Batch Systems & Parallel Application Launchers

Running your jobs on an HPC machine

Partners

Funding

Horizon 2020
European Union Funding
for Research & Innovation

Reusing this material

Outline

- What are batch systems and why do we need them?
- How do I use a batch system to run my jobs?
  - Concepts
  - Job submission scripts
  - Interactive jobs
- Scheduling
- Parallel application launchers
- Best practice

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 on these resources
- Allows users to "fire and forget" large, long calculations or many jobs ("production runs")

Why do we need a batch system?

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

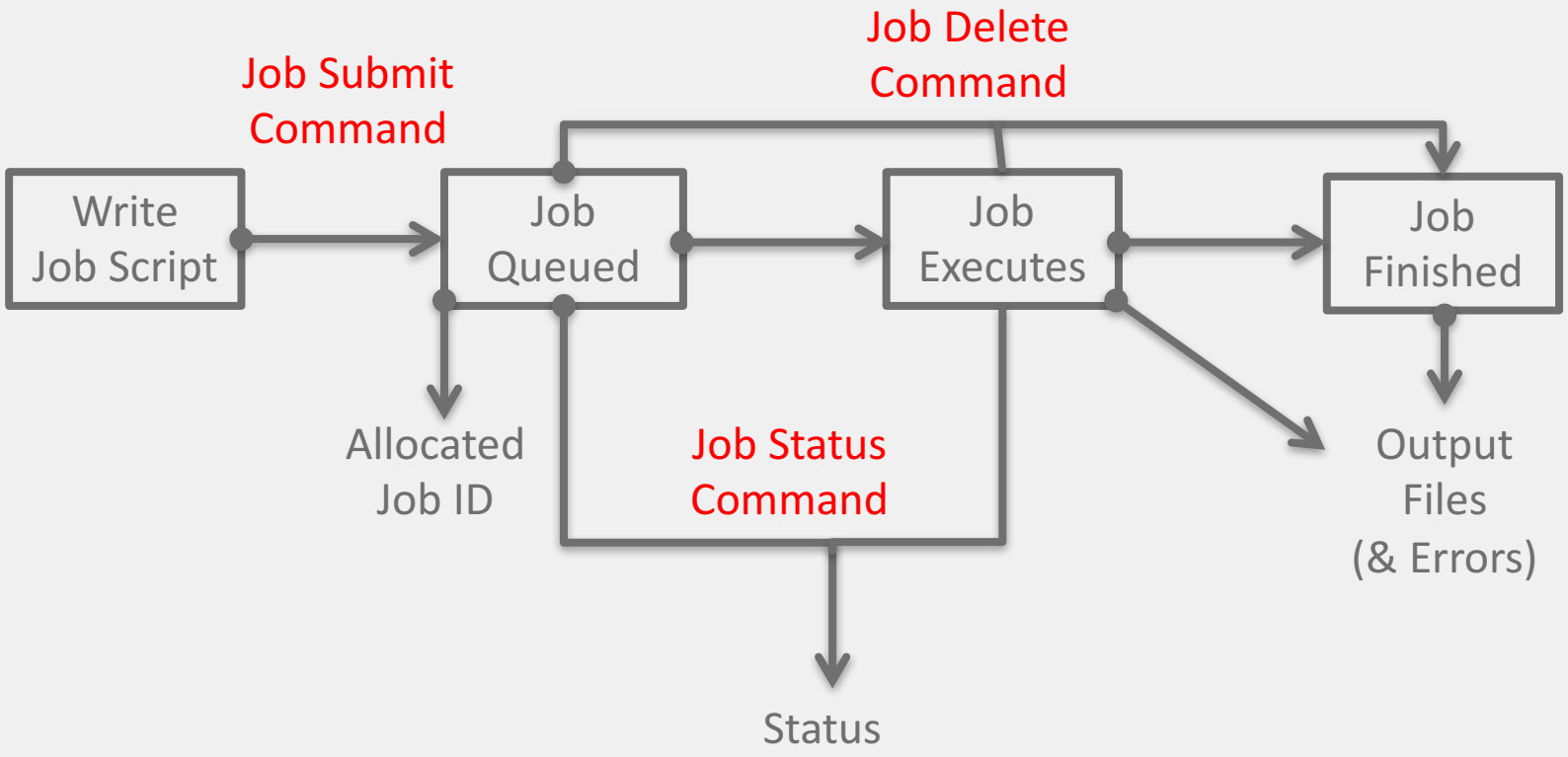
How to use a batch system

1. Write a job script specifying
   - Compute resources needed
   - Commands to run one or more calculations / simulations / analyses

