Slurm Batch

Slurm is a batch scheduler that enables users (you!) to submit long (or even short) compute 'jobs' to our compute clusters. It will queue up jobs such that the (limited) resources compute resources available are fairly shared and distributed for all users. This page describes basic usage of slurm at SLAC. It will provide some simple examples of how to request common resources.

Slurm is currently being tested and is scheduled for deployment on the SLAC Scientific Shared Data Facility. We welcome any suggestions and issues to be reported to unix-admin@slac.stanford.edu. Note that whilst we strive to keep the information on these pages up-to-date, there may be inconsistencies and/or incorrect information contained within.

By default, all users when they first use slurm will have access to the shared Account on the shared Partition with scavenger QoS.

If you belong to a group that has contributed hardware into the SDF, you will be eligible to use different Accounts and Partitions:

- We are testing the ability for your group/team Slurm Administrator to have the ability to add users to their Accounts (delegated administration). If you wish to represent your group/team to do this, please contact us!
- We will need to know which slurm Account to 'bill' you against (don't worry, there will be no $ charge for usage, it's purely for accounting and reporting). This Account will most likely be your immediate group/team that you work with. Please send your unix username and your group/team name to unix-admin@slac.stanford.edu.

We do NOT, and WILL NOT support AFS tokens with slurm. This will cause your jobs to fail if you try to write to anywhere under /afs (including your currently home ~ directories). We shall be deploying new storage in the near future, with dedicated home and data directories. In the meantime, It is recommended to use GPFS space if your group currently has any.

Why should I use Batch?

Whilst your desktop computer and or laptop computer has a fast processor and quick local access to data stored on its hard disk/ssd; you may want to run either very big and/or very large compute tasks that may require a lot of CPUs, GPUs, memory, or a lot of data. Our compute servers that are part of the Batch system allows your to do this. Our servers typically also have very fast access to centralised storage, have (some) common software already preinstalled, and will enable you to run these long tasks without impacting your local desktop/laptop resources.

Why should I use Slurm?

Historically, we have always use IBM's LSF as our Batch scheduler software. However, with new hardware such as GPU's, we have found that the user experience and the administrative accounting features of LSF to be lacking. Slurm is also commonly used across academic and laboratory environments and we hope that this commonality will facilitate easy usage for you, and simpler administration for us.

What should I know about using Batch?

The first thing to note is that you should probably be comfortable in a Unix 'command line' environment. LINKS?

When you submit a compute task to the batch system, this is called a Job. We need to charge each Job to an Account. You may also select what pool of servers to run the Jobs on - this is known as a Partition.

You should also acquaint yourself with slurm Accounts and Partitions.

What is a Slurm Account?

As the number of servers and GPUs in our environment is limited (but not small), we need to keep account of who uses what. In addition, as groups/teams can purchase their own servers to be added to the SDF we must provide a method of which allocated users can have priority access to the servers that were purchased for them. A slurm Account is basically something that you will charge your job against.

What is a Slurm Partition?

A Partition is a logical grouping of compute servers. These may be servers of a similar technical specification (eg Cascade Lake CPUs, Telsa GPUs etc), or by ownership of the servers - eg SUNCAT group may have purchased so many servers, so we put them all into a Partition.

Generally, all servers will be placed in the shared partition that everyone with a slac computer account will have access to (although at a low priority).

Users should contact their Coordinators to be added to appropriate group Partitions to get priority access to resources.

You can view the active Partitions on SDF with

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
shared*      up 7-00:00:00     21   unk* cryoem-gpu[02,04-09,11-15],ml-gpu[02-10]
shared*      up 7-00:00:00     10   idle cryoem-gpu[01,03,10,50],hep-gpu01,ml-gpu[01,11],nu-gpu[01-03]
ml           up   infinite      9   unk* ml-gpu[02-10]
ml           up   infinite      2   idle ml-gpu[01,11]
neutrino     up   infinite      3   idle nu-gpu[01-03]
cryoem       up   infinite     12   unk* cryoem-gpu[02,04-09,11-15]
cryoem       up   infinite      4   idle cryoem-gpu[01,03,10,50]
```

What is a Slurm Allocation?

In order to provide appropriate access for users to the hardware, an Allocation is created that defines what User can run on what Partition and charge against what Account (there's a bit more in the backend to this).

How do I use Slurm?

We are still testing the best way to deploy Slurm at SLAC, and as such, some of the examples and instructions that follow may be subject to change. If you have any opinions and or suggestions, we would love to hear from.

Slurm is installed on a limited number of hosts currently. We recommend logging on using ssh via a terminal:

```
ssh ocio-gpu01.slac.stanford.edu
```

In order to get the slurm binaries available, you will need to use modules to add the slurm binaries into your path environment:

