NQS : Queuing simulation jobs on the PCs

Julie McEnery

1 What is it for?

It was decided that since generating simulations is cpu intensive and does not require much disk I/O that they would be best run on the pc cluster. Ideally, we would like to have one job per pc running continuosly. NQS (Network Queuing System) was installed to make this easier. Instead of logging on to each machine checking to see how busy it is and starting a job if the machine is idle, one now simply submits a series of jobs to the queuing system and it takes care of the rest. The jobs are sent to a scheduler program which looks to see which pc is most idle and sends the job to that machine for execution. The system is currently set up to allow only one job to run per pc, if no machine is available, incoming requests sit waiting in the scheduling queue until a machine becomes free. Requests are sent out from the scheduling queue switching between users: for example, if I (Julie) submit 10 jobs to the queue, 4 of them will be passed to the pcs for immediate execution and 6 have to wait until a pc becomes available. So the scheduler queue (mc-sched) will have 6 entries.

julie5
julie6
julie7
julie8
julie9
julie10

If Stefan and Joe both now submit one job each, the order that the scheduler will send the jobs out to the destination machines for execution will be:

julie5
stefan1
joe1
julie6
julie7
julie8
julie9
julie10

So don't be discouraged if you see lots of jobs already in the queue, submit away and the system will try and allocate resources in a reasonably fair way.
Jobs submitted to the queue are run at a nice value of 19. Thus leaving plenty of resources free on the pcs for other (non-simulation) jobs.

2 How to submit a job?

Jobs are submitted to the queuing system using the qsub command. First you need to write a file (script) containing the commands you wish to have executed. For example, if I want to run a corsika job I could write a script called corsika.sh (the name of the script file is not important) containing the following lines:

```
cd /disk03/people/jmcenery/sims/prog/corsika_linux
corsika_561 < INPUTS
```

When NQS reads your script it does not know where it is, so the the first line of the script must be a cd to change into your working directory (the directory containing your programs). To submit this job to the queuing system, you need to log on to any pc and type:

```
qsub -q mc corsika.sh
```

This submits your job to the queue called mc (for Monte Carlo). From here your job is sent to the scheduler queue (mc-sched), which decides on which machine your job will be run and then sends your job to the queue mc-dest on that machine for execution. When your job is completed the standard output will be written to the file corsika.sh.o?? and the stardard error is written to corsika.sh.e??, where ?? is the id number associated with your request. These are the files to look at for more information when your job has completed. If your job seems to die unexpectedly, these files often provide some clues as to what went wrong.

3 You have submitted a job, now what?

Now that your job has been submitted it might be convenient to know if your job is running, where it is running, how to delete it, and how to view the standard output and error (the stuff that would usually go to your screen). The main command to check the status of your job is qstat. To find you the status of all jobs currently submitted in the queuing system one would type (from any pc):

```
qstat -ad
```

I have just typed this command on flossie and find that:

```
Request         I.D.   Owner    Queue                                    St
-------------- ------- -------- --------                                 --
script1            71  stefan   mc-dest                                  R
Destination machine: burst
Request         I.D.   Owner    Queue                                    St
-------------- ------- -------- --------                                 --
script3            72  stefan   mc-sched                                 R
script             84  atkins   mc-sched                                 R
script5            74  stefan   mc-sched                                 R
script4            76  stefan   mc-sched                                 R
script2            85  atkins   mc-sched                                 R
script3            86  atkins   mc-sched                                 Q
Destination machine: camila
```

```
Request         I.D.   Owner    Queue                                          St
-------------- ------- -------- --------                                       --
script2           68   stefan   mc-dest                                        R
Destination machine: wacko
Request         I.D.   Owner    Queue                                          St
-------------- ------- -------- --------                                       --
script6           75   stefan   mc-dest                                        R
Destination machine: thetick
Request         I.D.   Owner    Queue                                          St
-------------- ------- -------- --------                                       --
run1              45   jmcenery mc-dest                                        R
```

From this I can see that Stefan has three jobs running (on flossie, camila and wacko), I have one job running (on thetick), and that there are 6 jobs waiting in the scheduler (mc-sched) for a machine to become available. Suppose now I want to see the output from my job 'run1' which, in this case, happens to be running on thetick. I need to find my request id number by typing:

```
qstat -as @thetick
```

