Mox_scheduler

Mox uses a scheduler called slurm.

Below xyz is your hyak group name and abc is your UW netid.

To logon to mox.hyak:

```
ssh abc@mox.hyak.uw.edu
```

The above command gives you access to the login node of mox.hyak. The login node is only for logging in and submitting jobs. The computational work is done on a compute node. As shown below, you can get either an interactive compute node or submit a batch job. The build node is a special compute node which can connect to the internet.

To see information about your partition (aka allocations):

```
sinfo -p xyz
```

To see the usage of your group's nodes:

```
squeue -p xyz
```

To see your jobs:

```
squeue -u abc
```

Below mox specific command shows the number of nodes etc. of all allocations.

```
hyakalloc
```

Interactive Single Node Usage with srun:

The build node can connect to outside mox. It is useful for using git, transferring files to outside mox or getting files from outside mox, installing packages in R or Python etc.

To get an interactive build node for 2 hours:

For mox.hyak:

```
srun -p build --time=2:00:00 --mem=20G --pty /bin/bash
```

(Note: (a) --pty /bin/bash must be the last option in above command.

(b) If you do not get a build node with above values of --mem, then try smaller values.

(c) You can connect to the internet from the build node.)

To get an interactive node with 4 cores in your own group for 2 hours on mox:

```
srun -p xyz -A xyz --nodes=1 --ntasks-per-node=4 --time=2:00:00 --mem=100G --pty /bin/bash
```

Note that if you are using an interactive node to run a parallel application such as Python multiprocessing, MPI, OpenMP etc. then the number in the `--ntasks-per-node` option must match the number of processes used by your application. For example for the above srun command:

(a) For Python multiprocessing use code like "p=multiprocessing.Pool(4)".

(b) For MPI use, "mpirun -np 4 myprogram"

(c) For OpenMP use "export omp_num_threads=4"

(d) For GNU parallel use "-j 4" option

When you are using the build node with "-p build", then you do not need to give the -A option.

Also if your group has an interactive node then use the xyz-int for the -p option. For example:

srun -p stf-int -A stf --time=1:00:00 --mem=10G --pty /bin/bash

Note that an interactive node in your own group cannot connect to outside mox or ikt.

Specifying memory:

(1) It is important to use the --mem option to specify the required memory. If the memory is not specified then the the SLURM scheduler limits the usage of memory to

a default value even if more memory is available on the node. Usually the default value is quite low.

(2) For ikt.hyak regular compute node use, --mem=58G . For ikt.hyak for build node, use --mem=10G.

(3) If you know that your group has larger memory nodes, then you can use larger values in --mem option.

The value in --mem option should be smaller than the memory on the node. This is because the operating system also needs some memory

For 64GB nodes, use --mem=58G   (This is the smallest memory node on ikt.)

For 128GB nodes, use ---mem=120G (This is the smallest memory node on mox.)

For 256GB nodes, use --mem=248G

For 512GB nodes, use --mem=500G

For 192GB nodes, use --mem=185G

For 384GB nodes, use --mem=374G

For 768GB nodes, use --mem=752G

For the knl nodes, use --mem=200G

SLURM environment variables:

Issue below comand at an interactive node prompt to find the list of SLURM environment variables:

```
export | grep SLURM
```

Interactive Multiple Node Usage with srun:

If you are setting up an application which uses multiple nodes (e.g. Apache Spark), you will need interactive access to multiple nodes .

To get 2 nodes with 28 cores per node for interactive use:

```
srun --nodes 2 --ntasks-per-node 28 -p xyz -A xyz  --time=2:00:00 --mem=100G --pty /bin/bash
```

When the above command runs, then you will have been allocated 2 nodes but will be on one of the two allocated node. The --mem option is for memory per node.

In order to find the names of the nodes that you have been allocated, issue below command:

```
scontrol show hostnames
```

Once you know the node names, then you can use them for your work (e.g. Apache Spark, Hadoop etc.)

Below command will tell you about other SLURM environment variables.

```
export | grep SLURM
```

Using the checkpoint ckpt queue:

See below link for using the ckpt queue:

Mox_checkpoint

scancel

Use scancel to cancel jobs. For example to cancel a job with job id 1234:

```
scancel 1234
```

If your userid is abc then you can cancel all your jobs by using below command:

```
scancel -u abc
```

sstat

Use sstat for information (e.g. memory usage) about your currently running jobs.

```
sstat --format=jobid,MaxVMSize
```

For details of other format options see https://slurm.schedmd.com/sstat.html

sacct

Use sacct for information (e.g. memory usage) about your jobs which have completed.

```
sacct --format=jobid,elapsed,MaxVMSize
```

For details of other format options see https://slurm.schedmd.com/sacct.html

sreport

For historical usage for group xyz for month of March 2018:

```
sreport cluster UserUtilizationByAccount Start=2018-03-01 End=2018-03-31 Accounts=xyz
```

Batch usage Single Node:

