## How to Slurm:

*Background:*

Essentially, Slurm, or Simple Linux Utility for Resource Management, is a cluster management and job scheduling system that allows one to run multiple cluster tasks while allocating resources and reporting the status of said tasks. It allows one to simultaneously run the cluster and maximize resources.

Here, cluster management refers to Slurm's ability to allocate resources and maximize the efficiency of the GPUs for the jobs that are asked of it. For example, Slurm utilizes a queue that keeps track of jobs assigned, resources being used, the status of the job, etc.

Expanding upon that, Slurm has three primary functions (from the documentation): "It allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work"; second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes; third, it arbitrates contention for resources by managing a queue of pending work."

*Architecture*:

The architecture consists of a slurmd daemon running on each node and a "central" slurmctld "running on a management node". A daemon in this context is a computer program that runs in the background, rather than something the clients can interact with. However, clients can interact using specific commands: sacct, sacctmgr, salloc, sattach, sbatch, sbcast scontrol, scrontab, sdiag, sh5util, sinfo, sprio, squeue, sreport, srun, sstat, strigger, sview.

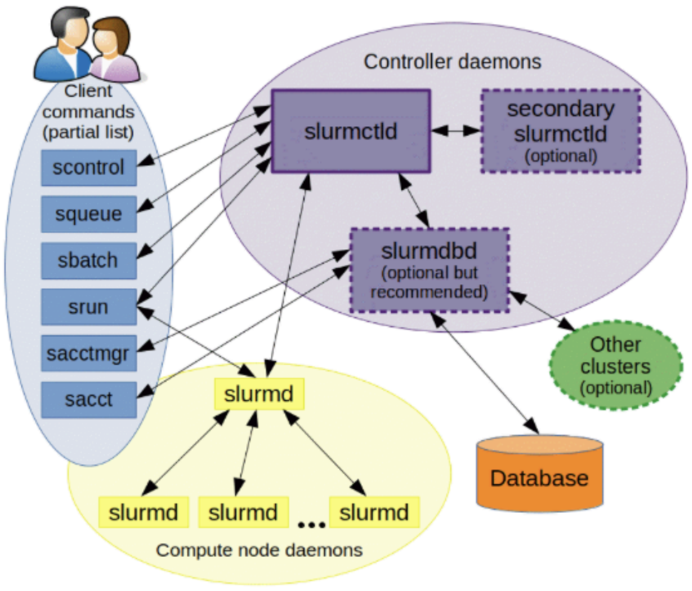


Figure 1. Slurm components

These daemons control nodes, which are the "compute resource" in Slurm, partitions, which "group nodes into logical sets", jobs, or allocations of resources assigned to a user for a specified amount of time, and job steps, or tasks within a job. The partitions are like job queues (documentation), which each have their own limits on time, people permitted to use, size of job, etc.

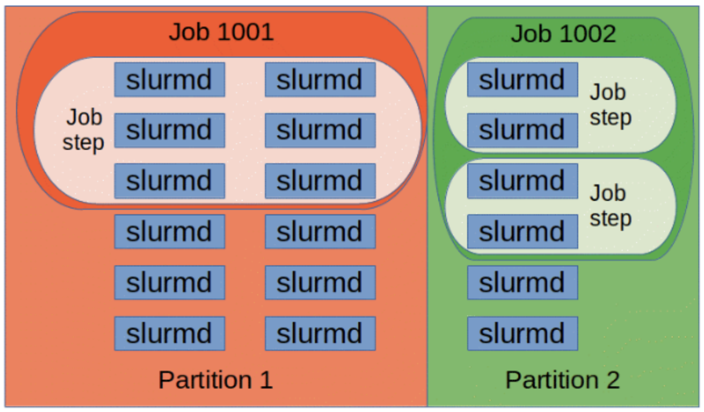
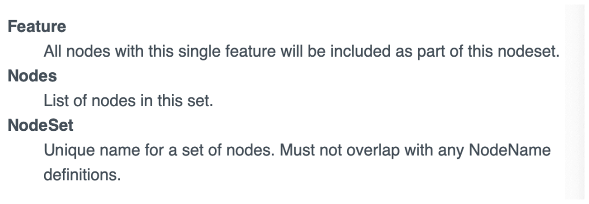


