Batch Systems

Running your jobs on an HPC machine

Outline

- What is a batch system?
- Why do we need them?
- How do I use a batch system to run my jobs?
  - Concepts
  - Resource scheduling and job execution
  - Job submission scripts
  - Interactive jobs
- Scheduling
- Best practice
- Common batch systems
  - Converting between different batch systems

Batch Systems

What are they and why do we need them?

What is a batch system?

- Mechanism to control access by many users to shared computing resources
- Queuing / scheduling system for users' jobs
- Manages the reservation of resources and job execution
- Allows users to "fire and forget" large, long calculations or many jobs ("production runs")

Why do we need a batch system?

- Ensure all users get a fair share of compute resources (demand usually exceeds supply)
- To ensure the machine is utilised as efficiently as possible
- To track usage - for accounting and budget control
- To mediate access to other resources e.g. software licences

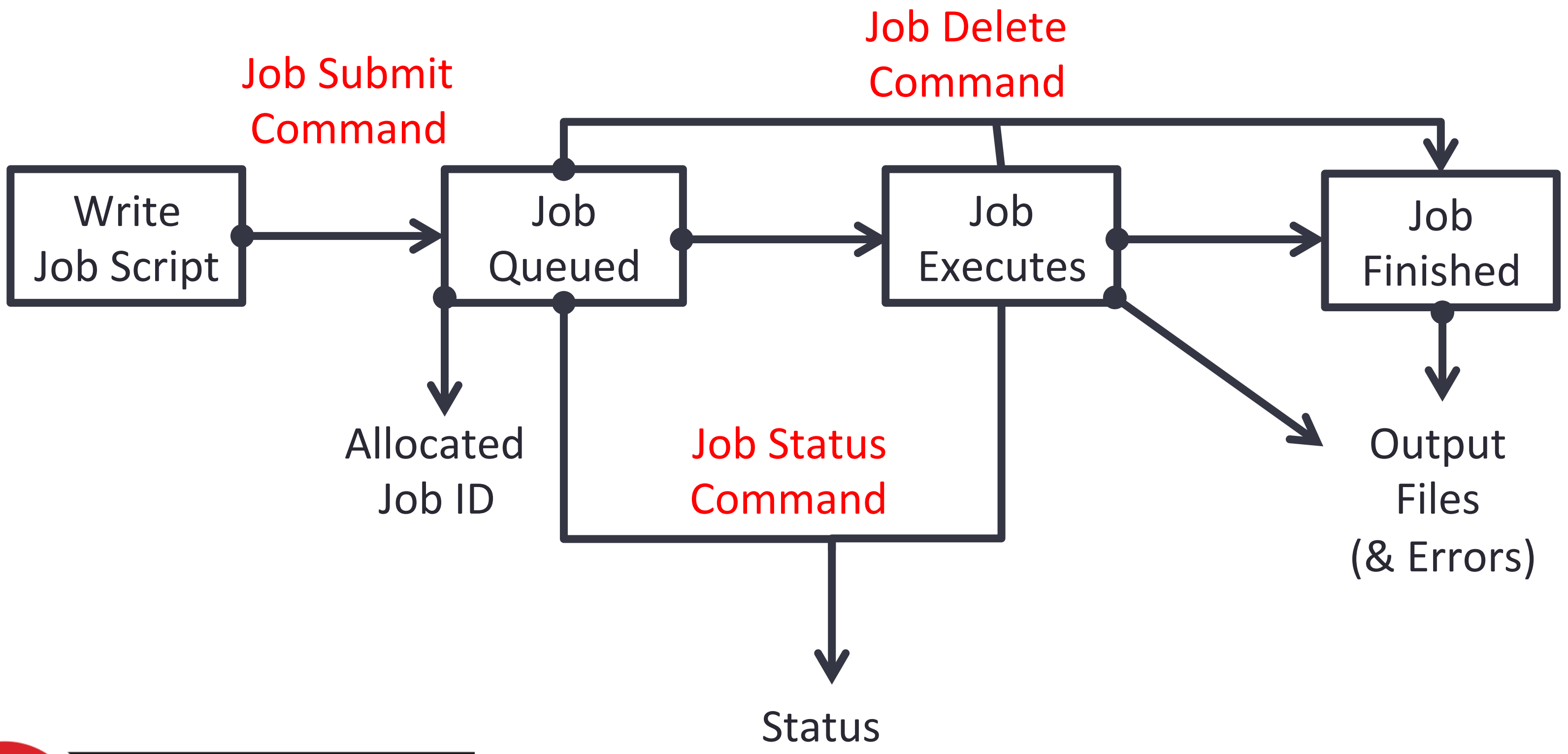
How to use a batch system

1. Set up a job, consisting of:
   - Commands that run one or more calculations / simulations
   - Specification of compute resources needed to do this

2. Submit your job to the batch system
   - Job is placed in a queue by the scheduler
   - Will be executed when there is space and time on the machine
   - Job runs until it finishes successfully, is terminated due to errors, or exceeds a time limit

3. Examine outputs and any error messages

Batch system flow



Resource scheduling & job execution

- When you submit a job to a batch system you specify the resources it requires (number of nodes / cores, job time, etc.)

