PACE-ICE Documentation

Worms VIP Subteam Fall 2023 - Nathaly Jose-Maria

# **ICE-Cluster [1,2]**

ICE (Instructional Cluster Environment) is one cluster of many services provided by PACE. It is configured with the same hardware and software as Phoenix, another cluster provided by PACE. This cluster is specifically for educational experimentation with HPC and GPU programming separate from production, or real-world industry, research resources.

## **Cluster Structure [3]**

The individual ICE cluster is structured with a head node, several computational nodes, several storage servers, and two schedulers. They are defined respectively as follows:

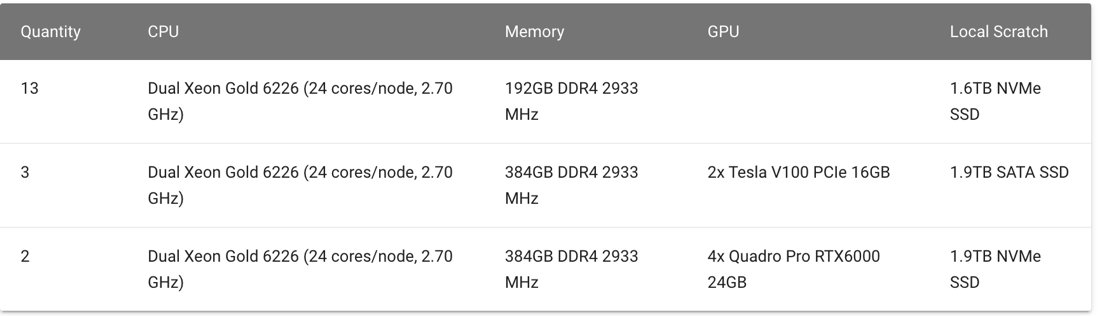
### Head Nodes

The access point into the cluster. There is one, and it is shared by all permissible accounts. You can submit jobs from here, edit & compile small programs, and access storage servers. However, because it is a shared access point, it should be limited to solely preparing files for computational nodes and submitting jobs for use of those computational nodes. The easiest way of getting to the head node is through Open OnDemand. Read on for more on Open OnDemand and Getting Started.

### Computational Nodes [2, 3]

These nodes, sometimes referred to as computer nodes or computer nodes, are the nodes responsible for running all computations directed into the cluster. They are only accessible when assigned to a [job](#_Jobs_[3]) by a scheduler. The node accesses specified storage servers to pull files and information necessary for calculations. This is why it is imperative to ensure the proper files are placed in the head node prior to submitting a job.

The computational nodes available for use through the ICE cluster are as follows. Each node is connected via InfiniBand HDR100 (100 Gbps) interface to PACE’s fabric. Moreover, each user is provided with 15GB of storage available from across all nodes and is backed up daily.