Figure 2. Slurm entities

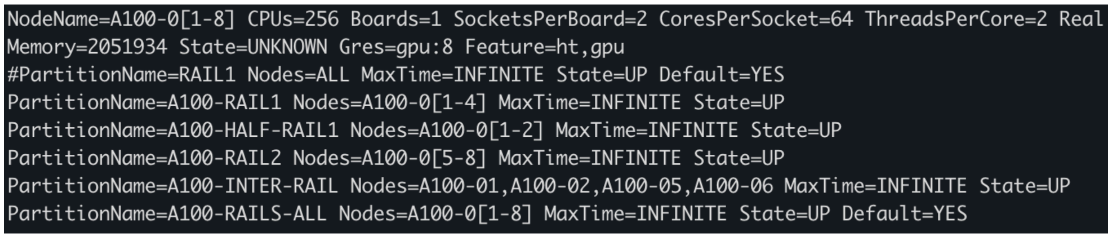
Priority-ordered jobs are assigned to nodes within a partition until resources have been exhausted, and a user can initiate parallel work within a partition by using job steps.

*Configurations*

Additionally, another important feature of Slurm is the configuration file, typically called “slurm.conf”. This file contains general Slurm configuration information, such as the nodes to be managed, the partitions, and other scheduling parameters within these partitions; it is accessible from anywhere in the cluster. There are many parameters available to define and split resources, which can be accessed here.



**Feature**
All nodes with this single feature will be included as part of this nodeset.
**Nodes**
List of nodes in this set.
**NodeSet**
Unique name for a set of nodes. Must not overlap with any NodeName definitions.

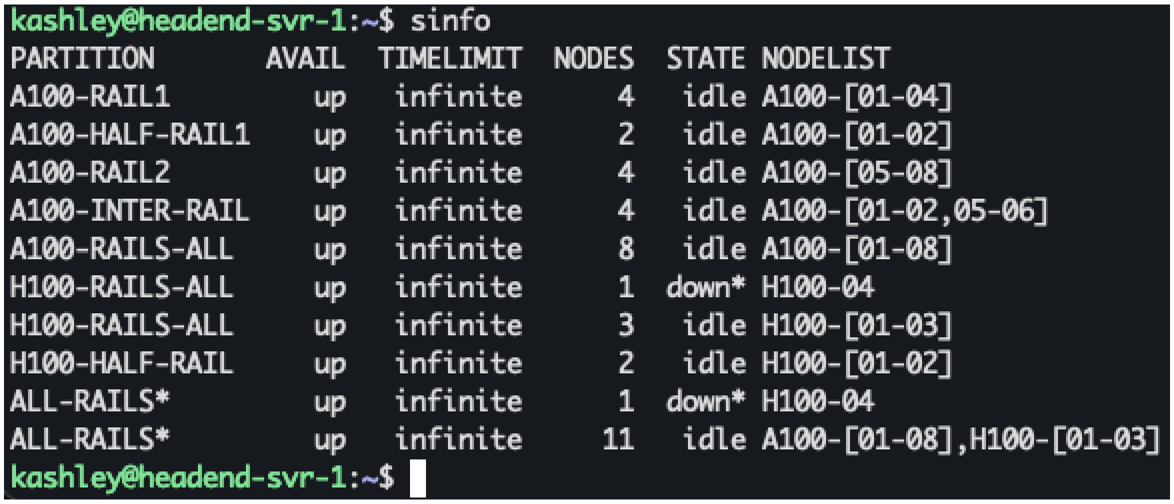


```
NodeName=A100-0[1-8] CPUs=256 Boards=1 SocketsPerBoard=2 CoresPerSocket=64 ThreadsPerCore=2 Real
Memory=2051934 State=UNKNOWN Gres=gpu:8 Feature=ht,gpu
#PartitionName=RAIL1 Nodes=ALL MaxTime=INFINITE State=UP Default=YES
PartitionName=A100-RAIL1 Nodes=A100-0[1-4] MaxTime=INFINITE State=UP
PartitionName=A100-HALF-RAIL1 Nodes=A100-0[1-2] MaxTime=INFINITE State=UP
PartitionName=A100-RAIL2 Nodes=A100-0[5-8] MaxTime=INFINITE State=UP
PartitionName=A100-INTER-RAIL Nodes=A100-01,A100-02,A100-05,A100-06 MaxTime=INFINITE State=UP
PartitionName=A100-RAILS-ALL Nodes=A100-0[1-8] MaxTime=INFINITE State=UP Default=YES
```

Within the file, it contains a nodeset configuration with the following format, which can be used to name a specific set of nodes which facilitates the partition configuration.

