



# Batch Processing & Queuing

## Introduction

For almost all general purpose HPC systems intended to serve the computational needs of a number of projects and/or individuals, batch processing is the dominant mode of interaction between the user and system. This is in contrast to interactive processing where the user directly provides input and receives output from the computer in real time. The benefits of batch usage are that it provides uncontested access to resources, allows work to be queued up and completed without human intervention and can enable the system to be used in a more efficient manner overall by keeping it busy with jobs of different sizes and durations.


The typical way a user will interact with compute resources managed by a batch system is as follows:

**#1** Write a job script which describes the resources required (e.g how many CPUs and for how long), instructions such as where to print standard out and error, and the commands to run once the job starts.

**#2** Submit the job to the batch processing system which will then start the job once the requested resources are available and all policy conditions have been met. The user can query this system for information on resources available and other jobs in the system along with the state of their own jobs.

**#3** Once the job completes, the user will see all results as well as output that would normally appear on screen in previously specified files.

batch processing & queuing	
Introduction & rationale	1
Writing & submitting batch jobs	2
Monitoring	3
Scheduling	3



In short, the compute systems at ICHEC are meant to be shared so we need a way to allow many users run jobs at the same time in a fair manner, while trying to balance the conflicting demands for large resources (high numbers of CPUs and/or long runtimes) with quick turnaround (short time between submitting a job and having it start) and not having idle CPUs doing nothing.

# Writing & submitting batch jobs

Every job must be submitted via a job command file or using the equivalent command line arguments to qsub.

## > qsub

In order to run jobs on the system you must submit them to a batch queue. Different queues exist for different types of job and you can choose to explicitly send your job to a chosen queue (using the #PBS -q directive or qsub -q command) or you can let the batch system fit the job into an appropriate queue for you based on your specified CPU and walltime requirements.

To see what queues are available use the qstat -q command and the information on scheduling policy on the ICHEC website. Note that not all queues listed by qstat -q are available to users and that the Walltime and Node columns list the maximum runtime and node count for jobs in a given queue.

To submit a PBS script type: qsub scriptname.pbs

Sometimes, for debugging purposes, it can be useful to launch a shell as a batch job and get an interactive session on compute nodes where you can see immediately what happens when launching a program. In these cases, an Interactive Job can be used. For example, if I wanted to test my MPI program on 8 CPUs, I could request an interactive job for 1 hour and then be given a shell on one of Stokes' compute nodes.

## Sample PBS script for Stokes:

```
# set the shell - bash shell
#!/bin/bash
# Tell pbs to use XX nodes and YY
# processors per node, this is fixed at 8
#PBS -l nodes=4:ppn=8
# resource limits: Wallclock time ([[h:]m:]s)
#PBS -l walltime=1:00:00
# To have a record of account information
# for a particular project
# An example project_name is iclif002
#PBS -A iclif002
# join the standard output and error files
#PBS -j oe
# send me email when job begins (b), ends (e)
# and/or aborts (a)
#PBS -m bea
# send it to this address
#PBS -M me@my_email.ie
# inherit the current environment
#(necessary if pre-loading modules for example)
#PBS -V

# This job's working directory
# echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
# For a MPI run on the Stokes system
mpiexec ./job
```

## Interactive session on Stokes

```
user@stokes1:~> module load mvapich2-gnu/1.2pl
user@stokes1:~> mpif77 -o hello -ffree-form hello_mpi.f
user@stokes1:~> qsub -I -l nodes=1:ppn=8,walltime=1:00:00 -V -A
iclif002
qsub: waiting for job 41224.stokes-svcs.ice.ichec.ie to start
qsub: job 41224.stokes-svcs.ice.ichec.ie ready
user@r2i1n7:~> mpiexec ./hello
node 2 :Hello, world
node 3 :Hello, world
node 4 :Hello, world
node 5 :Hello, world
node 6 :Hello, world
node 7 :Hello, world
node 1 :Hello, world
node 0 :Hello, world
user@r2i1n7:~>
```

Note however that this method should only be used for debugging and not for production runs as network breaks or timeouts will kill the job. Also, please exit the shell when you are no longer using the interactive session so that the resources can be released for other users.



The `showq` command displays information on the current status of jobs.

```
> showq
```

To cancel a job you should use the `canceljob` command.

```
> canceljob JOBID
```

## Monitoring Jobs

Frequently Used Batch System Commands	
<code>qsub SUBMIT_SCRIPT</code>	submit jobscript to PBS
<code>qsub -l</code>	submit an interactive-batch job
<code>qsub -q queue</code>	submit a job directly to specified queue
<code>qstat -q</code>	list all queues on system
<code>qstat -Q</code>	list queue limits for all queues
<code>showq</code>	list all running, queued and blocked jobs
<code>showq -u userid</code>	list all jobs owned by user userid
<code>showq -r</code>	list all running jobs
<code>mybalance</code>	list the balance in CPU hours for each project you are a member
<code>qstat -f jobid</code>	list all information known about specified job
<code>canceljob JOBID</code>	delete job jobid
<code>qalter JOBID</code>	modify the attributes of the job or jobs specified by JOBID

## How Scheduling Works

Once a job is submitted to a queue, the scheduling software examines the total resources available, the resources in use by running jobs and the resources requested by other queued jobs and makes a number of decisions:

**#1** If there are sufficient resources available and no limits have been exceeded by the owner of the job (such as total number of running jobs owned by that user or total number of CPUs in use by their other jobs) then the scheduler will trigger the start of that job.

**#2** If the job can not start immediately then it is placed in a queue along with others. This is not, however, a FIFO (First In, First Out) queue and the order jobs are listed with `showq` does not indicate the order in which they will start. Also, different queues will have different resources available to them so one can only compare jobs within the same queue. The two main scheduling components which determine when and in what order a number of queued jobs will start in are (a) Backfill and (b) Priority.

**#3** Backfill seeks to keep a system busy by filling short and/or small jobs in scheduling spaces where bigger jobs wouldn't fit. For example, if a 100 CPU system has an 80 CPU job running which won't finish for 5 hours then backfill will start a 6 hour, 10 CPU job in advance of a 1 hour, 30 CPU job. As such, smaller and/or shorter jobs have a greater chance of starting early due to backfill.

**#4** The Priority of a queued job is calculated by adding together a number of components, some of which are mentioned below. Higher priority jobs are more insulated against being backfilled and when they reach the status of being the highest priority job, they essentially get resources reserved for them so that they will start as soon as possible. The priority of a job is calculated as a weighted sum of components, including the following:

- Queue time: jobs accrue priority while waiting in the queue
- Fair share: targets are used to apply a fair level of priority across projects by taking historical usage into account. Projects which haven't used the system much will automatically gain a higher priority than projects which have used a lot of resources over the past month.
- Expansion factor: this is used to modify job priority based on requested walltime. Shorter jobs are given extra priority.
- Explicit priorities set to boost certain queues, projects or other attribute classes.

