The WTCHG Research Computing Core

Talk 4: Monitoring and troubleshooting

Robert Esnouf (robert@strubi.ox.ac.uk; robert@well.ox.ac.uk),
Jon Diprose (jon@well.ox.ac.uk) & Colin Freeman (cfreeman@well.ox.ac.uk)

Generic emails: support rescomp@well.ox.ac.uk
users rescomp-users@well.ox.ac.uk

A series of introductory talks…

A set of six talks of about 45 minutes each (plus questions)

- Talk 1: What is the ResComp Core?
  (Mon 23/1 10:00 Room B; Robert Esnouf)
- Talk 2: A basic introduction to Linux
  (Wed 25/1 10:00 Room B; Robert Esnouf)
- Talk 3: Submitting jobs to the cluster
  (Thu 26/1 10:00 Room B; Robert Esnouf)
- Talk 4: Monitoring and troubleshooting
  (Mon 30/1 11:00 Room B; Robert Esnouf)
- Talk 5: ResComp centrally-managed applications
  (Wed 1/2 10:00 Room B; Jon Diprose)
- Talk 6: Doing your own thing (compiling and customizing)
  (Thu 2/2 10:00 Room B; Jon Diprose)

The story so far…

The ResComp cluster has different types of node:

- 288 "A" cores @ 2.67GB/core; 640 "B" cores @ 8GB/core; 1720 "C" cores & 768 "D" cores @ 16GB/core; 144 "H" cores @ 42.66GB/core
- Login/submission nodes: rescomp[1-2] and the project servers

Files must be on the (chargeable) GPFS filesystems:

- /gpfs0: /users/<*group*>/<*user*>, /apps/well, /mgmt
- /gpfs1 & /gpfs2: /well/<*group*>

Refresher of basic Linux concepts

- User accounts, groups, commands, shells (bash)
- Environments and environment variables: $PATH and $LD_LIBRARY_PATH
- File streams (stdin, stdout and stderr) redirection & pipes ("<","<","|")
- Foreground and background jobs ("&"), "nohup" and "screen"
- Writing, executing and sourcing a shell script

The story so far…

Cluster computing is non-graphical and non-interactive!

- Sun Grid Engine (SGE) using a group-based share tree policy

Your job is to submit jobs, our job is to ensure that they start

- Don't wait for free slots, don't expect jobs to start immediately!

Submit scripts to the SGE scheduler using "qsub"

- Embed "qsub" arguments using "#$" lines
- Specify where stdout and stderr should go
- "#$ -pe shmem|mpi <*n*>" for multicore or MPI jobs
- Modify jobs with "qalter", "qrls", "qmod" & "qdel"
- Monitor jobs and SGE with "qstat", "qacct", "qhost" & "qconf"

Different queues: typically short (<24 hours) & long (<7 days)

- Jobs killed if they exceed time or memory limits
- Special queues for relion jobs and on project servers
- Use "qstat –j <*job_id*>" for details on queued and running jobs
- Use "qacct –j <*job_id*> [-t *task*]" for details of completed jobs

Overview of this talk…

Looking at "qstat" in more detail

The in-house monitoring commands

"qsum"

"qload"

A reminder of "qacct" for finished jobs

The ResComp Cluster workshop

Looking at "qstat" in more detail

```
-bash-4.1$ qstat -u hatton
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
1979389 0.20626 POBI_GATK_ hatton       r     01/21/2015 22:24:16 himem.qh@compH000                 16
1979390 0.00000 POBI_GATK_ hatton       hqw   01/21/2015 22:27:00                                   30
1979391 0.00000 POBI_GATK_ hatton       hqw   01/21/2015 22:28:50                                   30
-bash-4.1$
```

"qstat" is main tool for looking at queued and running jobs

- By default "qstat" just shows your jobs
- use "qstat –u <*user,user...*>" for other people
- use "qstat –u "*"" for everyone
- Add " -s rs" for running/suspended jobs, "-s p" for pending jobs

Queued jobs sorted by priority, fields are:

- $JOB_ID, priority, $NAME, $USER, state, submit or start time, execution queue instance, # slots, [$SGE_TASK_ID or task range]

Output is typically 2,000–20,000 lines long for all users!