2. Submit your job to the batch system
   - Job 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, or
     - is terminated due to errors, or
     - exceeds a time limit

3. Examine outputs and any error messages

Batch system flow



Batch system concepts

- Queue - logical scheduling category, can correspond to:
  - Different time constraints
  - Special feature nodes (large memory, 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 walltime limit, no limit on number of nodes)
  - “short” queue (max 20 minutes & 8 nodes, weekdays 08:00-20:00 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

Common Batch systems

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

Job script example – PBS on ARCHER

```
#!/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

Program name

Parallel application launcher

Number of parallel instances of program to launch

Job script example - SLURM

```
#!/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)

Program name

Parallel application launcher

Number of parallel instances of program to launch

Interactive jobs

- Many HPC machines allow both batch and interactive jobs
  - Interactive jobs allow you to run on compute nodes directly from the command line – no need for a script
  - Can be useful for testing, debugging, profiling, setting up new simulations, software development, visualisation and data analysis
- Set up and charging varies from machine to machine
- Common for HPC machines to have some nodes dedicated to interactive work
  - May bypass the batch scheduler for instant access
  - May be limited in performance, available libraries, parallelism, etc.

Good practice

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

Scheduling

a multidimensional game of Tetris..

Resource scheduling & job execution

- Specify the resources your job requires
  - number of nodes / cores
  - job time
  - etc.
- Batch system schedules these resources to become available
- Once started, job can use these resources however it likes:
  - a single run on all requested cores for the full time
  - multiple shorter runs on all cores, one after the other
  - multiple smaller runs in parallel for the full time, each on a subset of the requested cores

Scheduling

- Complex algorithms try to schedule jobs against resources as efficiently as possible according to a configured scheduling policy

- Scheduling policy varies from machine to machine, and controls the relative importance in job prioritisation of:
  - Waiting times
  - Large vs small jobs
  - Long vs short jobs
  - Power consumption
  - High overall machine utilisation

Scheduling

- Backfilling:
  - Assign all jobs priority according to scheduling policy
  - Starting with the highest priority job, if required resoures are available then run it. Then check the next highest priority job on the list, etc.
  - For the highest priority job J that can *not* currently run, schedule it to run as soon as needed resources are due to become available
  - Schedule any lower priority jobs that can finish before J is due to start
  - This fills usage gaps and improves resource utilisation
- Active area of research
- How long until jobs typically run once queued?
  - See http://archer.ac.uk/status/#heatmap

Parallel Application Launchers

Launching parallel applications

- To run an MPI-parallel applications, need to use a parallel application launcher (mpirun, mpiexec, or aprun)
  - provided by an MPI library
- Uses information provided by the batch system
  - Identities of nodes available for job
- Launches the desired number of processes, each running an independent instance of the application
  - Sets up their environment so they can exchange messages
- Used to control number of processes per node
  - Fewer processes than cores can mean some cores unused but mean a problem fits into memory → still faster solution
  - Hybrid (MPI + threads) applications can use “spare” cores with threads

Launching parallel applications

- Advanced options available to tie processes to cores
  - For better performance
- Threaded applications don't need mpirun etc.
  - Can control how threads are are tied to cores in other ways
- For hybrid (MPI + threaded) applications, application launcher can control thread as well as process placement
  - c.f. Cray's parallel application launcher, aprun

Parallel application launcher commands	
mpirun –n 48 my_app	Launch 48 instances, default number per node
mpiexec –n 48 --npernode 6 my_app	Launch 48 instances, 6 per node (needs 8 nodes)
aprun –n 2 –d 12 my_app	Launch 2 instances, spaced 12 cores apart to allow room for threads (Cray/ARCHER)