SLURM

Introduction to SLURM

- When you login to HPC cluster, you land on **Login Nodes**
  - Login nodes are not meant to run jobs
  - These are used to submit jobs to **Compute Nodes.** You can setup your files and data on Login node. If compilation/installation of an application is required, user must do it on login node.
- To submit job on the cluster, you need to write a scheduler job script
- SLURM - Simple Linux Utility for Resource Management
- It is a workload manager that provides a framework for job queues, allocation of compute nodes, and the start and execution of jobs.

How to set Environment ?

- By default no application is set in your environment. User must explicitly set required ones

- **module** is the utility (also command name) to enable use of applications / libraries / compilers available on the HPC cluster.
- Module structure on IIT KGP Cluster
  - apps/<application name>/version : Applications available on the cluster
  - compiler/<compiler name>/version : Compilers available on the cluster
  - lib/<library name>/version : Available libraries
  - ….

How to set Environment ?

- Some Important commands :
  - **module avail** To see the available software installed on HPC system
    - list of precompiled applications
    - different compilers and libraries (compilers include GNU, Intel, PGI)
  - **module list** Shows the currently loaded modules in your shell
  - **module load <Name of the module>**
    - module load compiler/intel/2018.2.199 (to set Intel compilers version 2018 in your environment)
    - module load apps/namd/2.12/impi2018/cpu (to set NAMD app version 2.12 in your environment)

How to set Environment ?

- Some Important commands :

  - **module unload <Name of the module>**
  - **module purge** To clear all the loaded modules

Conda Env

- To use Python / conda / DL framework on a cluster -
  Load the corresponding module
  $ module load python/conda-python/3.7_new
- Available DL Frame work in above module (both cpu and gpu)
  Tensorflow - 2.0.0
  pytorch - 1.3.1
  caffe - 1.0

Conda Env

- **Create local conda env**
- To create an environment:

  $ conda create --name myenv

  OR

  $ conda create -n myenv

- Specifying a location for an environment

  $ conda create --prefix <PATH>

  e.g . $ conda create --prefix ./envs

Conda Env

- I need a conda package(s) that isn't installed, what do I do?

create your own conda local environment and install required package .

e.g . $ module load python/conda-python/3.7_new

$ conda create -n myenv

$ source activate myenv

$ conda install <package-name>

Conda Env

- **Create local conda env for tensorflow 1.14.0 with python 3.6**

```
$ module load python/conda-python/3.7_new
$ conda create -n tf 1.4 python=3.6 tensorflow=1.14.0
```

- Activate the new environment

```
$ source activate tf1.4
        OR
$ conda activate tf1.4
```

Tip : Install all the package(s) that you want in the environment at the same time. Installing 1 package at a time can lead to dependency conflicts.

Job Submission

SLURM: Sample Job Script for Serial Jobs

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=1           # specify number of CPU cores per node
#SBATCH --time=00:10:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=blast              # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=standard          # specify type of resource such as
CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/mpiblast/1.6.0/intel

### Your command for Serial Execution (i.e. execution on one CPU core)
time  <executable name with required parameters OR application command>
```

Resource specifications

(user must choose values appropriate for her/his job)

All necessary settings for your applications

SLURM: Job Script for Parallel Jobs on CPUs

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=8           # specify number of CPU cores per node
#SBATCH --time=01:00:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=gromacs            # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=cpu               # specify type of resource such as
                                        CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/gromacs/15.2.2019/intel

### Your command for Parallel Execution (e.g. with MPI)
time  mpirun  -np  $SLURM_NTASKS  <executable name or application
command>
```

SLURM: Job Script for Parallel Jobs on GPUs

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes requested
#SBATCH --ntasks-per-node=20          # specify number of CPU cores per node
#SBATCH --gres=gpu:1                  # specify no. of GPU devices per node
#SBATCH --time=01:0:00                # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=namdgpu            # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=gpu               # specify type of resource i.e. GPU