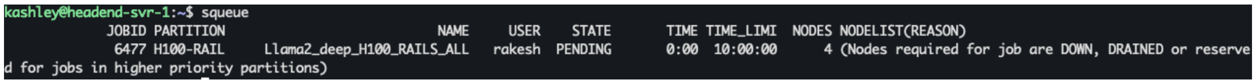
The partition configuration allows one to “establish different job limits or access controls for various groups of nodes”. While nodes can belong to multiple partitions, and thus different constraints can be made on the same node, jobs can only be allocated resources within a partition. Each partition can have its own parameters, also detailed here. Some examples are included in the partition configuration above. In this example, we have parameters such as Nodes, MaxTime, State, Default, and more, each varying among the partitions.

*Interaction/Navigation*:



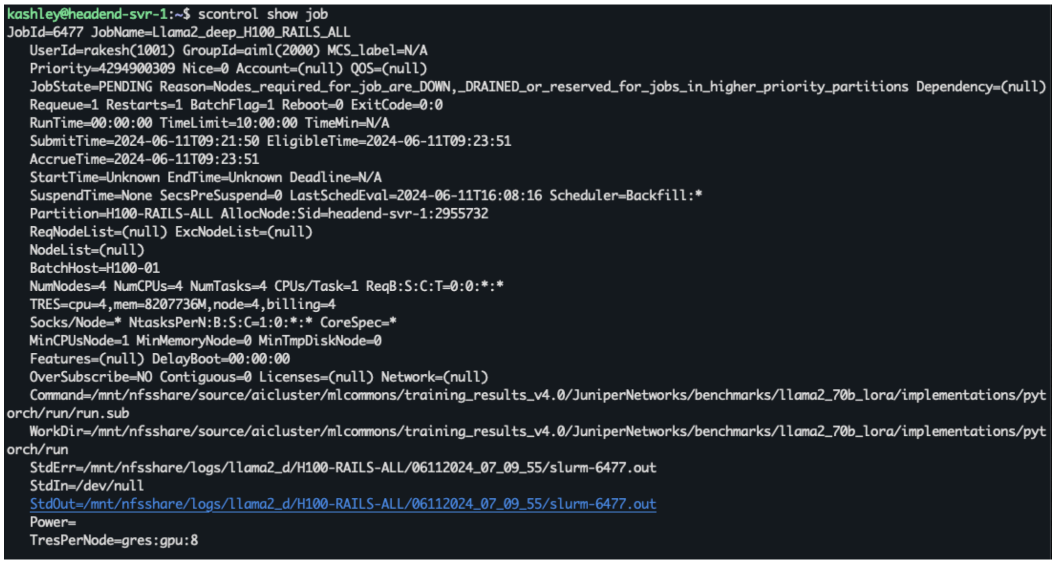
```
kashley@headend-svr-1:~$ sinfo
PARTITION       AVAIL  TIMELIMIT  NODES  STATE NODELIST
A100-RAIL1         up   infinite      4   idle A100-[01-04]
A100-HALF-RAIL1    up   infinite      2   idle A100-[01-02]
A100-RAIL2         up   infinite      4   idle A100-[05-08]
A100-INTER-RAIL    up   infinite      4   idle A100-[01-02,05-06]
A100-RAILS-ALL     up   infinite      8   idle A100-[01-08]
H100-RAILS-ALL     up   infinite      1  down* H100-04
H100-RAILS-ALL     up   infinite      3   idle H100-[01-03]
H100-HALF-RAIL     up   infinite      2   idle H100-[01-02]
ALL-RAILS*         up   infinite      1  down* H100-04
ALL-RAILS*         up   infinite     11   idle A100-[01-08],H100-[01-03]
kashley@headend-svr-1:~$
```

Let's unpack how one might use Slurm. Once logged into the cluster, one can use any of the aforementioned commands to check on the status of jobs, allocate resources, etc. For instance, if I wanted to check on the status of my cluster, I might use "sinfo", as pictured above. Here, we can see the different partitions within the cluster, and the associated number of nodes, the nodes, the time limit available for the partition, and the status. The state represents the status, where "idle" means not in use (and hence available), and "down" indicates that the nodes aren't responding.

