HPC Practical

1 General commands

1.1 Obtaining documentation for commands

You can obtain documentation for (most) commands using the `man` (for "manual") command: `man <command>`

For example, try the following:

```
man sbatch
man squeue
man scancel
```

1.2 Submitting jobs

The script *scripts/myscript.sh* generates 100 random numbers using a simple `for` loop:

```
#!/bin/bash

for i in {1..100}; do
  echo $RANDOM >> SomeRandomNumbers.txt
done

sort SomeRandomNumbers.txt
```

When submitting a script to the cluster we should specify a partition, time limit, memory allocation and number of cores. You can also set additional parameters as shown, such as the job name and output files.

You can submit your job with the command:

```
sbatch -p general \
       -N 1 \
       -n 1 \
       --mem 100M \
       -t 1:00:00 \
       -o myscript.out \
       -e myscript.err \
       scripts/myscript.sh
```

Examine the content of `myscript.out` and `myscript.err`.

`myscript.out` contains the output from the `sort` command as this was not redirected to a file. `myscript.err` is empty as there were not errors when running the script.

A better approach is shown in `scripts/myscript2.sh` where the resource allocation is done inside the shell script:

```
#!/bin/bash
#SBATCH -p general # partition (queue)
#SBATCH -N 1 # number of nodes
#SBATCH -n 1 # number of cores
#SBATCH --mem 100 # memory pool for all cores
#SBATCH -t 0-2:00 # time (D-HH:MM)
#SBATCH -o myscript.out # STDOUT
```

```
#SBATCH -e myscript.err # STDERR


echo "Generating 100 random numbers"

for i in {1..100}; do
  echo $RANDOM >> SomeRandomNumbers2.txt
done

sort SomeRandomnumbers2.txt
```

Now you can submit your job with the command, as a simple command:

```
sbatch scripts/myscript2.sh
```

Again, examine the content of `myscript.out` and `myscript.err`. This time, we have an error - what happened?

If you want to test your job and find out when your job is estimated to run use (note this does not actually submit the job):

```
sbatch --test-only myscript.sh
```

2 Other commands on SLURM

2.1 Information on cluster status

Check available partitions and nodes, The column NODES(A/I/O/T) shows number of nodes in the states "allocated/idle/other/total" for each SLURM partition.

```
sinfo -s
sinfo -l
sinfo -a
sinfo -T
```

Query configuration and limits for a specific partition:

```
scontrol show partition <partition>
```

Check on one node:

```
scontrol show node <nodeid>
```

2.2 Information on submitted jobs

List all jobs submitted jobs:

```
squeue -u
```

List all current jobs for a user:

```
squeue -u <username>
```

List all running jobs for a user:

```
squeue -u <username> -t RUNNING
```

List all pending jobs for a user:

```
squeue -u <username> -t PENDING
```

List detailed information for a job (useful for troubleshooting):

```
scontrol show jobid -dd <jobid>
```

List status info for a currently running job:

```
sstat --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID -j <jobid> --allsteps
```

2.3 Information on finished jobs

Once your job has completed, you can get additional information that was not available during the run. This includes run time, memory used, etc. To get statistics on completed jobs by jobID:

```
sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed
```

To view the same information for all jobs of a user:

```
sacct -u <username> --format=JobID,JobName,MaxRSS,Elapsed
```

Check the default account your jobs will use:

```
sacctmgr show user <username> format=user%20s,defaultaccount%30s
```

See all account associations for your username

```
sacctmgr list association where users=$USER format=account%30s,user%20s,qos%120s
```

2.4 Controlling jobs

To cancel one job:

```
scancel <jobid>
```

To cancel all the jobs for a user:

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

To cancel all the pending jobs for a user:

```
scancel -t PENDING -u <username>
```

To cancel one or more jobs by name:

```
scancel --name myJobName
```

To pause a particular job:

```
scontrol hold <jobid>
```

To resume a particular job:

```
scontrol resume <jobid>
```

To requeue (cancel and rerun) a particular job:

```
scontrol requeue <jobid>
```

Check job history:

```
sacct -X -u <username>
```

Check job history (jobid, number of nodes, list of nodes, job state and exit code) for user su01 in specified time period (10-15 November 2018)

```
sacct \
  -X \
  -u <username> \
  -o "jobid,nnodes,nodelist,state,exit" \
  -S 2018-11-10 \
  -E 2018-11-15
```

Check memory usage for the completed job with the job id:

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

3 Case study

You have received a RNASeq dataset of a human tissue sample here:

~/scratcha/hpc_training/fastq

The dataset consists of four fastq files

1. You would like to check the data quality of this data set
2. You would like to align this dataset into the human genome

The tools you will need to use are FastQC for data quality and STAR for alignment. These tools, along with many others, can be found in the bioinformatics **software** directory: **/home/bioinformatics/software**

The scripts to perform this analysis are already in the **scripts** directory of the course materials.

3.1 QC

To run the quality controls we have a script *scripts/run_fastqc.sh*:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --mem 4G
#SBATCH -t 1:00:00
#SBATCH -o fastqc_%j.out
#SBATCH -e fastqc_%j.err
#SBATCH -J fastqc

fastqFile=${1}

/home/bioinformatics/software/fastqc/fastqc-v0.11.8/fastqc ${fastqFile}

/home/bioinformatics/software/fastqc/fastqc-v0.11.8/fastqc --version
```

To submit one job for each fastq file using this script we create another script containing a for loop - *scripts/step1_qc.sh*:

```
#!/bin/bash

projectDir=~/scratcha/hpc_training
fastqDir=${projectDir}/fastq

for fastqFile in ${fastqDir}/*fastq; do
    sbatch ${projectDir}/scripts/run_fastqc.sh ${fastqFile}
done
```

To run the qc simply run this script at the command line - no need to submit this script:

```
./scripts/step1_qc.sh
```

3.2 Alignment

To run the alignment we have a script *scripts/run_star.sh*:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --mem 8G
#SBATCH -t 1:00:00
#SBATCH -o star_align_%j.out
#SBATCH -e star_align_%j.err
#SBATCH -J star_align

fastqFile=${1}
sampleName=${2}
bamDir=${3}

refDir=/mnt/scratcha/bioinformatics/reference_data/reference_genomes
starGenomeDir=${refDir}/homo_sapiens/GRCh38/star-2.7.2b

outputName=${bamDir}/${sampleName}

/home/bioinformatics/software/STAR/STAR-2.7.3a/bin/STAR \
    --runThreadN 4 \
    --genomeDir ${starGenomeDir} \
    --outSAMtype BAM Unsorted \
    --outSAMmapqUnique 60 \
    --outSAMunmapped Within KeepPairs \
    --readFilesIn ${fastqFile} \
    --outFileNamePrefix ${outputName} \

echo -n "STAR version: "
/home/bioinformatics/software/STAR/STAR-2.7.3a/bin/STAR --version
```

To submit one job for each fastq file using this script we create another script containing a for loop - *scripts/step2_align.sh*:

```
#!/bin/bash

projectDir=~/scratcha/hpc_training
fastqDir=$projectDir/fastq
```

```
bamDir=$projectDir/bam

mkdir $bamDir

for fastqFile in $fastqDir/*fastq; do
   sampleName=`basename ${fastqFile} .fastq`
   sbatch ${projectDir}/scripts/run_star.sh ${fastqFile} ${sampleName} ${bamDir}
done
```

To run the alignment:

```
./scripts/step2_align.sh
```