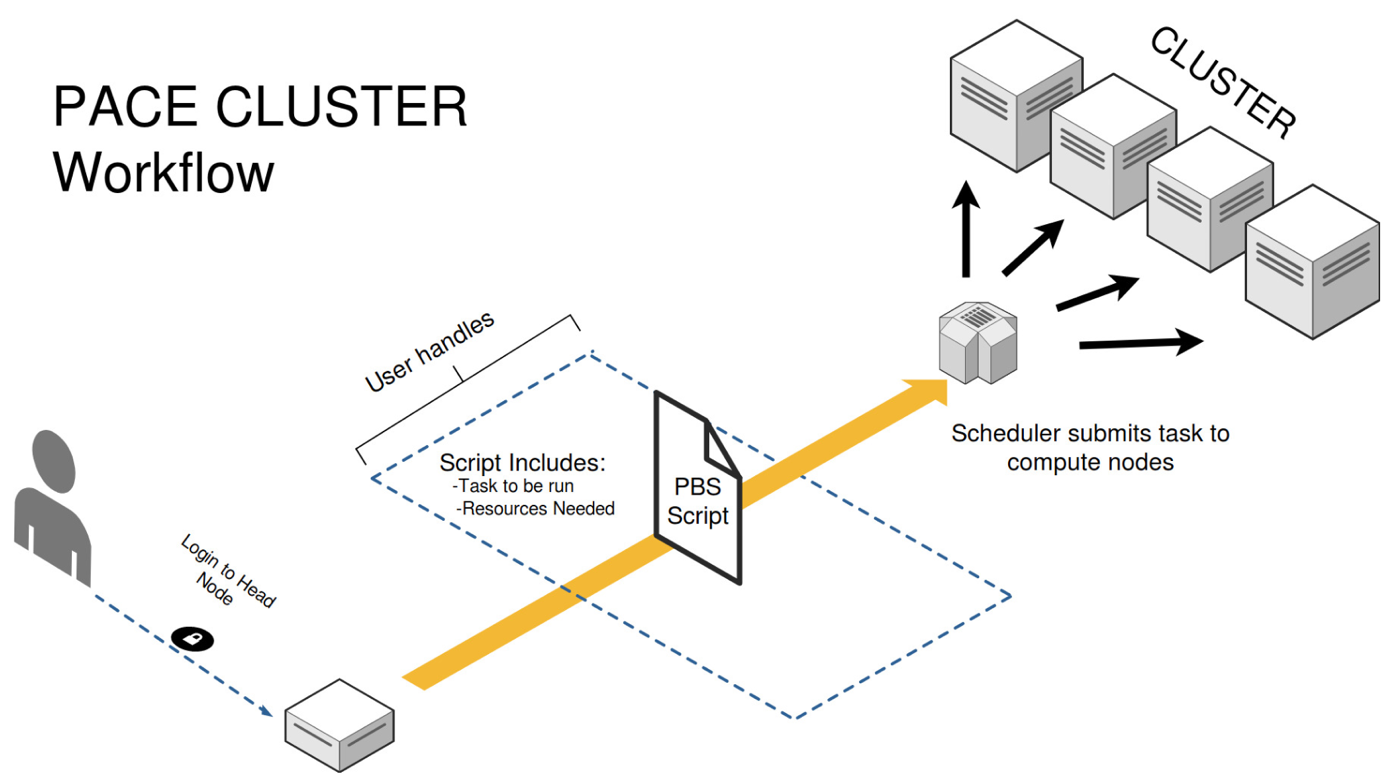
### Storage Servers

This is where the files necessary for computational node computations are held and operated from/on. Servers are accessible through the head node, and as noted, each user is assigned a maximum of 15GB of storage for use. To check one’s current utilization of storage, one must run the command pace-quota.

### Scheduler

The scheduler manages job requests for utilizing computational nodes. The request is managed by assigning resources on computational nodes for an individual job. This ensures fair use of the shared resources (the computational nodes).

## **Workflow [4,5]**





1. Login onto Cluster through Head Node
2. Submit PBS script
   1. Ensure all necessary files exist in cluster storage
3. Submit Job
   1. Scheduler submits task to Compute Nodes
4. Collect Results
   1. Computer Nodes execute script & produce output

## **Jobs [3]**

Jobs are requests from the scheduler to assign space on a computational node. Space is assigned when it becomes available. As such, all jobs wait in the same queues for available space. However, a specific queue must be specified for your computation independent of the type of job you submit.

### Batch Jobs

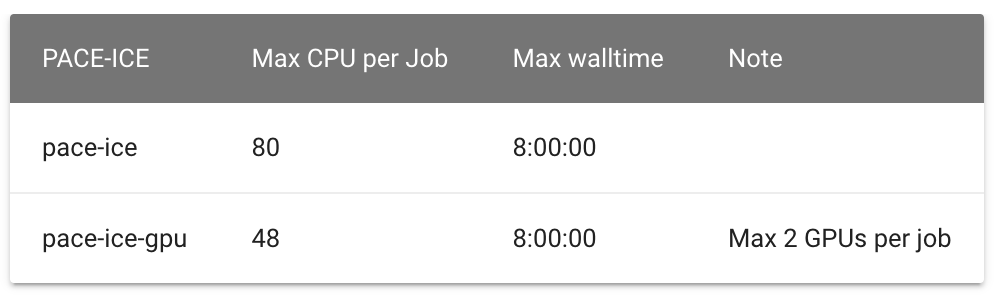
This job type is for “submit and forget” workflow. This means it’s ideal for large computations requiring many CPUs and longer computations requiring more time. To create a batch job, write a PBS script with the commands necessary for your computation, and submit the request to the scheduler using the command qsub.