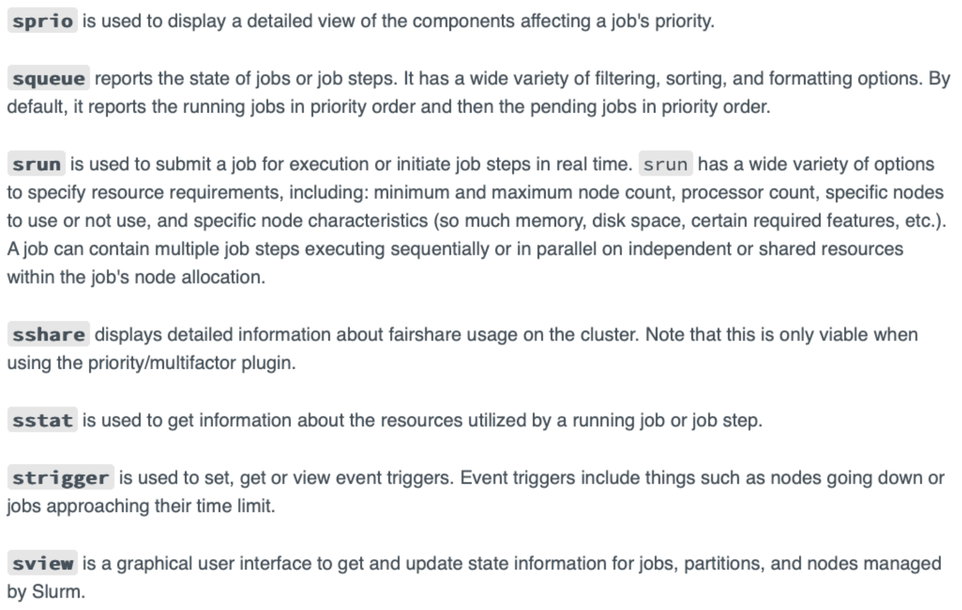


```
kashley@headend-svr-1:~$ squeue
             JOBID PARTITION                           NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
              6477 H100-RAIL     Llama2_deep_H100_RAILS_ALL   rakesh  PENDING       0:00  10:00:00      4 (Nodes required for job are DOWN, DRAINED or reserve
d for jobs in higher priority partitions)
```

Another important command is "squeue" (pictured above), which displays the job queue and all the jobs that currently have resources allocated. Each job has a job ID, the name of the partition used, the user the resources were allocated to, the time limit for the partition, the number of nodes allocated, etc (this information is also available in the logs). While a job is running, we can use the command "scontrol show job" to see a more detailed description of the job and its resources. Another command to note is the "srun" command, which allows for heterogenous jobs, or "jobs in which each component has virtually all job options available including partition, account and QOS (Quality Of Service)".



```
kashley@headend-svr-1:~$ scontrol show job
JobId=6477 JobName=Llama2_deep_H100_RAILS_ALL
   UserId=rakesh(1001) GroupId=aiml(2000) MCS_label=N/A
   Priority=4294900309 Nice=0 Account=(null) QOS=(null)
   JobState=PENDING Reason=Nodes_required_for_job_are_DOWN,_DRAINED_or_reserved_for_jobs_in_higher_priority_partitions Dependency=(null)
   Requeue=1 Restarts=1 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=10:00:00 TimeMin=N/A
   SubmitTime=2024-06-11T09:21:50 EligibleTime=2024-06-11T09:23:51
   AccrueTime=2024-06-11T09:23:51
   StartTime=Unknown EndTime=Unknown Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-06-11T16:08:16 Scheduler=Backfill:*
   Partition=H100-RAILS-ALL AllocNode:Sid=headend-svr-1:2955732
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   BatchHost=H100-01
   NumNodes=4 NumCPUs=4 NumTasks=4 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=4,mem=8207736M,node=4,billing=4
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
   Command=/mnt/nfsshare/source/aicluster/mlcommons/training_results_v4.0/JuniperNetworks/benchmarks/llama2_70b_lora/implementations/pyt
orch/run/run.sub
   WorkDir=/mnt/nfsshare/source/aicluster/mlcommons/training_results_v4.0/JuniperNetworks/benchmarks/llama2_70b_lora/implementations/pyt
orch/run
   StdErr=/mnt/nfsshare/logs/llama2_d/H100-RAILS-ALL/06112024_07_09_55/slurm-6477.out
   StdIn=/dev/null
   StdOut=/mnt/nfsshare/logs/llama2_d/H100-RAILS-ALL/06112024_07_09_55/slurm-6477.out
   Power=
   TresPerNode=gres:gpu:8
```



`sprio` is used to display a detailed view of the components affecting a job's priority.

`squeue` reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.

`srun` is used to submit a job for execution or initiate job steps in real time. `srun` has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared resources within the job's node allocation.