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/namd/2.12/impi2018/cpu

### Your command for Parallel Execution with GPUs (e.g. with MPI)
time   mpirun   -np   $SLURM_NTASKS    <executable name or application command>
```

SLURM: Other Useful Commands

- **squeue** To see the status of all jobs submitted on the cluster
  squeue -u <user name> To see status of user's jobs only. Also shows job-id.
  - On a HPC cluster many jobs may be running at the same time
  - If the resources requested by a user's job are not free, the job will be put in queue and will be taken up for execution once the resources are available
  - To minimize the waiting time in queue, the requested resources such as execution time, number of nodes and cores per node must be carefully chosen within the job script
  - If the actual job execution exceeds the time duration specified in the job script, the job will be terminated

- **sinfo** Provides the basic information about the resources on HPC cluster such as
  - Partitions/queue such as for cpu / gpu / high memory nodes
  - Number of nodes for each type and their numbering/names
  - State of the nodes

- **scancel <job ID>** To delete the submitted jobs

Demo 1

Job Submission : Gromacs

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=8           # specify number of CPU cores per node
#SBATCH --time=01:00:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=gromacs            # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=cpu               # specify type of resource such as
                                        CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/gromacs/15.2.2019/intel

### Your command for Parallel Execution (e.g. with MPI)
time  mpirun  -np  $SLURM_NTASKS  <executable name or application command>
```

Job Submission : LAMMPS

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=8           # specify number of CPU cores per node
#SBATCH --time=01:00:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=lammps             # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=cpu               # specify type of resource such as
                                        CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/lammps/7Aug19/intel

### Your command for Parallel Execution (e.g. with MPI)
time  mpirun  -np  $SLURM_NTASKS  <executable name or application
command>
```

Demo 2

Job Submission : OpenFoam

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=8           # specify number of CPU cores per node
#SBATCH --time=01:00:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=openfoam           # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=cpu               # specify type of resource such as
                                        CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/openfoam/6.0/intel

### Your command for Parallel Execution (e.g. with MPI)
time  mpirun  -np  $SLURM_NTASKS  <executable name or application
command>
```

Job Submission : NAMD

```
#!/bin/sh
#SBATCH -N 1                          # specify number of nodes
#SBATCH --ntasks-per-node=8           # specify number of CPU cores per node
#SBATCH --time=01:00:00               # specify maximum duration of run
                                        in hours:minutes:seconds format
#SBATCH --job-name=namd              # specify job name
#SBATCH --error=job.%J.err            # specify error file name
#SBATCH --output=job.%J. out          # specify output file name
#SBATCH --partition=cpu               # specify type of resource such as
                                        CPU/GPU/High Memory etc.

cd  $SLURM_SUBMIT_DIR                 # change to directory from where job is submitted

### Set your environment (e.g. load required compiler, application)
module  load  apps/namd/2.12/impi2018/cpu

### Your command for Parallel Execution (e.g. with MPI)
time  mpirun  -np  $SLURM_NTASKS  <executable name or application command>
```

Demo 3

Job Submission : DL / conda env

```
#!/bin/sh
#SBATCH -N 1                    # specify number of nodes
#SBATCH --ntasks-per-node=8     # specify number of CPU cores per node
#SBATCH --time=01:00:00         # specify maximum duration of run   in hours:minutes:seconds format
#SBATCH --job-name=tf-1.14.0    # specify job name
#SBATCH --error=job.%J.err      # specify error file name
#SBATCH --output=job.%J. out    # specify output file name
#SBATCH --partition=standard    # specify type of resource such as CPU/GPU/High Memory etc.
cd   $SLURM_SUBMIT_DIR          # change to directory from where job is submitted
### Set your environment (e.g. load required compiler, application)
        module   load   python/conda-python/3.7_new
###  activate your local environment
          source  activate  tf1.14
### Your command for  Execution (
  python <yourcode.py>
```