

Compute Cluster Workshop



Mike Smith

Introduction

What is a compute cluster?

- A bunch of individual machines (**nodes**) tied together
 - Nodes are often heterogeneous
 - No. of CPU cores, Memory, Disk space, ...
- Special software is used to represent those machines as a pool of shared resources
- This software gives you ability to ask for a chunk of this pool to run your software

What is a compute cluster?

- Tailored to batch processing (=jobs)
- Interactive use possible
- You don't care on which machine your job is running
- If you do, you can ask for specific resources to be allocated to you

High **P**erformance **C**omputing (HPC): the (effective) use of multiple computers to do things you couldn't do on a single machine.

When is HPC useful?

- When you want to get results faster than what your laptop can offer
 - Compute Intensive: Task requiring a large amount of computation
 - e.g. more rigorous sequence alignment
 - Memory Intensive: Task requiring a large amount of memory
 - e.g. scaling up from bacterial to human genome
 - Data Intensive: Task involved operating on a large amount of data
 - e.g. 50 human genomes

Where to find help

- Training like this one
- Wiki: <https://wiki.embl.de/cluster/>
- chat.embl.org #cluster
- itsupport@embl.de
- clusterNG mailing list
- Meetings as needed
 - When there are new things to announce and explain
- Bio-IT meetings, Coding Club

Jobs & Scheduling

How do I work with a cluster?

- Typically interact with a frontend (head) node
- A **job scheduler** manages where and when tasks are run
 - There are many options available e.g. LSF, Torque, Slurm, Condor, Univa Grid Engine
- Matches job requirements with available resources
- If no slots are available a job will wait until resources are available

Slurm

- “**S**imple **L**inux **U**tility for **R**esource **M**anagement”
- One of the most popular HPC schedulers
- All fancy features are first developed for Slurm
- Currently running 17.11
- Regular updates for bug fixes and new features

How do I connect to the cluster?

- Connect to the cluster frontend node via `ssh`

```
ssh <username>@login.cluster.embl.de
```



This is the frontend node

Obtaining example program

- Use git to download

```
git clone https://git.embl.de/msmith/embl_hpc.git
```

How do I run a program on the cluster?

- Don't run anything on the frontend node! (except this one time...)

```
hostname
```

- Our first **job**

```
srun hostname
```

job: a resource allocation & the **steps** run within it (just one in above)

step: single task run by scheduler

srun submits a job step to the cluster

Training reservation

- You only need to use this during our session today

```
srun --reservation=training hostname
```

- Isolates us from the rest of EMBL

Reservation: collection of resources reserved for particular users/groups/time period

Example program

- Program should be present in the *'exercises'* directory
- Takes two arguments
 - -t Time to wait in seconds
 - -m Amount of memory to use in MB

```
./hpc_example -t 10 -m 100
```

**(Remember not to run
on the login node!)**

- Prints arguments to screen -> creates list -> waits -> prints memory usage -> exits

Submit example program

```
srun --reservation=training \  
./hpc_example -t 10 -m 100
```

Submitting Example program

- **srun** is not convenient, use **sbatch** to run in background
- We need to use a script - `batch_job.sh`

```
sbatch --reservation=training \  
batch_job.sh
```

sbatch submits a **job script** to the cluster

job script: simple script that combines resource requests and job steps

Viewing jobs

```
queue
```

- We can filter the list to be more specific

```
queue --user <username>  
queue --reservation training
```

`queue` lists current jobs

Examining output

- Default output is a file based on the JobID e.g `slurm-15273607.out`
- You can change this
- Use `srun/sbatch --output=output.txt`

```
sbatch --output=<output.txt> \  
      --reservation=training \  
      ./batch_job.sh
```

- Append to a file with `--open-mode=append`

Options in the batch script

- All options can also go in the script itself
- Start option lines with `#SBATCH`

Quick recap

- Don't run things on the head node!
- Submit jobs using `sbatch` (and `srun`)
- View status of jobs with `squeue`
- Edit the location of output with `--output=<filename>`
- Options can be at command line or in script with `#SBATCH`

Questions?

Experiment with settings

- We modify our script to accept arguments
- Submit several jobs, try using more memory

```
sbatch --reservation=training \  
./batch_job.sh 20 ???
```

reminder: the 2nd option controls the maximum memory that the job will use

Our node has 256GB or 256,000MB

Resources management

Reserving additional resources

- Sharing resources between users is a key function of the job scheduler
- Jobs may be killed or slow down if they try to use more than their allocated share
- Use `scontrol` to view the cluster configuration & default values

```
scontrol show partition
```

`scontrol show` configuration of the cluster

partition: collection of resources with common attributes (also known as a queue)

Requesting additional resources

- Sharing resources between users is a key function of the job scheduler
- Jobs may be killed or slow down if they try to use more than their allocated share
- Try reserving an appropriate amount of memory

```
#SBATCH --mem=<XXX>
```

```
sbatch --mem=8200 \  
--reservation=training \  
./batch_job.sh 30 8000
```

Requesting additional resources

- Try reserving a LARGE amount of memory

```
sbatch --mem=100gb \  
      --reservation=training \  
      ./batch_job.sh 300 5000
```