"qstat" job state strings

```
-bash-4.1$ qstat -u hatton
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
1979389 0.20626 POBI_GATK_ hatton       r     01/21/2015 22:24:16 himem.qh@compH000                 16
1979390 0.00000 POBI_GATK_ hatton       hqw   01/21/2015 22:27:00                                   30
1979391 0.00000 POBI_GATK_ hatton       hqw   01/21/2015 22:28:50                                   30
-bash-4.1$
```

Running jobs

- r: running
- t: transitioning (in or out, usually in)
- dr: registered for deletion (e.g. a dead node; "qdel –f <*job_id*>")

Suspended jobs (no user action required)

- s: suspended by a superior queue (queue subordination)
- S: suspended by administrator
- T: threshold suspension

Queued jobs

- qw: queue wait
- hqw: held in queue wait
- Eqw: error state held in queue wait ("qmod –cj <*job_id*>")

Job suspension…

A suspended job is one that just sits there:

- It has an inactive process (and shepherd)
- It consumes resources such as memory
- It's wall time still ticks although no work is being done

Jobs suspended buy a superior queue (state "s")

- Only by ginn.qh@compH002 at the moment
- Not considered generally useful as memory is committed anyway
- Pushing a process memory into swap space is s..l..o..w..

Jobs suspended buy a an administrator (state "S")

- We may suspend either a job or a queue without warning
- Fairly rare event, usually across the whole cluster
- Main use is to reduce I/O load temporarily, or perhaps power load

Jobs suspended owing to threshold limits (state "T")

- The execd monitors ("uptime") load on each execution host
- If short-time load exceeds (#cores+2) no new jobs start
- If short-time load exceeds (#cores × 1.5) jobs suspended "at random"

Threshold suspension…

If short-time load exceeds (#cores × 1.5) jobs are suspended

- The suspended job (state "T") may not be the one causing the stress
- Ask for >1 slot with "#$ -pe shmem <n>" and $NSLOTS is set for you
- $NSLOTS is not set (to be 1) by default

Clusters are very different to workstations

- Workstations you often only use one core, rest are mostly idle
- Any parallelization for a workstation is usually good
- Clusters use schedulers to pack jobs onto all cores at all times
- Imperfect parallelization is often bad(!)
- For a job set the first job might finish earlier, last job will finish later

SGE will not play nicely with some modern parallelizations

- "pthreads" cause particular problems
- Fast context switching: great for high-latency asynchronous processes
- Classic example is network related activity, often just waiting for work
- Need to run of special queues without threshold suspension
- For mathematical computing benefits seem modest

… and now for something a bit simpler

The output of qstat can be seriously verbose

- Output is typically 2,000–20,000 lines long for all users!

We have two in-house monitoring tools:

qsum (see qsum –h)

- reformatted from the output of qstat –ext –u "*"
- compactly summarizes running and queueing jobs

qload (see qload –h, qload –ho & qload –hn)

- reformatted from the outputs of qstat –s rs –ext –t –u "*" and qhost –q
- simple text-based diagram of running jobs
- qcheck is an alias for "qload –nall –wn 1 –xp"

Both probably still have bugs in!