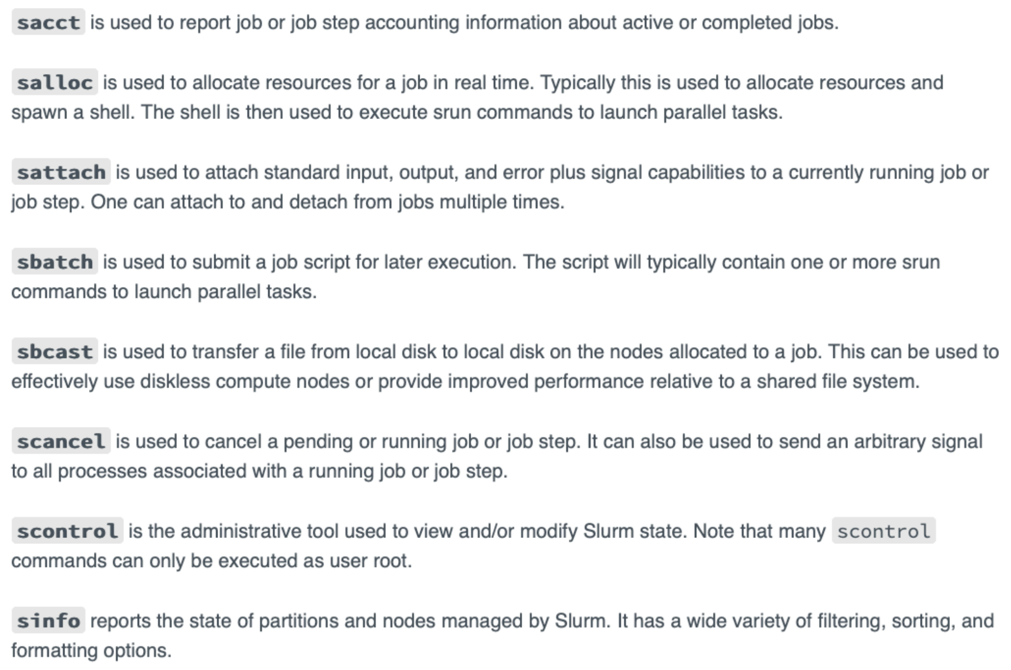
`sshare` displays detailed information about fairshare usage on the cluster. Note that this is only viable when using the priority/multifactor plugin.

`sstat` is used to get information about the resources utilized by a running job or job step.

`strigger` is used to set, get or view event triggers. Event triggers include things such as nodes going down or jobs approaching their time limit.

`sview` is a graphical user interface to get and update state information for jobs, partitions, and nodes managed by Slurm.

To the left, I've included a list of some of the most common commands that one might use.



`sacct` is used to report job or job step accounting information about active or completed jobs.

`salloc` is used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.

`sattach` is used to attach standard input, output, and error plus signal capabilities to a currently running job or job step. One can attach to and detach from jobs multiple times.

`sbatch` is used to submit a job script for later execution. The script will typically contain one or more srun commands to launch parallel tasks.

`sbcast` is used to transfer a file from local disk to local disk on the nodes allocated to a job. This can be used to effectively use diskless compute nodes or provide improved performance relative to a shared file system.

`scancel` is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

`scontrol` is the administrative tool used to view and/or modify Slurm state. Note that many `scontrol` commands can only be executed as user root.

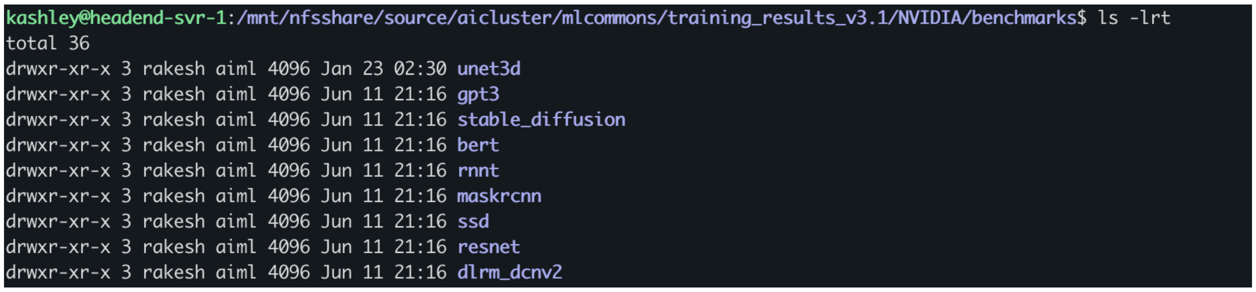
`sinfo` reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

*Installation Information*

For the rest of this document, I am using version 21.08.5 of Slurm – to check, simply use the Slurm command "sinfo -V" or "rpm -qa | grep slurm". If you do not have Slurm set up, you must download authentication service MUNGE, and then Slurm, and then install Slurm using RPM or Debian packages; finally, you must create a configuration file. Make sure this configuration file is accessible across all nodes, and that you install the configuration file in <sysconfdir>/slurm.conf. Then, run the slurm control daemon (slurmctld) and then the slurm daemons.