Submit a batch job from mox login node:

```
sbatch myscript.slurm
```

Below is an example slurm script. The --mem option is for memory per node. Replace the last line "myprogram" with the commands to run your program (e.g. load modules, copy input files, run your program, copy output files etc.).

```
#!/bin/bash

## Job Name

#SBATCH --job-name=myjob

## Allocation Definition

## The account and partition options should be the same except in a few cases (e.g. ckpt queue and genpool queue).

#SBATCH --account=xyz
#SBATCH --partition=xyz

## Resources

## Total number of Nodes

#SBATCH --nodes=1

## Number of cores per node

#SBATCH --ntasks-per-node=28

## Walltime (3 hours). Do not specify a walltime substantially more than your job needs.

#SBATCH --time=3:00:00

## Memory per node. It is important to specify the memory since the default memory is very small.
## For mox, --mem may be more than 100G depending on the memory of your nodes.

## For ikt, --mem may be 58G or more depending on the memory of your nodes.

## See above section on "Specifying memory" for choices for --mem.

#SBATCH --mem=100G

## Specify the working directory for this job

#SBATCH --chdir=/gscratch/xyz/abc/myjobdir

##turn on e-mail notification

#SBATCH --mail-type=ALL

#SBATCH --mail-user=your_email_address

## export all your environment variables to the batch job session

#SBATCH --export=all
```

myprogram

Batch usage Multiple Nodes:

If you use multiple nodes in one batch job, then your program should know how to use all the nodes. For example, your program should be a MPI program.

Use the same batch file as above batch file for single node except change the number of nodes as shown below.

The value for the --nodes option should be equal to or less than the total number of nodes owned by your group.

The value of the --ntasks-per-node option should be no greater than 40 as it represents the maximum number of cores per node you are requesting and no node type has more than this. However, the nodes you are able to access may not have 40 cores so your job could pend indefinitely. Please check which resources are available to you and their limits in determining what you can or should request.

For ikt:

```
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=16
```
For mox:
```
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=28
#    OR
```

#SBATCH --ntasks-per-node=40

Self-limiting your number of running jobs:

At times you may wish to self-limit the number of your jobs that will be run simultaneously in order to leave nodes in your group's allocation available for immediate use by others in your group. To achieve this, you can assign one of the "MaxJobs<N>" QOSs to your job, where <N> is the number (1 through 10) of jobs with the same QOS selected that will be run at a time. NOTE: This is not enabled on the ckpt partition.
For example, adding the following to your script:

#SBATCH --qos=MaxJobs2

...would tell the scheduler to only allow two jobs with the 'MaxJobs2' QOS selected to be run at a time. Any jobs you start without the 'MaxJobs2' QOS set would not be limited, and jobs with a different MaxJobs<N> QOS set will be limited separately (so, for a example, you could have one set of submitted jobs with the MaxJobs1 QOS selected, and another set with the MoxJobs2 QOS selected. This would result in a total of up to three running jobs at a time; one from the former set, and two from the latter).

Common Slurm Error messages:

Below are some common error messages from slurm. You may see other errors in slurm .out and .err files.

(1) If you use "--mem 120G" and your program uses more memory than 120G then your slurm job will end and you will get below error in the slurm output file. Slurm output files have names like slurm-348658.out.
```
slurmstepd: error: Exceeded job memory limit
```

(2) If you use "--time X" and X is greater than the time to the next mox.hyak or ikt.hyak maintenance day then your job will not start and your will get below error:

```
(ReqNodeNotAvail, UnavailableNodes:n[<node numbers list>]
```

(3) If you use "-p xyz -A xyz" and you do not belong to the xyz group then you will get below error:

```
Unable to allocate resources: Invalid account or account/partition combination specified
```

More details are here:

Hyak mox Overview

https://slurm.schedmd.com/quickstart.html

https://slurm.schedmd.com/documentation.html

===== Below is for Advanced users only ====

Below is about salloc. Do not use salloc unless you have a specific reason.

Interactive Multiple Node Usage with salloc:

If you are setting up an application which uses multiple nodes (e.g. Apache Spark), you will need interactive access to multiple nodes .

To get 2 nodes for interactive use:

```
salloc -N 2 -p xyz -A xyz  --time=2:00:00 --mem=100G
```

When the above command runs, then you will have been allocated 2 nodes but will still be on the login node.

If you issue a command like below then srun will run the command hostname on each node:

```
srun hostname
```

In order to find the names of the nodes that you have been allocated, issue below command:

```
scontrol show hostnames
```

Once you know the node names, then you can use them for your work (e.g. Apache Spark, Hadoop etc.)

Below command will tell you about other SLURM environment variables.

```
export | grep SLURM
```

srun vs salloc

If no nodes have been allocated then (1) and (2) are equivalent.

(1) `srun -p xyz -A xyz --time=4:00:00  --mem=100G --pty /bin/bash`

(2) `salloc -p xyz -A xyz --time=4:00:00   --mem=100G`   This allocates the node.

   `srun --pty /bin/bash`   This uses the same node.

If nodes have already been allocated by using salloc then srun just uses those nodes.

Alternative to srun is to allocate a node and then ssh to it e.g.

Allocate a node:

```
salloc -p xyz -A xyz --time=4:00:00 --mem=100G
```

Find the name of the allocated node:

```
export | grep LIST
```

If above command gives a hostname of n1234 then you can ssh to n1234 and use it for your work.

If you you use srun to get an interactive node then when you exit the node, your access to the node also ends.

If you use salloc followed by ssh to get an interactive node then when you exit the node, the node will still be allocated to you. Hence, you can again ssh to the node till the till the time in the salloc command is over.