A summary of running & queuing jobs

```
-bash-4.1$ qsum
--------------------------------------------------------------------------------
     USERS       QUEUES   JOBS CORES ARRYS TASKS  PROJECTS             STATES (FIRST:LAST)
--------------------------------------------------------------------------------
     abhay    relion.qc      2   192     0     0  strubi.prjc          r=2
   acortes     short.qc    160   160     0     0  mcvean.prjc          r=160
   amorris      long.qc    593   593     0     0  morris.prjc          r=593
   amorris     short.qc      8    64     0     0  morris.prjc          r=8
     batty     short.qc    297   306    21   296  donnelly.prjc        r=297
      frot      long.qc     67    67     0     0  mcvean.prjc          r=67
    hatton     himem.qh      1    16     0     0  donnelly.prjc        r=1
      juha    belmont.q      1    14     0     0  strubi.prjc          r=1
      kvik     short.qc      1     1     0     0  bsg.prjc             r=1
   leffler    spencer.q      1     1     0     0  spencer.prjc         r=1
     luigi    relion.qc      1    96     0     0  strubi.prjc          r=1
   petkova      long.qc      1     1     1     1  donnelly.prjc        dr=1
   schroff      long.qc     21   126     0     0  mcvean.prjc          r=21
   sknight     short.qc      1     1     0     0  brc.prjc             r=1
  tomparks      long.qc      2     2     0     0  hill.prjc            r=2
--------------------------------------------------------------------------------
               RUNNING   1157  1640    22   297
--------------------------------------------------------------------------------
     abhay      queued      1    24     0     0  strubi.prjc          qw=1 (4:4)
   acortes      queued    872   872     0     0  mcvean.prjc          qw=872 (595:2749)
   amorris      queued    241  1928     0     0  morris.prjc          qw=241 (792:1466)
     batty      queued    403  8059   403  8059  donnelly.prjc        qw=403 (1578:3135)
      frot      queued    907   907     0     0  mcvean.prjc          qw=907 (39:1725)
    hatton        held      2    60     0     0  donnelly.prjc        hqw=2 (3136:3137)
        ip      queued      1    10     0     0  bsg.prjc             qw=1 (1:1)
      juha      queued      1    24     0     0  strubi.prjc          qw=1 (2:2)
     luigi      queued      1    24     0     0  strubi.prjc          qw=1 (3:3)
   schroff      queued    708  4248     0     0  mcvean.prjc          qw=708 (5:2370)
--------------------------------------------------------------------------------
               QUEUED    3137 16156   403  8059
--------------------------------------------------------------------------------
```

qsum: what about me?

You can ask qsum just to report on named user(s)

- qsum –u <*username,username…*>
- qsum –u "*"

If you omit the username, qsum assumes you are interested in yourself

- qsum –u
- Equivalent to a compact summary of the output of qstat

```
-bash-4.1$ qsum -u
--------------------------------------------------------------------------------
     USERS      QUEUES   JOBS CORES ARRYS TASKS  PROJECTS            STATES (FIRST:LAST)
--------------------------------------------------------------------------------
      frot     long.qc     71    71     0     0  mcvean.prjc         r=71
--------------------------------------------------------------------------------
               RUNNING     71    71     0     0
--------------------------------------------------------------------------------
      frot      queued    907   907     0     0  mcvean.prjc         qw=907 (1:907)
--------------------------------------------------------------------------------
               QUEUED    907   907     0     0
--------------------------------------------------------------------------------
```

qsum: what about us?

You can just ask qsum to report on a named project(s)

- qsum –p <*proj,proj…*>
- Matches any project name for which each *proj* is a substring
- e.g. mcvean matches mcvean.prja, mcvean.prjb & mcvean.prjc
- e.g. prjb matches all groups *group*.prjb projects

If you omit the project name, qsum assumes you are interested in your default project base name

- qsum –p

```
-bash-4.1$ qsum -p
--------------------------------------------------------------------------------
     USERS      QUEUES  JOBS CORES ARRYS TASKS  PROJECTS            STATES (FIRST:LAST)
--------------------------------------------------------------------------------
   acortes    short.qc   158   158     0     0  mcvean.prjc         r=158
      frot     long.qc    71    71     0     0  mcvean.prjc         r=71
   schroff     long.qc    21   126     0     0  mcvean.prjc         r=21
--------------------------------------------------------------------------------
               RUNNING   250   355     0     0
--------------------------------------------------------------------------------
   acortes      queued   875   875     0     0  mcvean.prjc         qw=875 (692:2490)
      frot      queued   907   907     0     0  mcvean.prjc         qw=907 (29:1374)
   schroff      queued   708  4248     0     0  mcvean.prjc         qw=708 (1:2082)
--------------------------------------------------------------------------------
               QUEUED  2490  6030     0     0
--------------------------------------------------------------------------------
```