```
module load slurm
```

We will likely have the above command automatically run, so it may not be necessary later.

Common commands are:

srun	request a quick job to be ran - eg an interactive terminal
sbatch	submit a batch job to run
squeue	show jobs
scancel	cancel a job
scontrol show job	show job details
sstat	show job usage details
sacctmgr	manage Associations

How can I get an Interactive Terminal?

We are experiencing problems with the shared partition with interactive jobs: srun will claim that it's waiting for resources, but in fact the allocation fails immediately. we have yet to experience the same issue with other partitions. Your batch jobs should continue to function correctly, however.

use the srun command

```
module load slurm
srun -A shared -p shared -n 1 --pty /bin/bash
```

This will then execute /bin/bash on a (scheduled) server in the Partition shared and charge against Account shared. This will request a single CPU, launch a pseudo terminal (pty) where bash will run. You may be provided different Accounts and Partitions and should use them when possible.

Note that when you 'exit' the interactive session, it will relinquish the resources for someone else to use. This also means that if your terminal is disconnected (you turn your laptop off, loose network etc), then the Job will also terminate (similar to ssh).

How do I submit a Batch Job?

We are NOT support AFS as part of slurm deployment. We shall be migrating home directories and group directories over to our new storage appliances as part of SDF deployment. If you wish to access your AFS files, please copy them over to the new storage. *elaborate.

use the sbatch command, this primer needs to be elaborated:

Create a job submission script (text file) script.sh:

```
#!/bin/bash

#SBATCH --account=shared
#SBATCH --partition=shared
#SBATCH --qos=scavenger
#
#SBATCH --job-name=test
#SBATCH --output=output-%j.txt
#SBATCH --error=output-%j.txt
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=12
#SBATCH --mem-per-cpu=1g
#
#SBATCH --time=10:00
#
#SBATCH --gpus 1

<commands here>
```

In the above example, we submit a job named 'test' and output both stdout and stderr to the same file (%j will be replaced with the Job ID). We request a single Task (think of it as an MPI rank) and that single task will request 12 CPUs; each of which will be allocated 1GB of RAM - so a total of 12GB. By default, the --ntasks will be equivalent to the number of nodes (servers) asked for. In order to aid scheduling (and potentially prioritising the Job), we limit the length of the Job to 10 minutes.

We also request a single GPU with the Job. This will be exposed via CUDA_VISIBLE_DEVICES. To specify specific GPU's, see below.

You will need an account (see below). All SLAC users have access to the "shared" partition with a quality of service of "scavenger". This is so that stakeholders of machines in the SDF will get priority access to their resources, whilst any user can use all resources as long as the 'owners' of the hardware isn't wanting to use it. As such, owners (or stakeholders) will have qos "normal" access to their partitions (of which such hosts are also within the shared partition).

Then, in order to submit the job:

```
module load slurm
sbatch script.sh
```

You can then use the command to monitor your job progress:

```
squeue
```

And you can cancel the job with

```
scancel <jobid>
```

How can I request GPUs?

You can use the --gpus to specify gpus for your jobs: Using a number will request the number of any gpu that is available (what you get depends upon what your Account/Association is and what is available when you request it). You can also specify the type of gpus by prefixing the number with the model name. eg

```
# request single gpu
srun -A shared -p shared -n 1 --gpus 1 --pty /bin/bash

# request a gtx 1080 gpu
srun -A shared -p shared -n 1 --gpus geforce_gtx_1080_ti:1 --pty /bin/bash

# request a gtx 2080 gpu
srun -A shared -p shared -n 1 --gpus geforce_rtx_2080_ti:1 --pty /bin/bash

# request a v100 gpu
srun -A shared -p shared -n 1 --gpus v100:1 --pty /bin/bash
```

How can I see what GPUs are available?