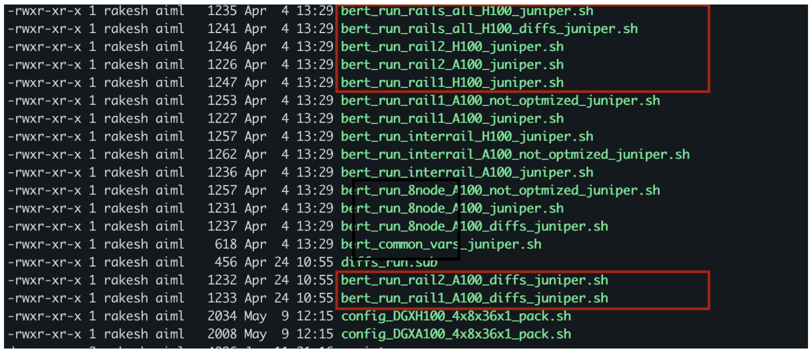
*How To Run Within The Cluster:*

1. SSH into the cluster. Depending on admin status, you will need to set up a password for all servers.
2. Then, cd into the correct model in the correct file, or use the correct shorthand. For example, a command to go directly to run the BERT model is "cdbert".



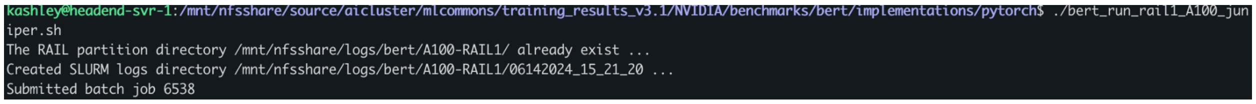
```
kashley@headend-svr-1:/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks$ ls -lrt
total 36
drwxr-xr-x 3 rakesh aiml 4096 Jan 23 02:30 unet3d
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 gpt3
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 stable_diffusion
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 bert
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 rnnt
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 maskrcnn
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 ssd
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 resnet
drwxr-xr-x 3 rakesh aiml 4096 Jun 11 21:16 dlrm_dcnv2
```

Running "cdbert" and listing its contents yields this (trimmed as the list was too long).



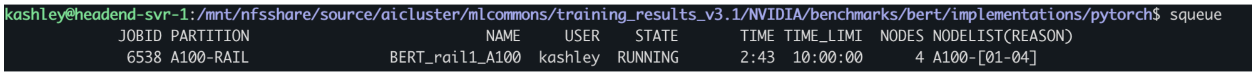
```
-rwxr-xr-x 1 rakesh aiml   1235 Apr  4 13:29 bert_run_rails_all_H100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1241 Apr  4 13:29 bert_run_rails_all_H100_diffs_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1246 Apr  4 13:29 bert_run_rail2_H100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1226 Apr  4 13:29 bert_run_rail2_A100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1247 Apr  4 13:29 bert_run_rail1_H100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1253 Apr  4 13:29 bert_run_rail1_A100_not_optmized_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1227 Apr  4 13:29 bert_run_rail1_A100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1257 Apr  4 13:29 bert_run_interrail_H100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1262 Apr  4 13:29 bert_run_interrail_A100_not_optmized_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1236 Apr  4 13:29 bert_run_interrail_A100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1257 Apr  4 13:29 bert_run_8node_A100_not_optmized_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1231 Apr  4 13:29 bert_run_8node_A100_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1237 Apr  4 13:29 bert_run_8node_A100_diffs_juniper.sh
-rwxr-xr-x 1 rakesh aiml    618 Apr  4 13:29 bert_common_vars_juniper.sh
-rwxr-xr-x 1 rakesh aiml    456 Apr 24 10:55 diffs_run.sub
-rwxr-xr-x 1 rakesh aiml   1232 Apr 24 10:55 bert_run_rail2_A100_diffs_juniper.sh
-rwxr-xr-x 1 rakesh aiml   1233 Apr 24 10:55 bert_run_rail1_A100_diffs_juniper.sh
-rwxr-xr-x 1 rakesh aiml   2034 May  9 12:15 config_DGXH100_4x8x36x1_pack.sh
-rwxr-xr-x 1 rakesh aiml   2008 May  9 12:15 config_DGXA100_4x8x36x1_pack.sh
```