Why is no queue listed for queued jobs?

qsum **information is farmed from the output of** qstat

- qstat **does not show queue information for queued jobs**

A job can be submitted to many queues

- qsub –q short.qa,short.qb,short.qc –P mcvean.prja <*script.sh*>

Knowing about target queues could be very useful

- **Information available in** qstat –j <*job_id*>
- qsum –Q **(limited to first 100 jobs in the queue)**

```
-bash-4.1$ qsum -Q -u amorris
*** Too many queued jobs for "-q"
*** Remaining jobs listed simply as "queued"
-------------------------------------------------------------------------------------------
     USERS      QUEUES  JOBS CORES ARRYS TASKS  PROJECTS             STATES (FIRST:LAST)
-------------------------------------------------------------------------------------------
   amorris     long.qc   590   590     0     0  morris.prjc          r=590
   amorris    short.qc     8    64     0     0  morris.prjc          r=8
-------------------------------------------------------------------------------------------
               RUNNING   598   654     0     0
-------------------------------------------------------------------------------------------
   amorris      queued   141  1128     0     0  morris.prjc          qw=141 (101:241)
   amorris    short.qc   100   800     0     0  morris.prjc          qw=100 (1:100)
-------------------------------------------------------------------------------------------
                QUEUED   241  1928     0     0
-------------------------------------------------------------------------------------------
```

What else can qsum do?

Not much else at the moment!

- Check with qsum –h

```
-bash-4.1$ qsum -h

qsum is a script to show the ownership of running and queued jobs across the cluster.
The number of cores for each queue slot is defined in qqueue.conf in the same directory as qsum.
With no arguments the output is for all users.

Syntax:  qsum [-h] [-u [USER,...]] [-p [PROJECT,...]] [-q]

  -h               - show this help information
  -u [USER,...]    - shows jobs for the specified user(s) only
                     If no USER then then current user is shown
  -p [PROJECT,...] - shows jobs for the specified project(s) only
                     If no PROJECT then the current user's default project is used
                     (matches all projects with PROJECT as any substring)
  -Q               - shows target queues for submitted jobs [limited to 100 queued jobs unless "root"]
```