```
# sinfo -o "%12P %5D %14F %7z %7m %10d %11l %42G %38N %f"
PARTITION    NODES NODES(A/I/O/T) S:C:T   MEMORY  TMP_DISK   TIMELIMIT                                   AVAIL_FEATURES
GRES                                       NODELIST
shared*      1     0/1/0/1        2:8:2   191567  0          7-00:00:00  gpu:v100:
4                                 nu-gpu02                               CPU_GEN:SKX,CPU_SKU:4110,CPU_FRQ:2.10
GHz,GPU_GEN:VLT,GPU_SKU:V100,GPU_MEM:32GB,GPU_CC:7.0
shared*      8     0/1/7/8        2:12:2  257336  0          7-00:00:00  gpu:geforce_gtx_1080_ti:
10                cryoem-gpu[02-09]                      CPU_GEN:HSW,CPU_SKU:E5-2670v3,CPU_FRQ:2.30GHz,GPU_GEN:
PSC,GPU_SKU:GTX1080TI,GPU_MEM:11GB,GPU_CC:6.1
shared*      14    0/0/14/14      2:12:2  191552  0          7-00:00:00  gpu:geforce_rtx_2080_ti:
10                cryoem-gpu[11-15],ml-gpu[02-10]        CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
shared*      1     0/1/0/1        2:12:2  257336  0          7-00:00:00  gpu:geforce_gtx_1080_ti:10(S:
0)           cryoem-gpu01                           CPU_GEN:HSW,CPU_SKU:E5-2670v3,CPU_FRQ:2.30GHz,GPU_GEN:PSC,
GPU_SKU:GTX1080TI,GPU_MEM:11GB,GPU_CC:6.1
shared*      3     0/3/0/3        2:12:2  191552  0          7-00:00:00  gpu:geforce_rtx_2080_ti:10(S:
0)           cryoem-gpu10,ml-gpu[01,11]             CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
shared*      3     0/3/0/3        2:8:2   191567  0          7-00:00:00  gpu:v100:4(S:0-
1)                          cryoem-gpu50,nu-gpu[01,03]            CPU_GEN:SKX,CPU_SKU:4110,CPU_FRQ:2.10GHz,
GPU_GEN:VLT,GPU_SKU:V100,GPU_MEM:32GB,GPU_CC:7.0
shared*      1     0/1/0/1        2:12:2  257330  0          7-00:00:00  gpu:geforce_gtx_1080_ti:8(S:0),gpu:
titan_x hep-gpu01                                     CPU_GEN:HSW,CPU_SKU:E5-2670v3,CPU_FRQ:2.30GHz,GPU_GEN:PSC,
GPU_SKU:GTX1080TI,GPU_MEM:11GB,GPU_CC:6.1
ml           9     0/0/9/9        2:12:2  191552  0          infinite    gpu:geforce_rtx_2080_ti:
10                ml-gpu[02-10]                          CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
ml           2     0/2/0/2        2:12:2  191552  0          infinite    gpu:geforce_rtx_2080_ti:10(S:
0)           ml-gpu[01,11]                          CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
neutrino     1     0/1/0/1        2:8:2   191567  0          infinite    gpu:v100:
4                                 nu-gpu02                               CPU_GEN:SKX,CPU_SKU:4110,CPU_FRQ:2.10
GHz,GPU_GEN:VLT,GPU_SKU:V100,GPU_MEM:32GB,GPU_CC:7.0
neutrino     2     0/2/0/2        2:8:2   191567  0          infinite    gpu:v100:4(S:0-
1)                          nu-gpu[01,03]                         CPU_GEN:SKX,CPU_SKU:4110,CPU_FRQ:2.10GHz,
GPU_GEN:VLT,GPU_SKU:V100,GPU_MEM:32GB,GPU_CC:7.0
cryoem       8     0/1/7/8        2:12:2  257336  0          infinite    gpu:geforce_gtx_1080_ti:
10                cryoem-gpu[02-09]                      CPU_GEN:HSW,CPU_SKU:E5-2670v3,CPU_FRQ:2.30GHz,GPU_GEN:
PSC,GPU_SKU:GTX1080TI,GPU_MEM:11GB,GPU_CC:6.1
cryoem       5     0/0/5/5        2:12:2  191552  0          infinite    gpu:geforce_rtx_2080_ti:
10                cryoem-gpu[11-15]                      CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
cryoem       1     0/1/0/1        2:12:2  257336  0          infinite    gpu:geforce_gtx_1080_ti:10(S:
0)           cryoem-gpu01                           CPU_GEN:HSW,CPU_SKU:E5-2670v3,CPU_FRQ:2.30GHz,GPU_GEN:PSC,
GPU_SKU:GTX1080TI,GPU_MEM:11GB,GPU_CC:6.1
cryoem       1     0/1/0/1        2:12:2  191552  0          infinite    gpu:geforce_rtx_2080_ti:10(S:
0)           cryoem-gpu10                           CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,
GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
cryoem       1     0/1/0/1        2:8:2   191567  0          infinite    gpu:v100:4(S:0-
1)                          cryoem-gpu50                          CPU_GEN:SKX,CPU_SKU:4110,CPU_FRQ:2.10GHz,
GPU_GEN:VLT,GPU_SKU:V100,GPU_MEM:32GB,GPU_CC:7.0
```

How can I request a GPU with certain features and or memory?

TBA... something about using Constraints. Maybe get the gres for gpu memory working.

What Accounts are there?

Accounts are used to allow us to track, monitor and report on usage of SDF resources. As such, users who are members of stakeholders of SDF hardware, should use their relevant Account to charge their jobs against. We do not associate any monetary value to Accounts currently, but we do require all Jobs to be charged against an Account.

Account Name	Description	Contact

shared	Everyone	Yee
cryoem	CryoEM Group	Yee
neutrino	Neutrino Group	Kazu
cryoem-daq	CryoEM data acquitision	Yee
ml	Machine Learning Initiative	Daniel
suncat	SUNCAT Group	Johannes
hps	HPS Group	Omar
atlas	ATLAS Group	Yee/Wei
LCLS	LCLS Group	Wilko

What Partitions are there?

Partitions define a grouping of machines. In our use case the grouping to refer to science and engineering groups who have purchased servers for the SDF. We do this such that members (or associates) of those groups can have priority access to their hardware. Whilst we give everyone access to all hardware, by default, users who belong to groups who do not own any stake in SDF will have lower priority access and use of stakeholder's resources.

Partition Name	Purpose	Contact
shared	General resources; this contains all shareable reasources, including GPUs	Yee
ml	Machine Learning Initiative GPU servers	Daniel / Yee
cryoem	CryoEM GPU servers	Yee
neutrino	Neutrino GPU servers	Kazu
suncat	SUNCAT AMD Rome Servers	Johannes
hps	HPS AMD Rome Servers	Omar
fermi	Fermi (LAT) AMD Rome Servers	Richard
atlas	ATLAS GPU Servers	Yee / Wei
lcls	LCLS AMD Rome Servers	Wilko

Help! My Job takes a long time before it starts!

This is often due to limited resources. The simplest way is to request less CPU (-N) or less memory for your Job. However, this will also likely increase the amount of time that you need for the Job to complete. Note that perfect scaling is often very difficult (ie using 16 CPUs will run twice as fast as 8 CPUs), so it may be beneficial to submit many smaller Jobs where possible. You can also set the --time option to specify that your job will only run upto that amount of time so that the scheduler can better fit your job in.

The more expensive option is to buy more hardware to SDF and have it added to your group/teams Partition.

You can also make use of the Scavenger QoS such that your job may run on any available resources available at SLAC. This, however, has the disadvantage that should the owners of the hardware that your job runs on requires its resources, your may will be terminated (preempted) - possibly before it has completed.

What is QoS?

A Quality of Service for a job defines restrictions on how a job is ran. In relation to an Allocation, a user may preempt, or be preempted by other job with a 'higher' QoS. We define 2 levels of QoS:

scavenger: Everyone has access to all resources, however it is ran with the lowest priority and will be terminated if another job with a higher priority needs it

normal: Standard QoS for owners of hardware; jobs will (attempt) to run til completion and will not be preempted. normal jobs therefore will preempt scavenger jobs.

Scavenger QoS is useful if you have jobs that may be resumed (checkpointed) and if there are available resources available (ie owners are not using all of their resources).

You may submit to multiple Partition with the same QoS level:

