Monitoring the HPC cluster

Why monitor the cluster

When you submit a job you request a resource allocation in the cluster (number of nodes, number of CPUs, memory etc). When the job is executed all of those resources are allocated to that job; even if they're not being used as intended by your script they won't be available to anyone else whilst your job is running.

Resource availability

Checking the cluster utilisation before you submit a job allows you to adjust your script according to the resources available. Requesting too many resources for a single job may leave your job sitting in the queue for a long time, potentially days. Your job may get executed earlier if you request fewer resources (12 CPU's instead of 24 CPU's). Even if the job takes twice as long to compute it may still finish earlier if it progresses through the queue more quickly!

Code performance

You may be using a command that is multi-threaded and therefore can utilise multiple CPUs at once. Such programs typically require setting the number of CPU cores to use. Checking that your job is using the number of CPU's expected means your jobs will finish in an optimal time frame. Also, most processing pipelines are a mixture of single threaded and multi-threaded tasks. If your job spends a long time in the single threaded tasks there may be little benefit in requesting a large number of cores for that job. Again, your job could spend a long time in the queue waiting for resources to become available before it's executed, and once those resources are allocated to a job they're not available for anyone else whether they're in use or not.

Monitoring the queues

The *qstat* command shows information about the queue.

```
$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1220444.tizard1            ...build_idx.pbs jan                    0 Q tizard
1241008.tizard1            Activator        ahao                   0 Q tizard
1241230.tizard1            MARE2DEM         ydidana                0 Q tizard
1246866.tizard1            zentest          luz             284:06:3 R tizard
1246867.tizard1            zentest          luz             282:48:5 R tizard
1246868.tizard1            zentest          luz             283:53:1 R tizard
1246869.tizard1            zentest          luz             283:59:0 R tizard
1249511.tizard1            Ce4O2Pt-1-253    akarayilan      237:01:1 R tizard
1249514.tizard1            Ce4O2Pt-1-256    akarayilan      204:09:3 R tizard
1249527.tizard1            Ce4O2Pt-1-269    akarayilan      184:29:5 R tizard
1249549.tizard1            Ce4O2Pt-1-291    akarayilan      116:43:1 R tizard
1249554.tizard1            Ce4O2Pt-1-296    akarayilan      93:25:11 R tizard
1249557.tizard1            Ce4O2Pt-1-299    akarayilan      87:43:33 R tizard
```

```
1249625.tizard1            ...me.FR07958956 mcorbett        1508:45: R tizard
1249626.tizard1            ...me.FR07958751 mcorbett        1668:27: R tizard
1249627.tizard1            ...me.FR07958750 mcorbett        1185:44: R tizard
1249628.tizard1            ...me.FR07958743 mcorbett        1077:07: R tizard
1249735.tizard1            DirkBoehe        dboehe          21:40:40 R tizard
1249754.tizard1            kartxx_300       akartusinski    17:51:44 R tizard
1249759.tizard1            kartpipe2ph      akartusinski    05:08:15 R tizard
1249764.tizard1            intercept1       tberezowski     108:08:4 R tizard
1249773.tizard1            equilibration    bfielke         17:57:18 R tesla
1249779.tizard1            ...i14_8-8-gIIID rhudson         08:17:30 R tizard
```

To look at just your own jobs use *qstat -u username.*

To list the available queues and associated memory and maximum walltimes, use the *qstat* command.

```
$ qstat -q
server: tizard1
Queue             Memory CPU Time Walltime Node  Run Que Lm  State
----------------- ------ -------- -------- ----  --- --- --  -----
gtx3                23gb    --    50:00:00   --    0   0 --   E R
short               48gb    --    05:00:00   --    0   0 --   E R
gtx                 23gb    --    50:00:00   --    0   0 --   E R
tesla              126gb    --    100:00:0   --    2   0 --   E R
gpu                 94gb    --    50:00:00   --    0   0 --   E R
workshop            48gb    --    05:00:00   --    0   0 --   E R
tizard              --      --    100:00:0   --  374 -219 --   E R
                                                 ----- -----
                                                   376  -219
```

It is also possible that your job is not able to run and has become blocked. The *showq* command will list any blocked jobs. Jobs are usually only blocked when you've requested resources that will never become available, such as requesting more nodes than are actually present in the cluster, or more memory than is available on a single node.

Check the details of a specific job in more detail using *qstat -f jobid (*just use the numeric part of the JobID).

Monitoring the cluster

The pbstop command is a cluster monitoring tool. It provides an overview of the worker nodes in the cluster and displays the current cluster usage, node status, CPU usage and the job queue. You can use it to view the cluster utilisation and available resources.

