backfilling

- all.q reprioritises the jobs in the queue every 15 seconds
- qstat shows this priorised order
- after sorting, schedd tries to dispatch the highest priority job
- if its dispatched, the priorisation starts again
- what if the highest priority job cannot be run?
  - by default, SGE goes downwards on the queue and runs the first job it can - overriding the priorities!
  - resource reservation and backfilling!!!

Resource reservation (CPU's, mem...)

- happens when the highest priority job cannot be run and
- this job requests RR and
- there are free resources

Suppose a parallel job at the top of the queue has triggered RR

- queue is resorted as if the RR job had been dispatched
- if, after reservation, there are free CPUs, a job is dispatched normally
- if there are no free CPU's, SGE tries to backfill

Backfill

- critcal point is h_rt resource
- SGE looks at currently running jobs, checks when they will be finished at the latest
- let these times be $T(1), \ldots, T(n)$
- let the highest priority job which fits in the RR'd CPUs have h_rt value such that, if dispatched now, it will finish at $T(n+1)$
- if $T(n+1) < \min\{T(1), \ldots, T(n)\}$, the job $n+1$ is dispatched
- rationale being, it does not hurt anyone

Urgency values

- priorities calculated from three values
- default (and topaasi's) setup makes two of these zero, only one called urgency is non-zero
- measures the resources the job needs
- higher values give higher priorities (backwards? no!)
- encourages users to reserve just the resources they really need

How to make this happen

- please read sge_priority man page
- this are topaasi's config; it works well

How to make this happen: qconf -msconf

1. “max_reservation” must be $>$ 0 (topaasi has 20)
2. useful to se “weight_waiting_time” above zero
   - enables any job to eventually go through - RR or not
   - topaasi has this $=$ 500
3. set “default_duration” to non-zero value; topaasi has 337 hours; is not enforces, but read on...
4. I set “notify_time” to 10 minutes, but not sure of its meaning Who knows?
5. A bug in SGE6 makes about 30% of parallel jobs to vanish; two workarounds exist
   1. set “reprioritize_interval” to “00:02:00” and “schedule_interval” to “00:01:00”
   2. install SGE 6.0u4, it contains another workaround
6. set “h_rt” for the queue; it’s enforced; topaasi has 337 hours - for a reason (qconf -mq all.q)

Now your queues are set up, but the users need to know what to do to use them. There are several things the user should know. The following are all command line parameters for qsub (they can also be specified in the job script).

-R y Parallel jobs need this in order to reserve resources. Otherwise they will not! You may restrict use of -R y to users or groups.

-l h_rt Maximum wall clock time a job may use. Used for backfilling and enforced by the scheduler by sending a SIGKILL to the job when this is reached. At least LAM jobs WILL NOT GET this signal, though (only mpirun gets it).

-l s_rt Soft wall clock limit; used to warn the job before it hits the h_rt value. Scheduler sends SIGUSR1 when this is reaced, but again I think LAM jobs won't get it (unless mpirun passes it to the processes). Also, it's useless to set this $\geq$ h_rt, but qsub does NOT check for this.

-l h_cpu Per cpu time limit; otherwise like h_rt, but please note that CPU time is always $\leq$ wall clock time. In an ideal situation they equal, but this never happens in practice.

-l s_cpu Per cpu soft time limit; scheduler tells the job that this has been reached by sending it SIGXCPU. Again, LAM...

-notify Needed to get the signals to the jobs - if I have understood correctly. It does no harm to specify this in any case.

Changing queue configuration: qconf -mq <queue_name>; changing scheduler configuration: qconf -msconf and changing complexes: qconf -mc. Replace -m by -s and you'll see current values (you'll see them with -m, too, but then you can also edit them; with -s you cannot).
Further reading regarding the SGE configuration is in queue_conf, sge_priority and sched_conf manpages. There is also complex.5 manpage, but you need to point man directly to it (or change your MANPATH) since there is another complex.5, which is earlier in MANPATH and decsribes the standard C library complex variables. :) SGE's complex.5 is at /opt/gridengine/man/man5/complex.5.