Job Submission with SLURM Workload Manager

1. Main entities

- Nodes: The physical computing resources available for processing.
- Partition: A logical grouping of nodes used for organizing and managing resources.
- Jobs: The allocation of resources to a user for a specified duration, encompassing all the necessary computations.
- Tasks: The specific computing resources assigned to a process or set of processes within a job or job step. (By default, each task corresponds to one CPU core.)

2. Commands

sinfo	- view information about nodes and partitions.Information about HPC resources:
sbatch <your_batch_script.sh>	- submits a job script for execution. The submitted job stays in the queue until the requested resources become available.
squeue	- monitor your job (view information about jobs located in queue).
scontrol show job <your_job_id>	- list detailed information for a currently running job (useful for troubleshooting).
scancel <your_job_id>	- cancel or terminate jobs, job steps, or job arrays in a workload manager.

3. Simple job submission

Create “simple.sh” script with the following content in your directory

```
#!/bin/bash

#SBATCH --job-name=simple
#SBATCH --partition=compute
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --output=output-%j.out
#SBATCH --error=output-%j.err

hostname
lscpu
sleep 60
```

Run the script

```
$ sbatch simple.sh
```

Check the job

```
$ squeue -u $USER
```

The scheduler will automatically create an output file that in your directory will contain the result of the commands run in the script file.

4. Parallel MPI job on a single node

Copy this into your working directory as “hellompi.c”:

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

int main(int argc, char *argv[], char *envp[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  printf("Process %d on %s out of %d\n", rank, processor_name, numprocs);

  MPI_Finalize();
}
```

Load MPI config files

```
$ source /soft/libraries/openmpi/3.1.6/settings.sh
```

Compile the code

```
$ mpicc -o hellompi hellompi.c
```

Create “hellompi.sh” script with the following content in your directory

```
#!/bin/bash

#SBATCH --job-name=mpi_single_node          #job name
#SBATCH --nodes=1                           #compute node count
#SBATCH --output=output-%j.out
#SBATCH --error=output-%j.err
#SBATCH --time=02:00:00
#SBATCH --partition=compute
#SBATCH --ntasks-per-node=32                #run a 32-core job

. /soft/libraries/openmpi/3.1.6/settings.sh  #load MPI config files

mpirun –np 32 ./hellompi
```

Run the script

```
$ sbatch simple.sh
```

Check the job

```
$ squeue -u $USER
```

The scheduler will automatically create an output file that in your directory will contain the result of the commands run in the script file.

5. Parallel MPI job script using N nodes

Add the following parameters in your script.

```
#SBATCH --nodes=n                                  #n different nodes
```

6. Getting Help

For any issues or further assistance, contact the cluster support team: support@anscc.sci.am
Additionally, refer to the official Slurm Quick Start User Guide and Documentation for comprehensive details on using Slurm.