Submitting a batch job

The next step is to submit this batch job to the cluster.

Within the Class_Examples folder we unzipped you will find two pre-made submit scripts. One for 1 core `python_job.sh` and one for 5 cores `python_job_5_cores.sh` .

Here is how the 5 core job example looks:

```
#!/bin/bash -l

# Request 1 CPU for the job
#SBATCH --cpus-per-task=5

# Request 8GB of memory for the job
#SBATCH --mem=8GB

# Walltime (job duration)
#SBATCH --time=00:05:00

# Then finally, our code we want to execute.
time python3 invert_matrix.py
```

Note

All of the lines that begin with a #SBATCH are directives to Slurm. The meaning of the directives in the sample script are exampled in a comment line that precedes the directive. The full list of available directives is explained in the man page for the sbatch command which is available on discovery.

sbatch will copy the current shell environment and the scheduler will recreate that environment on the allocated compute node when the job starts.

Note

The job script does NOT run .bashrc or .bash_profile, and so may not have the same environment as a fresh login shell. This is important if you use aliases, or the conda system to set up your own custom version of python and sets of python packages. Since conda defines shell functions, it must be configured before you can call, e.g. conda activate my-env The simplest way to do this is for the first line of your script to be:

```
#!/bin/bash -l
```

which explicitly starts bash as a login shell

Now submit the job and check its status. We use sbatch to submit the job:

```
[john@x01 Class_Examples]$ sbatch python_job.sh
Submitted batch job 3111754
```

To see a quick view of the job status you can issue the squeue -u command. In the state (ST) field it shows you what your current job state is. Below we can see that my job is RUNNING with (R)

```
[john@x01 Class_Examples]$ squeue -u john
   JOBID    PARTITION     NAME        USER  ST     TIME  NODES NODELIST(REASON)
   3111754  standard      python_j    john  R      0:49      1 t02
```

If you would like to see detailed information of the job. You can use the `scontrol` command. Scontrol will provide information like what node my job is running on, how much walltime my job has left, and other useful information like the resources requested, and more.

```
[john@x01 Class_Examples]$ scontrol show job 3111754
JobId=3111754 JobName=python_job_5_cores.sh
   UserId=john(48374) GroupId=rc-users(480987) MCS_label=rc
   Priority=1 Nice=0 Account=rc QOS=normal
   JobState=FAILED Reason=NonZeroExitCode Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=1:0
   RunTime=00:00:01 TimeLimit=00:05:00 TimeMin=N/A
   SubmitTime=2024-11-04T12:59:58 EligibleTime=2024-11-04T12:59:58
   AccrueTime=2024-11-04T12:59:58
   StartTime=2024-11-04T13:00:05 EndTime=2024-11-04T13:00:06 Deadline=N/A
   PreemptEligibleTime=2024-11-04T13:00:05 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-11-04T13:00:05 Scheduler=Backfill
   Partition=standard AllocNode:Sid=slurm-fe01-prd.dartmouth.edu:177904
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=t02
   BatchHost=t02
   NumNodes=1 NumCPUs=5 NumTasks=1 CPUs/Task=5 ReqB:S:C:T=0:0:*:*
   ReqTRES=cpu=5,mem=8G,node=1,billing=5
   AllocTRES=cpu=5,mem=8G,node=1,billing=5
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=5 MinMemoryNode=8G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/dartfs-hpc/rc/home/p/d18014p/Class_Examples/python_job_5_cores.sh
   WorkDir=/dartfs-hpc/rc/home/p/d18014p/Class_Examples
   StdErr=/dartfs-hpc/rc/home/p/d18014p/Class_Examples/slurm-3111754.out
   StdIn=/dev/null
   StdOut=/dartfs-hpc/rc/home/p/d18014p/Class_Examples/slurm-3111754.out
   Power=
   TresPerTask=cpu:5
```

JOBID is the unique ID of the job – in this case it is 3111754. In the above example I am issuing scontrol to view information related to my job

The output file, ***slurm-3111754.out***, consists of three sections:

- A header section, ***Prologue***, which gives information such as JOBID, user name and node list.
- A body section which include user output to ***STDOUT***.
- A footer section, ***Epilogue***, which is similar to the header.
  If we use the command `cat` short for con·cat·e·nate, we can see a that the output file is exactly as we would expect it to be:

```
[john@x01 Class_Examples]$ cat slurm-3111754.out
i,mean 50 -1.09670952875e-17
i,mean 100 3.04038500105e-16
i,mean 150 1.24104227901e-17
i,mean 200 -1.60139176225e-16
i,mean 250 -1.17287454488e-16
i,mean 300 2.94829036507e-16
i,mean 350 4.66888358553e-17
i,mean 400 3.752857595e-15
i,mean 450 2.60083792553e-18
i,mean 500 2.05032635526e-16
i,mean 550 -3.80521832845e-16
i,mean 600 -3.07765049942e-17
i,mean 650 -9.39259624383e-17
i,mean 700 2.81309843854e-16
i,mean 750 -4.91502224946e-17
i,mean 800 7.35744459606e-17
i,mean 850 -5.23231103131e-18
i,mean 900 -5.52926185394e-17
i,mean 950 -3.26360319077e-16
i,mean 1000 -1.39343172417e-17
```

Here are some other useful commands to remember when interfacing with the scheduler.

sbatch sbatch submits a batch job to the queue

squeue squeue shows status of Slurm batch jobs

srun srun --pty /bin/bash runs an interactive job

sinfo sinfo show information about partitions

scontrol scontrol show job used to check the status of a running, or idle job

scancel scancel cancel job