3. Run one of the .sh files highlighted in red. Once you run it, this is the output:



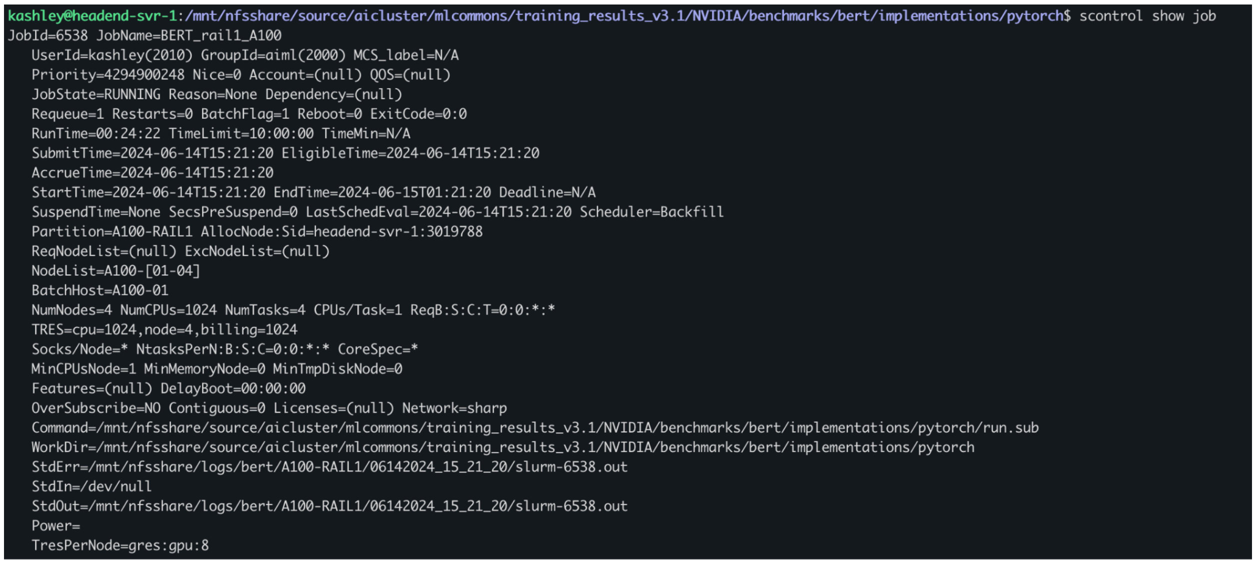
```
kashley@headend-svr-1:/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks/bert/implementations/pytorch$ ./bert_run_rail1_A100_jun
iper.sh
The RAIL partition directory /mnt/nfsshare/logs/bert/A100-RAIL1/ already exist ...
Created SLURM logs directory /mnt/nfsshare/logs/bert/A100-RAIL1/06142024_15_21_20 ...
Submitted batch job 6538
```

4. Then, if you run "squeue", your job should be up and running.



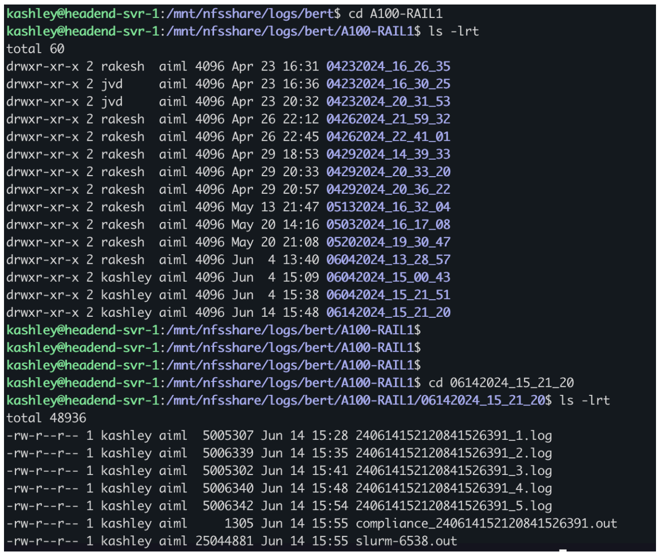
```
kashley@headend-svr-1:/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks/bert/implementations/pytorch$ squeue
             JOBID PARTITION                            NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
              6538 A100-RAIL                  BERT_rail1_A100  kashley  RUNNING       2:43  10:00:00      4 A100-[01-04]
```