- The batch system schedules a block of resources that meet these requirements to become available for your job to use

- When it runs your job can use these resources however it likes (specified in advance in your job script):
  - Run a single calculation / simulation that spans all cores and full time
  - Run multiple shorter calculations / simulations in sequence
  - Run multiple smaller calculations / simulations running in parallel for the full time

Batch system concepts

- Queue – a logical scheduling category that may correspond to a portion of the machine:
  - Different time constraints
  - Nodes with special features such as large memory, different processor architecture or accelerators such as GPUs, etc.
  - Nodes reserved for access by a subset of users (e.g. for training)
  - Generally have a small number of defined queues
  - Jobs contend for resources within the queue in which they sit

On ARCHER:

- "standard" queue (24 hour limit, no limit on number of nodes)
- "short" queue (max 20 minutes & 8 nodes, weekdays 09:00-17:00 only)

Batch system concepts

- Priority – numerical ranking of a job by the scheduler that influences how soon it will start (higher priority more likely to start sooner)

- Account name / budget code – identifier used to charge (£) time used
  - Jobs may be rejected when you submit with insufficient budget

- Walltime – the time a job takes (or is expected to take)

Batch system commands & job states

	PBS (ARCHER)	SLURM
Job submit command	qsub myjob.pbs	sbatch myjob_sbatch
Job status command	qstat –u $USER	squeue –u $USER
Job delete command	qdel #########	scancel #########

PBS job state (ARCHER)	Meaning
Q	The job is *queued* and waiting to start
R	The job is currently *running*
E	The job is currently *exiting*
H	The job is *held* and not eligible to run

Parallel application launcher commands

Use these commands inside a job script to launch a parallel executable

Parallel application launcher commands	
aprun –n 48 –N 12 –d 2 my_program	(ARCHER)
mpirun –ppn 12 –np 48 my_program	
mpiexec –n 48 my_program	

Job submission scripts

PBS example:

```
#!/bin/bash -login
#PBS -N Weather1
#PBS -l select=200
#PBS -l walltime=1:00:00
#PBS -q short
cd $PBS_O_WORKDIR
aprun -n 4800 weathersim
```

Linux shell to run job script in

Job name

Number of nodes requested

Requested job duration

Queue to submit job to

Changing to directory to run in

Parallel job launcher

Number of parallel instances of program to launch

Program name

Job submission scripts

SLURM example:

```
#!/bin/bash
#SBATCH -J Weather1
#SBATCH --nodes=2
#SBATCH --time=12:00:00
#SBATCH --ntasks=24
#SBATCH -p tesla
mpirun -np 24 weathersim
```

Linux shell to run job script in

Job name

Number of nodes requested

Requested job duration

Number of parallel tasks

Queue to submit job to (GPU queue)

Parallel job launcher

Number of parallel instances of program to launch

Program name

Interactive jobs

- Most HPC machines allow both batch and interactive jobs

- **Batch jobs** are non-interactive.
  - You write a *job submission script* to run your job
  - Jobs run without user intervention and you collect results at the end
- **Interactive jobs** allow you to use compute resources interactively
  - For testing, debugging/profiling, software development work
  - For visualisation and data analysis
- How these are set up and charged varies from machine to machine

Interactive jobs

- If using the same compute resource as batch jobs then need to request interactive jobs from the batch scheduler
  - Use same resource request variables as batch jobs (duration, size)
  - Wait until job runs to get an interactive session
  - Within interactive session run serial code or parallel programs using parallel launcher as for batch jobs
- May have a small part of the HPC machine dedicated to interactive jobs
  - Typically for visualisation & postprocessing / data analysis
  - May bypass the batch scheduler for instant access
  - May be limited in performance, available libraries, parallelism, etc.

Scheduling

- Complex scheduling algorithms try to run many jobs of different sizes on system to ensure maximum utilisation and minimum wait time

- Batch schedulers can implement scheduling policy that varies from machine to machine by allowing control over the relative importance to job prioritisation of:
  - Waiting times
  - Large vs small jobs
  - Long vs short jobs
  - Power consumption

Scheduling

- Backfilling:
  - Assign all jobs priority according to policy & scheduling algorithm
  - Starting with highest priority job, run each lesser priority job that can run with current free resources
  - For the highest priority job *A* that can not currently run, calculate when the required resources will become available and schedule job *A* to run at this time.
  - Until such time, run any less high priority jobs that will complete before job *A* starts and for which sufficient resources are currently available
  - This “fills gaps” and improves resource utilisation

- Active area of research

http://archer.ac.uk/status/

- How long until my job executes?

Best practice

- Run short tests using interactive jobs if possible
- Once you are happy the setup works write a short test job script and submit it to the batch system
- Finally, produce scripts for full production runs
- Remember you have the full functionality of the Linux command line (bash or other) available in scripts
  - This allows for sophisticated scripts if you need them
  - Can automate a lot of tedious data analysis and transformation
  - …be careful to test when moving, copying deleting important data – it is very easy to lose the results of a large simulation due to a typo (or unforeseen error) in a script

Migrating

Changing your scripts from one batch system to another

Batch systems

- PBS (on ARCHER), Torque
- Grid Engine
- SLURM
- LSF – IBM Systems
- LoadLeveller – IBM Systems

Conversion

- Usually need to change the batch system options
- Sometimes need to change the commands in the script
  - Particularly to different paths
  - Usually the order (logic) of the commands remains the same
- There are some utilities that can help
  - Bolt – from EPCC, generates job submission scripts for a variety of batch systems/HPC resources: https://github.com/aturner-epcc/bolt