- Look at the waiting jobs with `squeue -t PENDING`
- Only a small number of jobs will be allowed to run simultaneously

Requesting appropriate resources

- Understanding the compute requirements of your task is key to effective use of an HPC cluster
- Ask for too much
 - Your job will wait for a long time unnecessarily
 - Reserve resources you don't need, keeping others from using them
- Ask for too little
 - Job may be killed without finishing
 - You start using resources you haven't asked for, potentially slowing things down for everyone

Canceling unwanted jobs

- Cancel a single job

```
scancel <jobID>
```

- Cancel all jobs for a user

```
scancel -u <username>
```

Number of cores

- Many programs offer 'multi-threading' or 'multi-core'
- Make sure you request this with:

```
#SBATCH --ntasks=1
```

```
#SBATCH --cpus-per-task=8 (other integers are available)
```

- Be aware of the default behavior of the application!

Setting a time limit

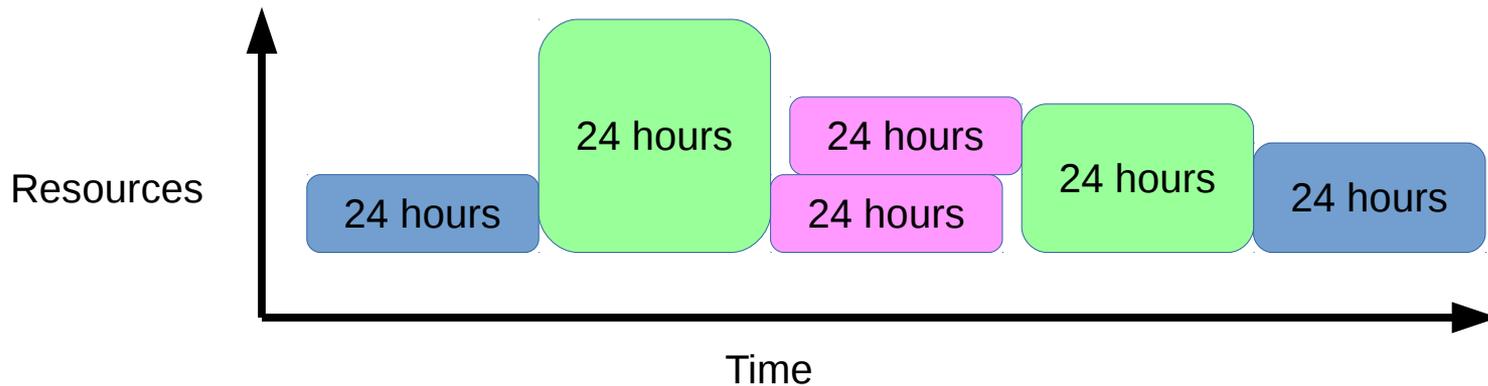
- Default time limit is 20 minutes (will be 5 minutes soon)
- Define a time limit with:

```
#SBATCH --time=<HH-DD:MM:SS>
```

```
sbatch --time=00-00:00:30 \  
--reservation=training \  
batch_job.sh 60 500
```

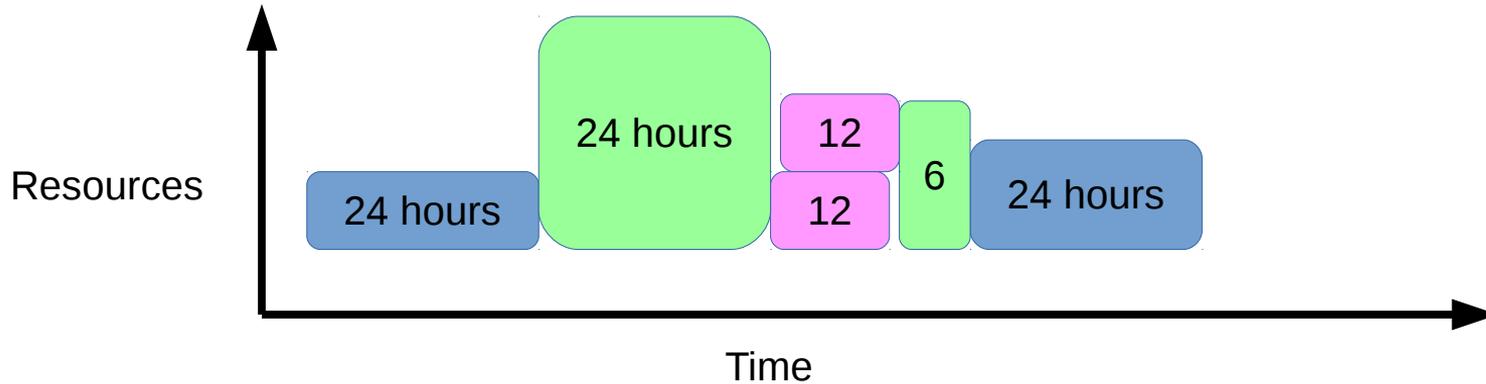
Time limits

- Providing a run time matters –
 - SLURM tries to slot short jobs into gaps
 - If every request has the same time, it can't do this



