**PBS Job Manual**

*Author: Daniel Elbich*

*Contact: delbich10@gmail.com*

*Getting Started*

PBS jobs are lines of code or scripts which are sent to the batch portion of the cluster. This cluster has dedicated resources for high computation tasks and can be faster than running code on the interactive node (see ACIManual). The only way to interact with this portion is through the command line. This can be done by either connecting to aci-b separately from your local machine or by logging into aci-i and submitting a job through the terminal there.

*Required Flags of a PBS Script*

The PBS system has many different options and flags to include (see below), but there are only a few that are required to run a job successfully.

Allocation – The name of the computing allocation you plan to use. Most PIs purchase dedicated resources for their lab to use to run PBS jobs

Nodes & Processors – The requested nodes and processors to be used for this job

Walltime – The requested time the job will run. If the actual processing exceeds this value the job will be canceled upon reaching the walltime.

Memory – The requested amount of RAM to dedicated

Following this you may enter in your actual code you would like to run. This could be actual code, like a for-loop, or a script to run over a number of subjects. But these are the basics – what resources you are using, what amount to use, and what to run. An example would look something like this (in the order above):

#!/bin/bash

#PBS -A de123\_a\_b\_s\_e\_default #Allocation

#PBS -l nodes=1:ppn=4 #Nodes & Processors per node

#PBS -l walltime=36:00:00 #Walltime in hours

#PBS -l pmem=4gb #Memory/RAM

sh myScriptHere.sh

Finally, to submit the script to the batch system, in terminal you enter:

qsub PBS\_Script.txt

which should submit the script under your user ID. To check the status of your job(s), you enter in terminal:

qstat -u USERNAME

where 'USERNAME' is your user ID. This will return terminal output with the name of all jobs, what allocation they are submitted to, requests nodes and memory, walltime, current running time, and the status of the job (e.g. queued up, running). Importantly, without the '-u' flag you will get the status of all jobs for all users on the system and would have to search through them for yours.

Now as with all computing you can increase the numbers of nodes, ppn, and pmem and the scripts will run more quickly. However there are 2 things to consider here. First of which is what allocation you are using. Most reservations have some limitation on the resources (as more costs more money) and you will have to work within the bounds of this. Second, your job may be stuck in the queue waiting until the amount of resources you request become available. If multiple people are running jobs & using resources, your job can be stuck waiting until resources become free. You will need to do some experimenting on how long your job should take and what resources you have available to find the right balance.

*Optional Flags of a PBS Script*

There are a number of optional flags in the PBS system that can be incredibly useful depending on your needs. The following are possible flags you can include (written in PBS notation):

#PBS -N jobname #Name of the submitted job. Max 15 characters!

#PBS -j oe #Save output & error logs to directory PBS script was submitted from

#PBS -o #Separate name (or save path) for output log

#PBS -e #Separate name (or save path) for error log

#PBS -m abe #Sends email notification for job status. Add any combination of a (aborts), b (begins), and e (ends) for particular status. The 'n' will only send an email if it aborts

#PBS -M [abc123@gmail.com](mailto:abc123@gmail.com) #Email address to send notifications

#PBS -S /bin/shell #Sets the type of shell used for code. Normally avoided unless code is not written in bash (standard)

#PBS -a [YYYY][MM][DD]hhmm #Delay running the job until specific time (military time). A value of 1700 would run the job at 5PM

#PBS -W depend=afterany:'jobid' #Delay running job until 'jobid' is completed