### Interactive Jobs

This job type is for computations requiring interactive use or live input during execution. It is likewise ideal for intensive computation. There are two methods of creating an interactive job; however, we will use the easier option: Open OnDemand.

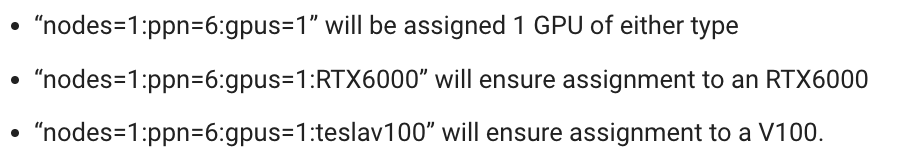
## **Queues [3]**

Queues are used by the scheduler to determine which job request will receive the space, or computational node allocation, that becomes available as processes complete. There are two queues to choose from depending on your computational needs. They are as follows.



ICE computational nodes operate on two GPUs: TeslaV100 and Quadro RTX6000. By default, jobs will be assigned to the first available GPU of either type. However, if a specific architecture is necessary, you may indicate so in your job request through an additional flag. To request the TeslaV100, use the identifiers teslav100, TeslaV100, or TeslaV100-16GB. To request the RTX6000, use the identifier RTX6000.

Here are examples of flag additions:



# **Open OnDemand [6]**

Open OnDemand is an interface that allows access to PACE clusters through a web browser. Everything is managed through a GUI, including file management, directory access, job submission, monitoring, output, & queue status, and interactive application execution.

## **Getting Started**

Access the head node [at this link](https://docs.pace.gatech.edu/ood/guide/#:~:text=PACE%2DICE-,PACE%2DICE%20OnDemand,-COC%2DICE). ***Before pressing the link, ensure you are connected to on-campus WiFi or the campus VPN***. Visit [Configure the GT VPN](https://docs.pace.gatech.edu/gettingStarted/vpn) for assistance with VPN setup.

Follow the steps detailed on the official PACE guide for operating the GUI [6].