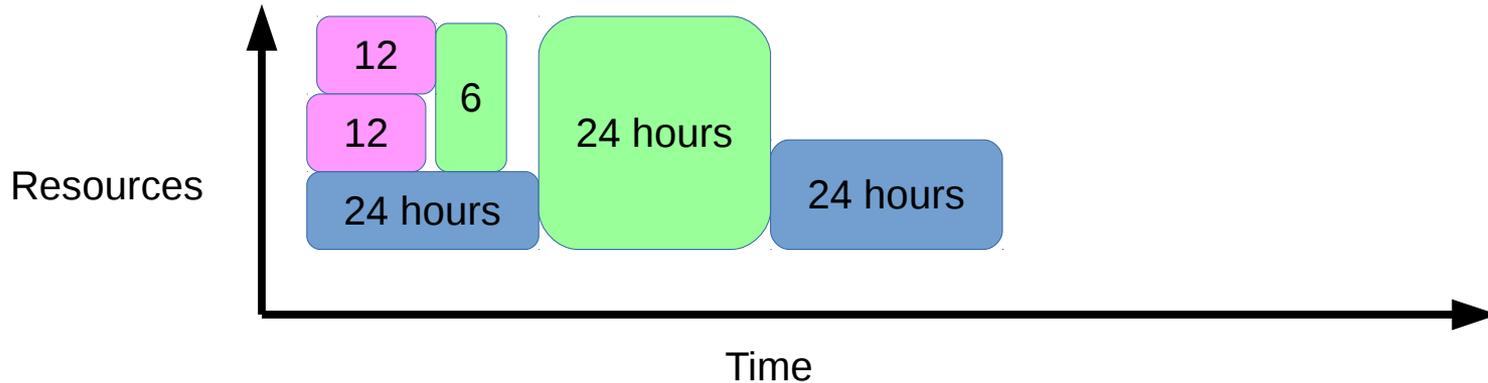
Time limits

- Default time limit is 20 minutes (will be 5 minutes soon)
- Providing a run time matters –
 - SLURM tries to slot short jobs into gaps
 - If every request has the same time, it can't do this



Time limits

- Default time limit is 20 minutes (will be 5 minutes soon)
- Providing a run time matters –
 - SLURM tries to slot short jobs into gaps
 - If every request has the same time, it can't do this



Resources summary

- Balance between asking for enough to run your job, but not too much
- Unfortunately, determining the right amount is hard
 - Try running a few **realistic** tests
 - Read manuals – often they have some guidelines
 - If it's your software, maybe you can work from the code
- Use `seff` to report efficiency of a finished job

```
seff <jobid>
```

Troubleshooting

CC-BY 2.0 <https://www.flickr.com/photos/gaetanlee/298160434/>



Job reporting

- You can get email notification of jobs finishing & details about their execution
- Use the `--mail-user=user@mail.com` option

```
sbatch --mail-user=<first.last>@embl.de \  
  --reservation=training \  
  batch_job.sh 20 500
```

- Report emails contain a lot of information
 - resource usage
 - efficiency of this usage vs what you requested

Why is my job not running?

- Slurm can tell you a reason:

```
scontrol show job <jobid>
```

- Many possible reasons:
 - Resources
 - Priority
 - Various limits

Why did my job fail?

- Use the `sacct` command to see information about recently-finished jobs

```
sacct  
sacct -j <jobid>
```

- Many possible exit codes:
 - Completed is the expected one
 - Failed
 - Timeout
 - Cancelled
 - ...

More complex jobs

Batch scripts

- Batch scripts can have more than one step
- Try modifying `batch_job.sh` to run the example program twice, with different parameters

Using software

- Most commonly-used software is provided centrally, as **modules**
- To use this software, you first need to load the corresponding module

```
module load BWA  
bwa index genome.fasta
```

`module load` add a specific software module to your working environment
module: package of pre-installed software, dependency-aware, optimized for hardware and environment

Using software

- Look at what modules are available with `module avail`, and search for something specific with `module spider <software>`

```
module avail
module spider samtools
```

`module avail` lists all modules (software & versions) available on the system

`module spider` search for all available modules (versions) for a particular program

Data Movement

- Always try to move data as close to compute as possible
- Nodes have >250GB of local \$TMPDIR, use it:
 - `--tmp=50gb` (select only nodes with at least 50GB of free space)
 - `--gres=tmp:50gb` (declare your job will use 50GB of \$TMPDIR)
- Copy your data to \$TMPDIR as first step in your job
- Copy your results from \$TMPDIR as last step of your job
- If you need more, copy your data to /scratch
 - Visible from all nodes
 - Each job gets a dedicated \$SCRATCHDIR

Real world example

E.coli sequence alignment

- Look at `exercises/bwa/bwa_batch.sh`
- Multi-step job with data movement, software loading and resource requirements

Conclusions

- Head node is for job submission only
- Understanding the requirements of your jobs is key
 - This can be hard :(
 - Doesn't need to be super precise, reasonable estimates are fine
- Remember the cluster is shared between all EMBL users – this includes you!
- Don't be afraid to try / ask for advice if you need it