Draft Intro Document for Blue Pebble – Experimental Users Only

Simon Burbidge

Last Updated 7 Nov 2019

The Blue Pebble system is available for experimental access for experienced test users only.

System status:

There is just one login node at present.

There are 43 compute nodes in use.

The new 100Gbit ethernet switches have been installed now.

All computes nodes are under PBSPro management

To login:

ssh bp1-login01b.acrc.bris.ac.uk

Use your UoB password

Storage  
Users have:

home directory, /home/<username>, $HOME , default quota 20GB and 500,000 files.

work directory, /work/<username>, $WORK , default quota 1,000Gb and 500,000 files.

Use the quota command to view current quotas and usage.   
quota -s  
is the most useful form.

*NOT AVAILABLE YET: scratch directory, /scratch/<username>, $SCRATCH -- which is a convenience link to $WORK . This is for future implementation and there is no promise to provide it. The link is provided to help with job script compatibility.*

tmpdir , located on /tmp of the local disk of the node the job is running on, $TMPDIR.  
This directory is unique for every job which runs and is created at the start of the job and destroyed at the end. This should be used for temporary working files for the duration of the run.

The filesystems aren’t tuned yet, performance may be poor, its on the list of things to do.

There are a few modules installed now. We will add more.  
Users should not install their own versions of common applications, we will do it centrally to avoid duplication and ensure that the installations are optimised.

For people who have their own code:

1. We only support the Intel compilers, other compilers should not be used. There should be no compatibility issues in doing this. We have support from Intel if bugs or problems are found
2. We only support the Intel MPI library. Other variants of MPI must not be used. This is important as it will significantly help the stability of Blue Pebble.

PBSPro

The queueing system is PBSPro – its quite like the system on BC3 rather than the SLURM system on BC4.  
Note that on Blue Pebble you should not submit to a specific queue – this will usually be disallowed. The queue selection is made by PBS on the basis of the resources you specify in the #PBS directives and the jobs automatically routed to an appropriate queue.

If you have jobs that fail to submit to the default queue, please let me know and I’ll fix it. I’ve set up a user oriented queue configuration – it stops you submitting non-optimal job sizes eg. 2 nodes with 4 cpus each (where 1 node with 8 cpus is preferable).

Memory and cpu resouces limits are being enforced. Users from BC3 , where resource enforcement is patchy, might find this a little difficult at the start , but in the longer term it will ensure a more stable and reliable system. Its important that you request the resources that are needed for the job reasonably accurately and not excessively. For example, don’t request all the memory on a node when you don’t need it. By doing this, the scheduler has the best information to do its job and it means that in future, when we purchase new nodes, we have information as to the size of work people actually run and we can tailor our purchases appropriately.

Jobs sscripts hould normally start with

#!/bin/sh

We don’t support any other login shells. If you want use others then you can call them inside your script.

Useful #PBS directives are:

#PBS -l walltime=hh:mm:ss set the job walltime limit  
#PBS -j oe Join stdout and stderr to the same file

Node resources are specified using the select directive:

#PBS -l select=N:ncpus=C:mem=M

Where N is the number of nodes , C is the number of physical CPUs (cores) per node, M is the amount of memory per node. Nodes on Blue Pebble currently have 24 or 28 cores. Multi-node jobs are expected to be in multiples of that number of cores eg:

#PBS -l select=2:ncpus=24 for a 48 core job.

Jobs which use less than a whole node (ie. Fewer than 24 cores) share nodes.

Single processors runs are specified like this:

#PBS -l select=1:ncpus=1

A run using 8 cpus would be specified like this:

#PBS -l select=1:ncpus=8

If you need to use more than the default amount of memory, you need to add that , eg:

#PBS -l select=1:ncpus=1:mem=4gb

We are expecting to run a lot of single core array jobs on Blue Pebble and the scheduler can make very good use of the odd spare cores for running array jobs.

Jobs start with their current directory in $TMPDIR, which is a directory on the local disk of the node that exists only for the duration of the job. Use this wherever you can, for performance!

MPI implementations should work with PBSPro and pick up the resources reserved in the job for the MPI run. Thus you should use  
mpiexec progname  
instead of   
mpirun -n <numprocs> progname

The parameters to set the resources for parallel jobs in PBS are mpiprocs and ompthreads eg:

#PBS -l select=2:ncpus=24:mpiprocs=24:ompthreads=1 --- gives you 48 mpi ranks across 2 nodes

#PBS -l select=2:ncpus=24:mpiprocs=1:ompthreads=24 --- gives your one mpi rank per node, and 24 threads per node (ie. 24 threads per mpi rank)

#PBS -l select=2:ncpus=24:mpiprocs=12:ompthreads=2 --- gives you 24 mpi ranks across 2 nodes and two threads per mpi rank. *(note for advanced users, using mpiprocs is the way to reduce the number of cores you use per node, eg if you are memory bandwidth limited, you should still reserve the 24 cpus so that no-one else grabs them)*

These set up the environment variables $MPIPROCS and $OMP\_NUM\_THREADS for the MPI environment.

**Job Sizing**

The normal maximum walltime for automatically scheduled jobs on Blue Pebble is 72 hours. We do run longer jobs by arrangement.

The vast majority of nodes on Blue Pebble have 24 cores, so if you have proper parallel code that scales up to 24 CPUs then that’s the optimum number to choose. If your runs don’t scale that well, then use the number of CPUs that gives the best performance.  
For multi-node jobs, you should use a multiple of 24 CPUs.

If you are running task farming or other multiple serial arrangements you are strongly recommended to switch to using array jobs (#PBS -J) instead. These are much more efficient and reliable.

**GPUs**

We have some nodes with graphics cards installed. These are NVIDIA RTX2080Ti cards.   
There are 4 cards per node and you can use any number you like ( only if your code supports multiple GPUs, of course).

To request gpus, simply add ngpus to your #PBS request and include loading the cuda module in your run script, eg:

#PBS -l select=1:ncpus=1:ngpus=1

module load lang/cuda

The system will allocate available gpu(s) for you and make them available via the usual CUDA environment.  
There are currently nodes with either GTX1080 gpus or RTX2080 gpus or RTX2080Ti gpus. If you wish to make jobs run on a particular gpu type, you can add a gputype=*typeofgpu* to your select directive. eg:

#PBS -l select=1:ncpus=1:ngpus=1:gputype=GTX1080  
or  
#PBS -l select=1:ncpus=1:ngpus=1:gputype=RTX2080

Some of the GPU nodes have an SSD storage device instead of a hard drive. This is needed for certain GPU applications that do a lot of random I/O on their data files. The $TMPDIR directory is located on the SSD. To make a job run on a node with an SSD rather than disk add ssd=true to the select directive. eg:

#PBS -l select=1:ncpus=1:ngpus=1:ssd=true