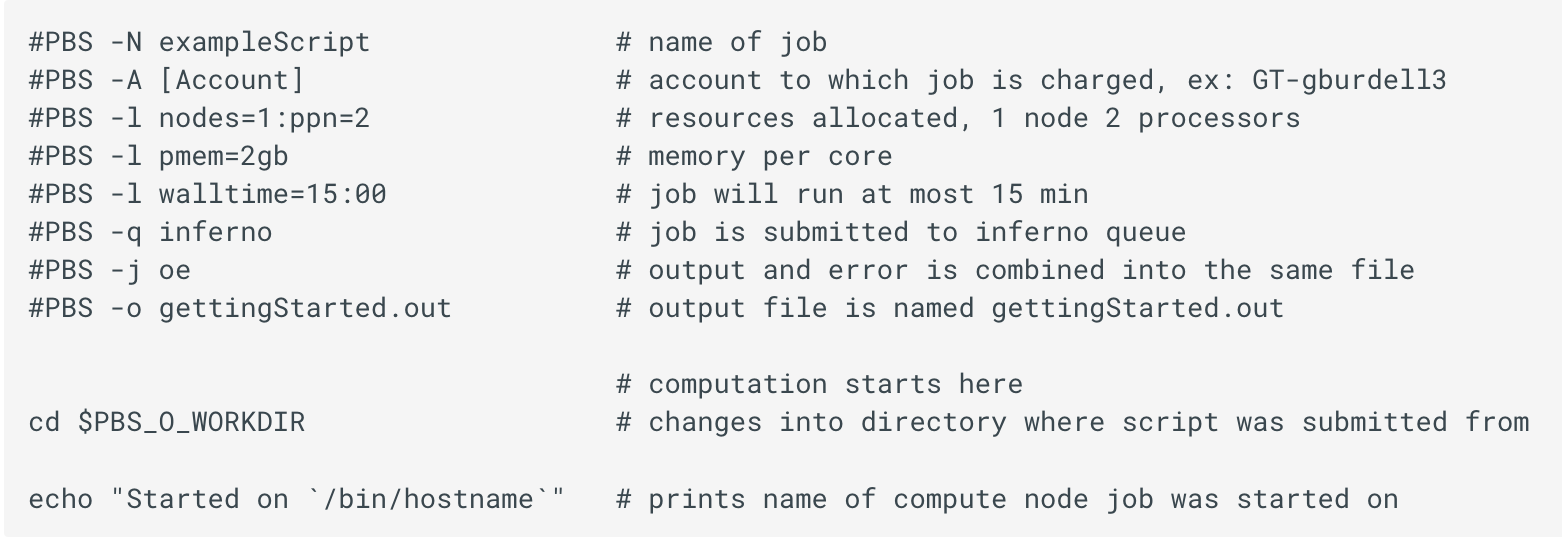
# **PBS Scripts [7]**

PBS are Portable Bash Scripts. They are used to tell the scheduler what to run and how to run it. More specifically, it tells the scheduler what resources the job will need (i.e. number of processors, memory, time, etc) and defines the computation (what application or software should be run).

## **Writing Scripts**

### Directives

PBS Scripts are made up of directives, which identify characteristics of your job that help shorten the qsub job submission requirements. They work like flags being passed into a Python script through your terminal. The following is an example of a PBS script made with directives:



One should add a blank line at the end of the script to ensure the script executes.

The following is a list of commonly used directives, but a full list can be found in the manual. The manual is accessible by logging into the luster and running man qsub.

|  |  |  |
| --- | --- | --- |
| Flag | Characteristic | Explanation & Examples |
| -A | Account | This defines the account to charge for the job.  This is required for Phoenix cluster job submissions.  If you don't know the charge accounts you have access to, run the command pace-quota |
| -l | Resource List | This defines the resources that the job requires through several variables.  walltime: how much time the job requires to run; likewise, how long until the job will be stopped  Ex: #PBS -l walltime=10:00:00  i.e. job requires 10 hours to run, will be stopped at 10 hours  nodes: how many nodes the job requires  ppn: how many processors per node the job requires  cores: how many cores your job requires  Ex: #PBS -l nodes=2:ppn=4  i.e. job requires two nodes and 4 processors per node, for a total of 8 processors  Ex: #PBS -l nodes=1:ppn=1:cores24  i.e. job requires 1 node with 1 processor per node and 24 cores  nodes=*nodeName*: reserve a job on a specific node where “*nodeName*” is your node’s name  Ex: #PBS -l nodes=rich133-p32-33-l.pace.gatech.edu+rich133-p32-33-r.pace.gatech.edu  i.e. job request 2 specific nodes, the first by the name “*rich133-p32-33-l.pace.gatech.edu*” and the second by the name “*rich133-p32-33-r.pace.gatech.edu*”, joined by a plus sign (+)  mem: how much memory the job requires over all nodes  #PBS -l mem=2gb - job requires 2gb over all nodes  feature: what processor type the job requires  Ex: #PBS -l feature=intel  I.e. ob requires an Intel processor  Ex: #PBS -l feature-amd  i.e. job requires an AMD processor |
| -q | Queue | This define the queue the job is going to be submitted to.  Ex: #PBS -q inferno |
| -N | Job Name | This defines the name for the job as it should appear in the queue.  Ex: #PBS -N mpiScript |
| -o | Output | This defines the name of the output file.  Ex: #PBS -o results.out |
| -j oe | Join | This combines the output and error into one file.  Ex: #PBS -j oe |
| -m <a,b,e> | Write (Email) | This will send a status email based on any combination of a,b,e.  b is when job begins, e is when job ends, and a is if job is aborted.  Ex: #PBS -m abe send an email for all three scenarios |
| -M | Send (Email) | This defines the email addresses to send status emails to.  Ex: #PBS -M user1@gatech.edu user2@gatech.edu will send emails to user1 and user2 |