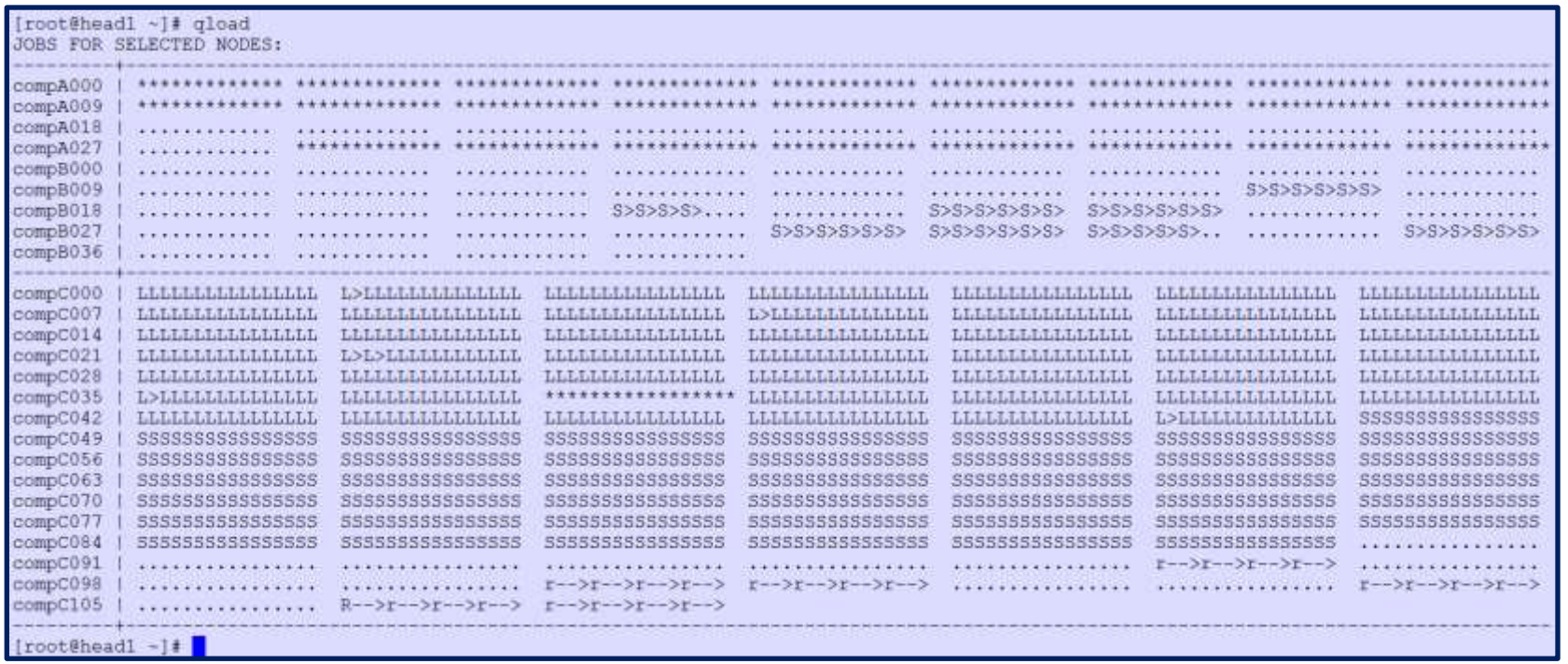
One newer feature is qsum -P

- Just displays "pending" (*i.e.* queueing and held) jobs

What are the compute nodes doing?

The in-house command to look at this is qload

- qload has many arguments, but for most users you won't need any
- qload –h: describes the input arguments
- qload –hn: describes how to select groups of nodes
- qload –ho: describes the output display



Basic output of qload

The default view is to show the main cluster nodes

The view is automatically adjusted to the terminal width

Multiple nodes are shown on one line (normally)

Only the first node on each line is named

- You can usually just count along

Nodes are separated by horizontal bars depending on number of slots (cores)

Each node is represented by a string of characters that is one more than the number of cores it offers

- Most characters are what each core is doing (next slide)
- The last character is a status code, usually blank for a happy node

What is each core doing?

There is one character per core for each node

- ".": A free core on which jobs can be scheduled
- ":": A core in use, but not for a job specified by input flags
- ",": A core on which jobs cannot be scheduled in the current config
- "*": A core on a node which is not running an execd (usually dead)
- "L": A job running in a long queue
- "S": A job running in a short queue
- "R": A job running in a relion queue
- "X": A job running in a special queue
- "L", "L>", "L->", "L-->": A job occupying 1, 2, 3 or 4 cores
- "r-->": A lower case letter is a slave process in multi-node MPI jobs

```
[root@head1 ~]# qcheck
JOBS FOR SELECTED NODES:
---------+-----------------
compC010 | LLLLSSSSSSSSLLLd
compC037 | ****************
compC067 | L---->SSSSSSSSLo
---------+-----------------
```

What are the status codes?

There can be many possible issues with a node

- A single letter represents the most serious issue with the node (last listed issue)

- " ": A blank status code indicates a happy node (the usual state)
- One of "aoACD": usually a temporary issue (load threshold, orphaned, suspend threshold, calendar suspend, calendar disable)
- "<": A node under stress: *i.e.* high load, memory use or swap use
- One of "cdsuE": usually an issue requiring an administrator (config ambiguous, disabled queue, suspended queue, unknown state, error)
- "+": A node on which there are one or more suspended jobs
- "*": A node which is not running an execd (usually dead)

```
[root@head1 ~]# qcheck
JOBS FOR SELECTED NODES:
---------+------------------
compC010 | LLLLSSSSSSSSLLLd
compC037 | ****************
compC067 | L---->SSSSSSSSLo
---------+------------------
```

How stressed are the nodes?