This results in:

```
mc-dest@thetick.lanl.gov;  type=BATCH;  [ENABLED, RUNNING];  PIPEONLY;  pri=16  lim=1
  0 exit;   1 run;   0 stage;   0 queued;   0 wait;   0 hold;   0 arrive;

         REQUEST NAME        REQUEST ID            USER  PRI    STATE     PGRP
    1:           run1       45.flossie.lanl.gov jmcenery  31  RUNNING    32142
mc-in@thetick.lanl.gov;  type=PIPE;  [ENABLED, INACTIVE];  PIPEONLY;  pri=16  lim=1
  0 depart;   0 route;   0 queued;   0 wait;   0 hold;   0 arrive;

mc@thetick.lanl.gov;  type=PIPE;  [ENABLED, INACTIVE];  pri=16  lim=1
  0 depart;   0 route;   0 queued;   0 wait;   0 hold;   0 arrive;
```

I can now print the contents of standard output or standard error to the screen by typing (respectively):

```
qcat -o 45.flossie.lanl.gov
qcat -e 45.flossie.lanl.gov
```

If I want to delete this job I simply type:

```
qdel -k 45.flossie.lanl.gov@thetick.lanl.gov
```

The -k switch is necessary when one wishes to kill a job that is currently running. The -k switch is unnecessary when deleting a job which has 'Q' (queuing) or 'W' (waiting) status. Occasionally, you will notice that jobs in the mc-sched queue on burst show status 'R' when displayed with the 'qstat -ad' command. These jobs are not in a running mode, (they are listed as routing) and can not be deleted in the usual way. If you need to delete a job that is in this state, you will have to get in touch with me to kill it for you.

4 Summary of useful commands

For your reference...

```
qsub -q mc <script> ...... Submit the commands listed in the file
                           <script> to the queue called mc.
qstat -ad           ...... List the status of all jobs in the nqs system
qstat -as @machine  ...... List complete info. for jobs on the specified
   machine
qdel -k requestid@machine. Kill the specified request running on machine
qcat -o requestid   ...... Print standard output of requestid to the screen
qcat -e requestid   ...... Print standard error of requestid to the screen
```

Of course, if you need further information about any of these commands you can always look at their man page.

5 Boring details, bugs etc

This little section is here to provide a little more information on how the system is set up, and how we work around the problems. On each machine, three queues are set up, called mc, mc-dest and mc-in, on burst (the machine that does the scheduling) an additional queue is set up called mc-sched. mc is a pipe queue that accepts requests from qsub and pipes them to the queue mc-sched on burst. mc-in accepts requests from mc-sched@burst and pipes them to mc-dest for execution. The mc-in queue is necessary for the load balancing to work. Jobs sit in the mc-sched queue on burst until a mc-dest queue on one of the pcs becomes free. This is all very nice, now for the problem. If all machines are busy, jobs sit in the scheduler queue waiting for a machine to become idle. After a job has been sitting in the scheduler queue for a certain period of time (about half an hour) its status changes from 'W' (waiting) to 'R' (routing), it then stays in mc-sched with status 'R' indefinitely regardless of whether machines have become availabe or not. This is a problem specific to nqs running on pentium IIs (pentium pros apparently have no problems).
In a little more detail, what is thought to be happening is the following: Scheduler tries to run a job, marks it as 'routing' and passes it to pipeclient to find a destination. Pipeclient tries to find a destination - none are available. Pipeclient should mark the request as requiring a reschedule. Either pipeclient fails to do this, or nqsdeamon fails to

notice. Pipeclient exits, but the job is still 'routing'. Next time scheduler looks at the job, it ignores the job because it is still 'routing'.
I am not able to fix (or even find) this problem. However, I have a brute force approach to getting around it. I altered the setup of the mc-sched queue on burst so that jobs were no longer piped to mc-dest@burst. A short script on burst shuts down the nqs daemon on burst, waits 100 seconds and then restarts the daemon. This has the effect of requeuing the jobs in the mc-sched queue with status 'Q' or 'W' so now they can be passed on to the destination machines for execution. This script is run every 30 minutes.
Obviously this situation is not ideal. Moving the scheduler to one of the SGIs would probably solve the problem (the machines in the queuing system need not all be the same type), however I have read of compiling difficulties for IRIX 6.5 so it may be quite a time consuming process to implement. So I have put it off in the hope that a bug fix for the linux machines becomes available soon.