### Creating a Script

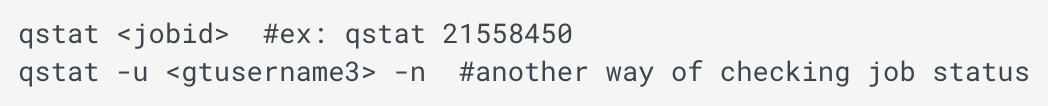
Copy the example script in [Appendix Figure 1](#_Appendix_1), paste the code into a text editor, save the document as a .pbs file, then send the script to your account on the cluster.

## **Submitting to the Scheduler**

Navigate to where the PBS script is located and submit by running qsub fileName.pbs where “*filename*” is the name of your actual file. A succesful submission will output a result similar to the following: 21558450.shared-sched.pace.gatech.edu. This is the jobID used to check the status of the job.

## **Check the Status of Your Job**

Depending on the resources available, your job will take anywhere from minutes to hours to complete. Type qstat to check the status of your job.



A grid will be displayed with more information about the job. The “S” column represents the status with “Q” meaning a job is in the queue, “R” meaning a job is running, and “C” meaning a job is complete. However, note that C can mean a job failed, it was deleted, or it finished running.

You can delete a job using the command qdel jobId where “*jobID*” is your job ID.

More useful commands are available [here](https://docs.pace.gatech.edu/gettingStarted/commands/) [9].

## **Accessing your Results**

All of a job’s output will appear in the output file already named by the directive in the PBS script and can be found in the same folder. Any additionally generated files will also appear I the same folder as the PBS script. Output file contents can be viewed through a text editor. When generating multiples files from a PBS script, consider creating a [local scratch storage](https://docs.pace.gatech.edu/storage/localTmp/) [8].

## **Environmental Variables**

Various environmental variables are generated in parallel with a batch job and can be used in the batch script. See [Appendix Figure 2](#_Appendix_1) for an example of some.

# **References**

1. PACE (2017). Services. *Partnership for an Advanced Computing Environment*. <https://pace.gatech.edu/services>
2. PACE (2017). PACE’s Instructional Clusters: PACE-ICE and COC-ICE. *PACE Cluster Documentation*. <https://docs.pace.gatech.edu/ice_cluster/ice/>.
3. PACE (2017). ICE User’s Guide. *PACE Cluster Documentation.* <https://docs.pace.gatech.edu/ice_cluster/ice-guide/>
4. PACE (2017). PACE: Examples and Guides for using PACE clusters. *PACE Cluster Documentation*. <https://docs.pace.gatech.edu/>
5. PACE (2017). New User Guide. *PACE Cluster Documentation.* <https://docs.pace.gatech.edu/gettingStarted/newUser/>
6. PACE (2017). Open OnDemand Instances for PACE”s Clusters. *PACE Cluster Documentation*. <https://docs.pace.gatech.edu/ood/guide/>.
7. PACE (2017). PBS Scripting Guide. *PACE Cluster Documentation*. <https://docs.pace.gatech.edu/software/PBS_script_guide/>.
8. PACE (2017). How to Use Local Scratch Storage. *PACE Cluster Documentation.* <https://docs.pace.gatech.edu/storage/localTmp/>.
9. PACE (2017). Helpful Commands. *PACE Cluster Documentation*. <https://docs.pace.gatech.edu/gettingStarted/commands/>.

“This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.”