You can ask qload to summarize the stress on each node

- Use the "-l" option to display load and memory usage (in GB)

```
jaga      | ..............................,,,,  2.1   8G
liono     | X.............................,,,,  0.1   4G
jarrah    | ................................  0.3   4G
belmont   | X------------>.................. 11.7  15G
```

- Use the "-ls" option to additionally display swap usage (in GB)

```
jaga      | ..............................,,,,  2.1   8G  0S
liono     | X.............................,,,,  0.1   4G  0S
jarrah    | ................................  0.3   4G  0S
belmont   | X------------>.................. 11.7  15G  0S
```

- Swap usage is usually zero, so not interesting!

Selecting sets of nodes with qload

qload has a very flexible definition of which nodes to display

- qload defaults to showing the main (A, B, C & D) compute nodes
- qload –np [-wn 1] shows just the project servers
- qload –nh shows just the high-memory servers
- qload –n belmont or qload belmont shows just one node (in detail)

```
This is usually simple to use and probably best demonstrated by some examples:

  jaga,liono                    Nodes named jaga and liono
  compC[000-107]                Nodes named from compC000 to compC107
  %a                            Nodes named named in the group %a, in this case the
                                "A" nodes: nodes from compA000 to compA035
  %b,^compB017                  Nodes in the group %b, in this case the "B" nodes, but
                                node compB017 is excluded from the list
  %bc6                          Nodes in the group %bc6, in this case blade chassis
                                number 6: nodes from compC090 to compC107
  compC[000-107:18]             Nodes in slot 1 in each blade chassis
  @relion.hgc                   Nodes in the SGE hostgroup @relion.hgc, in this case
                                nodes that can run "relion" MPI jobs

Currently defined GROUPS are: %all - all managed nodes; %ab - A and B nodes;
%abc - A, B and C nodes; %a - A nodes; %b - B nodes; %c - C nodes; %h - high
memory nodes; %p - the project servers; %t - the rescomp nodes; %bc[n] - the
nodes in each Fujitsu blade chassis, from 1 to 6

The current default selection is equivalent to "-n %abc"
```

Looking at the load on a single node

If you select a single node qload gives more detailed output

- Detailed output is also turned on with the "verbose" flag "-v"
- qload –nh -v shows just the detailed load on the high-memory servers

```
[root@head1 ~]# qload -nh -v
DETAILED QUEUE, JOB, MEMORY AND SWAP USE FOR ALL SELECTED NODES:
---------+-------------------------------------------------------------------
compH000 | H----------------------------------------------> 48.0   22G  0S
---------+-------------------------------------------------------------------
   himem.qh             BIP   0/48/48
---------+-------------------------------------------------------------------
8857370    0.000 segmentref juha         strubi.prjc      r     01/20/2016 10:50:25 himem.qh@compH000              48
---------+-------------------------------------------------------------------

---------+-------------------------------------------------------------------
compH001 | R----------------------------------------------> 16.9  219G  0Sd
---------+-------------------------------------------------------------------
   himem.qh             BIP   0/0/48        d
   relion.qh            BIP   0/16/16
---------+-------------------------------------------------------------------
8851461    0.092 relion     mgrange      strubi.prjc      r     01/20/2016 11:40:16 relion.qh@compH001             48
---------+-------------------------------------------------------------------

---------+-------------------------------------------------------------------
compH002 | G----------------------------------------------> 48.0   21G  0S+
---------+-------------------------------------------------------------------
   himem.qh             BIP   0/0/42        dS
   ginn.qh              BIP   0/48/48
   long.qc              BIP   0/6/48        So
---------+-------------------------------------------------------------------
8855637    0.008 fm         asanniti     jknight.prjc     S     01/19/2016 13:10:51 suspended@compH002              0
8855793    0.092 index      ginn         strubi.prjc      r     01/19/2016 15:37:43 ginn.qh@compH002               48
---------+-------------------------------------------------------------------
```