```
#!/bin/bash
#SBATCH --account=cryoem
#SBATCH --partition=cryoem,shared
#SBATCH --qos=scavenger
```

In the above example, a cryoem user is charging against their Account cryoem; she is willing to run the job whereever available (the use of the cryoem Partition is kinda moot as the cryoem nodes are a subset of the Shared Partition anyway).

is it possible to define multiple? ie cryoem with normal + shared with scavenger?

How can I restrict/contraint which servers to run my Job on?

You can use slurm Constraints. We tag each and every server that help identify specific Features that each has: whether that is the kind of CPU, or the kind of GPU that run on them.

You can view a servers specific Feature's using

```
$ module load slurm
$ scontrol show node ml-gpu01
NodeName=ml-gpu01 Arch=x86_64 CoresPerSocket=12
   CPUAlloc=0 CPUTot=48 CPULoad=1.41
   AvailableFeatures=CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:
7.5
   ActiveFeatures=CPU_GEN:SKX,CPU_SKU:5118,CPU_FRQ:2.30GHz,GPU_GEN:TUR,GPU_SKU:RTX2080TI,GPU_MEM:11GB,GPU_CC:7.5
   Gres=gpu:geforce_rtx_2080_ti:10(S:0)
   NodeAddr=ml-gpu01 NodeHostName=ml-gpu01 Version=19.05.2
   OS=Linux 3.10.0-1062.4.1.el7.x86_64 #1 SMP Fri Oct 18 17:15:30 UTC 2019
   RealMemory=191552 AllocMem=0 FreeMem=182473 Sockets=2 Boards=1
   State=IDLE ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2019-11-12T11:18:04 SlurmdStartTime=2019-12-06T16:42:16
   CfgTRES=cpu=48,mem=191552M,billing=48,gres/gpu=10
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

We are openly investigating additional Features to add. Comments and suggestions welcome.

Documentation PENDING.

Possibly add: GPU_DRV, OS_VER, OS_TYPE