To show jobs from all users and simply run pbstop without additional arguments.

```
$ pbstop
```

To show only your jobs use the -u option (replace *username* with your username).

```
$ pbstop -u username
```

You'll need to arrow up and down to scroll across the screen. There are several sections to the output of pbstop.

The first line shows that 538/1116 CPUs are presently in use across 21/32 of the worker nodes. The job queue has 164 jobs with 47 currently running.

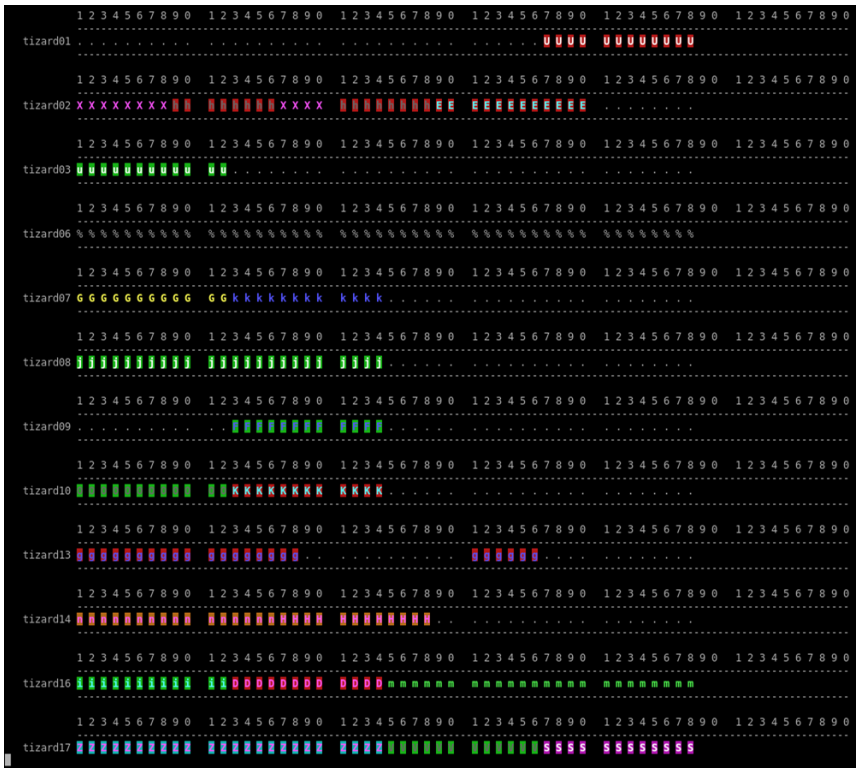
The second line describes the status of the 32 worker nodes.

```
Usage Totals: 538/1116 Procs, 21/32 Nodes, 47/164 Jobs Running
Node States:     5 down                3 down,offline          18 free                 4 job-exclusive         2 offline
```

Node states of "down", "offline" or "unknown" means the nodes are not presently available in the cluster. "Job -exclusive" and "busy" means all resources of those nodes are being used. "Free" means there are CPU resources available on those nodes.

The second section of *pbstop* is a table which lists all the worker nodes in the left most column (*tizard01, tizard02...tizard205*). The numbered columns along the top are the number of CPUs (in blocks of 10) belonging to each node. All Tizard nodes have 48 CPU cores. Each CPU is represented by either a white dot, if the CPU is idle, or a coloured character if the CPU is in use. Each different coloured character refers to a different job which is listed further down the screen.

Only the jobs in the "Running" state will appear on the table.



Monitoring your job

Once your job moves from the "Queued" state to the "Running" state you can see how your job behaves. Just because CPU resources have been allocated to your job doesn't necessarily mean your script is using all of those resources. Using *pbstop* one can see on which node a job is running. Running the *top* command will the actual CPU and memory usage in real-time.

From tizard1, we can use *ssh* to connect to tizard08 for us and run the *top* command; the output from the *top* command will be sent back to tizard1 for us to see. Since *top* will display a continuously updating screen, *ssh*

requires the *-t* option. You may also like to add *-u username* (using your own username) after the *-M* option to just list your own processes.

```
$ ssh -t tizard08 top -M
```

(The warning is normal and will occur only the first time you access a node.)

Both of the jobs below have been allocated 24 CPUs, however, one is actually using all 24 cores and the other just a single core. The *%CPU* column shows the CPU usage, where *100%* represents one CPU being fully utilised and *2400%* represents 24 CPU's being fully utilised.