# **Appendix**

#PBS -N exampleScript # name of job

#PBS -A [Account] # account to which job is charged, ex: GT-gburdell3

#PBS -l nodes=1:ppn=2 # resources allocated, 1 node 2 processors

#PBS -l pmem=2gb # memory per core

#PBS -l walltime=15:00 # job will run at most 15 min

#PBS -q inferno # job is submitted to inferno queue

#PBS -j oe # output and error is combined into the same file

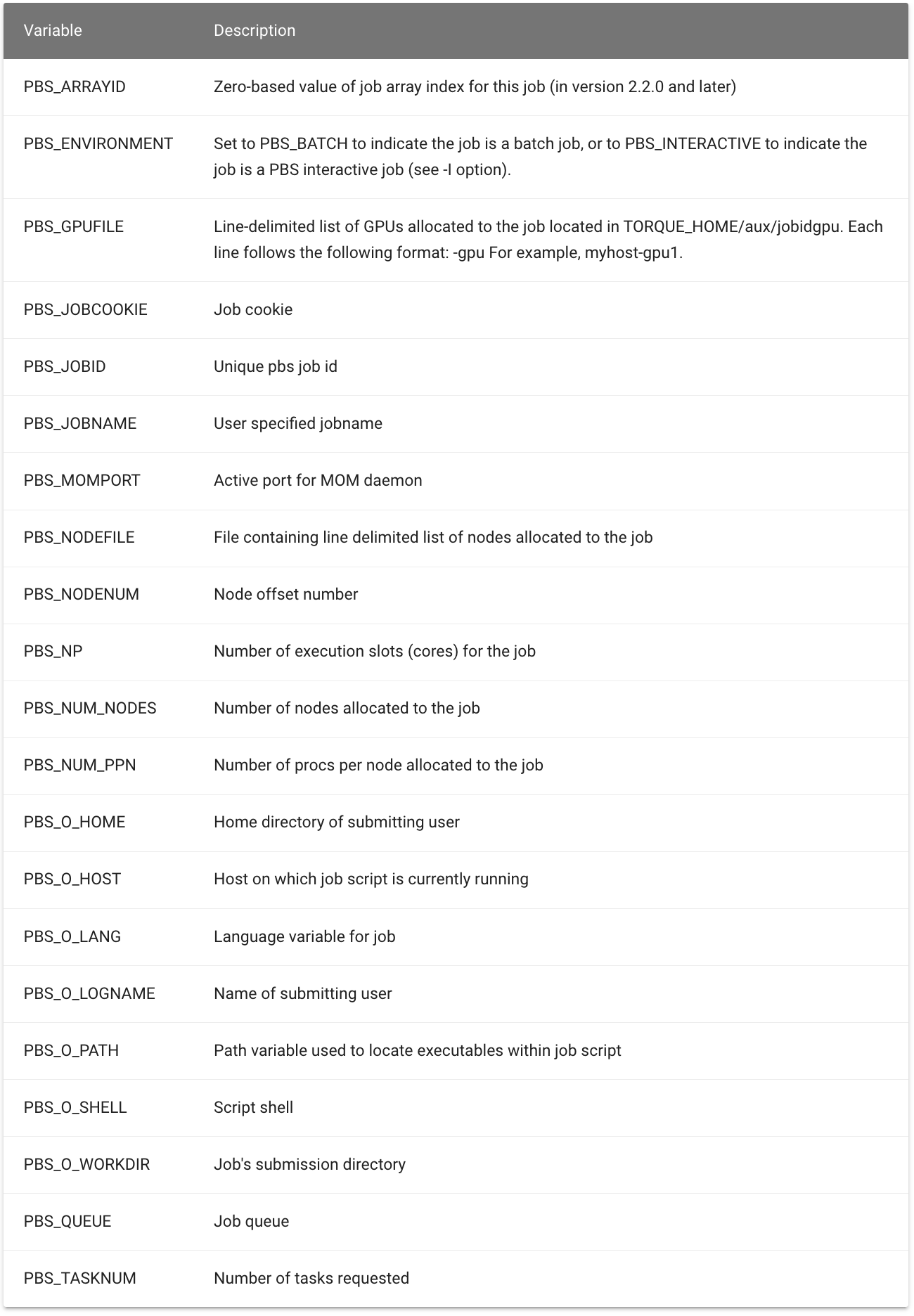
#PBS -o gettingStarted.out # output file is named gettingStarted.out

# computation starts here

cd $PBS\_O\_WORKDIR # changes into directory where script was submitted from

echo "Started on `/bin/hostname`" # prints name of compute node job was started on

**Figure 1. Example PBS Script**. Available for use and redistribution when writing scripts [7].



**Figure 2. Examples of Batch Job Environmental Variables [7].**