Other command line options for qload

qload has lots of useful command line options, but mostly aimed at system administrators

```
Syntax:  qload [-h|-hn|-ho] [-j JOB_ID,...] [-u [USER,...]] [-p [PROJ,...]]
               [-q [QUEUE,...]] [-l[s]] [-d] [-v] [-x] [-xp]
               [-wc CHARS|-wn NODES] [-nGROUP|[-n] LIST]

  -h                - show this help information
  -hn               - show help information for node list and group selection
  -ho               - show help information for the output format
  -j JOB_ID,...     - explicitly show the specified job ID(s) (overrides -u, -p and -q)
  -u [USER,...]     - explicitly show jobs for the specified user(s) only (not with -j)
  -p [PROJ,...]     - explicitly show jobs for the specified project(s) only (not with -j)
                      (matches all projects with PROJ as any substring)
  -q [QUEUE,...]    - explicitly show jobs for the specified queue(s) only (not with -j)
  -l[s]             - show load, memory use and [with ls] swap use in addition to jobs
  -d                - show information only for selected "dead" nodes)
  -v                - show verbose information for all selected nodes (default for a single node)
  -x                - show exclusively the subset of nodes running (selected) jobs
  -xp               - show exclusively the subset of nodes currently reporting problems
  -wc CHARS         - width of line in characters (only one of -wc or -wn)
  -wn NODES         - width of line in nodes (only one of -wc or -wn)
  -nGROUP           - show information for the specified group of nodes
  [-n] LIST         - show information for the specified list of nodes
```

What are the project servers doing?

The project servers are not shown in default qload output

- Select the node group "p" for the known project servers
- Select one node per line for the output
- Ask for details on the load, memory and swap space usage
- qload –np –wn 1 -ls

```
[root@head1 ~]# qload -np -wn 1 -ls
JOBS, LOAD, MEMORY AND SWAP USE FOR SELECTED NODES:
---------+-----------------------------------------
jaga     | ..............................,,,,  2.1   8G  0S
liono    | X.............................,,,,  0.1   4G  0S
jarrah   | ................................  0.3   4G  0S
belmont  | X------------>.................. 11.7  15G  0S
---------+-----------------------------------------
coolibah | ................  0.0   4G  0S
---------+-----------------------------------------
```

Are there any obvious problems?

qcheck reports on obvious potential cluster issues

- Mainly intended for system administrators
- qcheck is an alias for "qload –nall –wn 1 –xp "

```
[root@head1 ~]# qcheck
JOBS FOR SELECTED NODES:
---------+------------------
compC010 | LLLLSSSSSSSSSLLLd
compC037 | ******************
compC067 | L---->SSSSSSSSLo
---------+------------------
```