If you run "scontrol show job", then you will get a more detailed description of the job assigned:



```
kashley@headend-svr-1:/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks/bert/implementations/pytorch$ scontrol show job
JobId=6538 JobName=BERT_rail1_A100
   UserId=kashley(2010) GroupId=aiml(2000) MCS_label=N/A
   Priority=4294900248 Nice=0 Account=(null) QOS=(null)
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:24:22 TimeLimit=10:00:00 TimeMin=N/A
   SubmitTime=2024-06-14T15:21:20 EligibleTime=2024-06-14T15:21:20
   AccrueTime=2024-06-14T15:21:20
   StartTime=2024-06-14T15:21:20 EndTime=2024-06-15T01:21:20 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-06-14T15:21:20 Scheduler=Backfill
   Partition=A100-RAIL1 AllocNode:Sid=headend-svr-1:3019788
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=A100-[01-04]
   BatchHost=A100-01
   NumNodes=4 NumCPUs=1024 NumTasks=4 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1024,node=4,billing=1024
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=sharp
   Command=/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks/bert/implementations/pytorch/run.sub
   WorkDir=/mnt/nfsshare/source/aicluster/mlcommons/training_results_v3.1/NVIDIA/benchmarks/bert/implementations/pytorch
   StdErr=/mnt/nfsshare/logs/bert/A100-RAIL1/06142024_15_21_20/slurm-6538.out
   StdIn=/dev/null
   StdOut=/mnt/nfsshare/logs/bert/A100-RAIL1/06142024_15_21_20/slurm-6538.out
   Power=
   TresPerNode=gres:gpu:8
```

5. This information can also be accessed in the logs section of mnt/nfsshare. Simply use the shortcut command "cdlogs" and then cd into the correct model. In my case, I cd into BERT and then find the correct log. The Job ID is also visible in the above picture (6538).



```
kashley@headend-svr-1:/mnt/nfsshare/logs/bert$ cd A100-RAIL1
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1$ ls -lrt
total 60
drwxr-xr-x 2 rakesh  aiml 4096 Apr 23 16:31 04232024_16_26_35
drwxr-xr-x 2 jvd     aiml 4096 Apr 23 16:36 04232024_16_30_25
drwxr-xr-x 2 jvd     aiml 4096 Apr 23 20:32 04232024_20_31_53
drwxr-xr-x 2 rakesh  aiml 4096 Apr 26 22:12 04262024_21_59_32
drwxr-xr-x 2 rakesh  aiml 4096 Apr 26 22:45 04262024_22_41_01
drwxr-xr-x 2 rakesh  aiml 4096 Apr 29 18:53 04292024_14_39_33
drwxr-xr-x 2 rakesh  aiml 4096 Apr 29 20:33 04292024_20_33_20
drwxr-xr-x 2 rakesh  aiml 4096 Apr 29 20:57 04292024_20_36_22
drwxr-xr-x 2 rakesh  aiml 4096 May 13 21:47 05132024_16_32_04
drwxr-xr-x 2 rakesh  aiml 4096 May 20 14:16 05032024_16_17_08
drwxr-xr-x 2 rakesh  aiml 4096 May 20 21:08 05202024_19_30_47
drwxr-xr-x 2 rakesh  aiml 4096 Jun  4 13:40 06042024_13_28_57
drwxr-xr-x 2 kashley aiml 4096 Jun  4 15:09 06042024_15_00_43
drwxr-xr-x 2 kashley aiml 4096 Jun  4 15:38 06042024_15_21_51
drwxr-xr-x 2 kashley aiml 4096 Jun 14 15:48 06142024_15_21_20
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1$
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1$
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1$
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1$ cd 06142024_15_21_20
kashley@headend-svr-1:/mnt/nfsshare/logs/bert/A100-RAIL1/06142024_15_21_20$ ls -lrt
total 48936
-rw-r--r-- 1 kashley aiml  5005307 Jun 14 15:28 24061415212084152639l_1.log
-rw-r--r-- 1 kashley aiml  5006339 Jun 14 15:35 24061415212084152639l_2.log
-rw-r--r-- 1 kashley aiml  5005302 Jun 14 15:41 24061415212084152639l_3.log
-rw-r--r-- 1 kashley aiml  5006340 Jun 14 15:48 24061415212084152639l_4.log
-rw-r--r-- 1 kashley aiml  5006342 Jun 14 15:54 24061415212084152639l_5.log
-rw-r--r-- 1 kashley aiml     1305 Jun 14 15:55 compliance_24061415212084152639l.out
-rw-r--r-- 1 kashley aiml 25044881 Jun 14 15:55 slurm-6538.out
```