"qacct" holds all records of cluster activity

Last year, Winni was winning the race: 2,872,418,832s = 91y

- Note: WALLCLOCK can be greater than or less than CPU

```
[root@head1 ~]# head -n 2 qacct.log; qacct -o | sort -n -r -k 5
OWNER        WALLCLOCK         UTIME         STIME           CPU               MEMORY                 IO
=================================================================================================
winni       2382301270 2484037020.560  86901869.432 2872418832.182     62650812919.367        341297.179
abhay        238951659 1777352696.697   5636305.695 2442072111.750      7342403576.611        555685.395
quang       2542659858 1409670121.162  15253423.546 1957784086.727      6076923543.409     176989781.903
pkalbers    1142779152 805972918.808    7981886.165 1221543516.007      4178093461.965      28633874.572
frot         947906037 662813387.896     550379.697 942510221.023       997703349.339        153688.228
joezhu       824565186 340958934.266     381637.493 790850679.743       544538501.170         53450.581
amorris      690359156 623190114.065     761945.105 682648343.852      6579112255.301         62643.949
jfertaj      596822976 152187281.841     140985.078 577939910.148      2622039611.389          1418.408
oliver       478883764 466395403.787    3481083.429 472707510.995        86169920.099      23928193.913
schroff      438766689 371283525.901     413322.945 437491249.252      4795068604.388         38245.680
caina        451845868 399797873.482    9525394.522 429257863.259      2862026844.662        806922.898
batty        393329336 364774976.372    5316687.214 409486349.341      2763569699.786        938866.945
rwdavies     336233233 275059782.512     232214.047 332711743.168       654071255.454        286641.673
marchini     323791393 258724828.564     234473.072 320104407.769        12955558.847          4635.516
kelliott     590088552 285095615.337    7203728.199 304268302.842      2390072557.185      12387104.549
hatton       108858489 291071575.582    2195420.615 302895720.820      3185567938.943       1665207.424
vivekn        76437435 221658629.886    3236468.829 286479285.941       498629314.345         40618.322
george       361469236 268367822.955    1215984.700 272877916.849        77843980.304       3594082.408
clare        252384384 243803011.698     538274.621 245080599.465       236638233.951        792989.495
hilary       167915762 205278037.913    2691833.167 229413600.656       488251023.789        103563.557
juha          35572267 149301187.925    1131980.848 220528803.910       762496625.647        374600.590
gav          190481424 179369941.281    4385933.474 185636924.738       397727783.982          5075.299
acortes      179624039 172528141.517     398948.398 177127313.095       349164778.034        154049.526
davismcc      72767118 164915378.712    1376505.926 167866374.083       115951363.674         47067.287
rivas        196181952 141741895.327    1935606.141 150215822.117     34648132505.392        301986.678
mvdbunt      114634728 114921614.027   20386271.512 137896111.279       888692965.572        866882.327
leffler      123206138  96081001.791   33710361.695 134097940.508        78507748.458       2173864.879
wadh4116      54687282 103261708.562     400207.291 131159344.302      1804498138.848        288616.355
```

"qacct" for a single job or task

Use qacct –j <*job_id*> [-t <*task*>]

- qname: **queue job ran in**
- hostname: **where job ran**
- jobnumber: **job ID**
- taskid: **number of task for array job**

- start_time: **when job started**
- end_time: **when job ended**

- failed: **should be 0**
- exit_status: **exit status of script:**
  - **<128 user defined**
  - **>128 SGE signal**
- 134 = SIGABRT **e.g. memory exceeded**
- 137 = SIGKILL **e.g. time exceeded**

- ru_wallclock: **elapsed time**
- ru_utime: **user cpu time**
- cpu: **cpu time used**
- maxvmem: **maximum size reached by job**

```
[nilufer@login1 ~]$ qacct -j 3202376 -t 2
==============================================================
qname        short.qa
hostname     compA000.cluster3
group        zondervan
owner        nilufer
project      zondervan.prja
department   defaultdepartment
jobname      job
jobnumber    3202376
taskid       2
account      sge
priority     0
qsub_time    Thu Jun 20 09:46:34 2013
start_time   Thu Jun 20 09:47:26 2013
end_time     Thu Jun 20 09:47:27 2013
granted_pe   NONE
slots        1
failed       0
exit_status  0
ru_wallclock 1
ru_utime     0.322
ru_stime     0.058
ru_maxrss    0
ru_ixrss     0
ru_ismrss    0
ru_idrss     0
ru_isrss     0
ru_minflt    13165
ru_majflt    0
ru_nswap     0
ru_inblock   0
ru_oublock   0
ru_msgsnd    0
ru_msgrcv    0
ru_nsignals  0
ru_nvcsw     549
ru_nivcsw    32
cpu          0.380
mem          0.000
io           0.000
iow          0.000
maxvmem      0.000
```

… that's enough from me!

The final two talks are given by Jon Diprose:

- Talk 5: ResComp centrally-managed applications
  (1/2 10:00 Room B; Jon Diprose)

Talk 6: Doing your own thing (compiling and customizing)
(2/2 10:00 Room B; Jon Diprose)

Thank you for your attention… any questions?

Next talk 10:00am Wednesday… "ResComp centrally-managed applications"