Ivy Cluster (400 Nodes HPC Cluster)

Users Hands on Guide

Computing Systems Section

Computer Division

Indira Gandhi Centre for Atomic Research

1. Objective

The objective of this document is to give the overview of Ivy Cluster (400 Nodes HPC cluster) system installed at Computer Centre. This document helps the users to familiarize the software environment in order to utilize the cluster system efficiently. It also gives the systematic procedure for the beginners to do job submission and management in the Ivy Cluster.

2. Introduction

A computing cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource. A node in the cluster is a single or multiprocessor system with memory, I/O facilities and Operating System. A High-Performance Computing (HPC) Cluster typically has a large number of computers interconnected by high-speed, low-latency network. HPC Clusters are designed to use parallel computing to solve highly compute intensive problems.

A significant milestone is achieved in scientific computing at IGCAR through the commissioning of a High-Performance Computing Cluster with 400 nodes based on the latest Intel Xeon processors. The processors are based on Ivy Bridge microarchitecture and therefore the cluster is named as “Ivy Cluster”. This supercomputing cluster delivers a Peak (theoretical) performance of 207 TeraFLOPS and maximal sustained performance of around 180 TeraFLOPS with industry-standard HPL benchmark. The cluster is designed to meet the large-scale numerical and data-intensive computing requirements of the organization.

Ivy Cluster : Configuration

The Ivy Cluster comprise of two numbers of master node / head node, 400 compute nodes, 2 compute nodes with GPU accelerators of NVIDIA Tesla K40, 2 compute nodes with Intel Phi 7120P coprocessors and one management node. The overall configuration of the system is given in Fig. 1, depicting the major functional units and the inter-connections. Each cluster node is powered by 64-bit Intel Xeon dual twelve-core processor with Ivy Bridge architecture of clock speed of 2.7 GHz. Each node has 128GB DDR3 Fully Buffered Memory and two SATA 1 TB Hot-swappable Internal HDDs. The cluster has 9600 processor cores with 52 Terabytes of distributed memory. Each node gives the performance of about 500 GFLOPS with Linpack (SMP) benchmark.

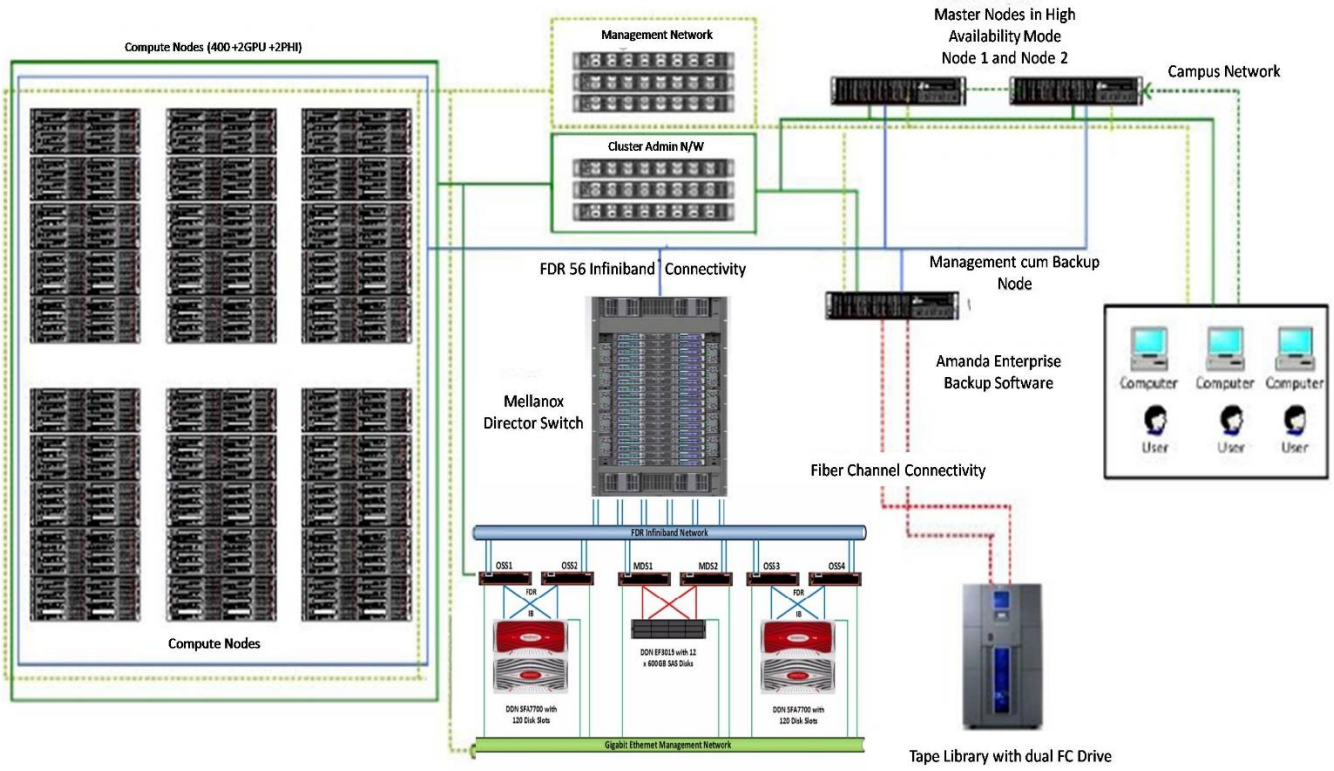

Fig. 1: Schematic of 400-node Ivy Cluster with network connectivity

The head node of the cluster is used for cluster administration and providing user interface for job submission and management. This node is configured in High Availability (HA) mode to increase availability of the cluster. In addition to the master node, a management node is also configured for the hardware management of all the compute nodes using IPMI. The compute nodes are the cluster nodes where the users' jobs actually run. The cluster is highly optimized to execute the parallel codes of the users.

The major functional units of the Ivy Cluster are described in the following section.

Interconnect Networks

The cluster system has three interconnect networks. These are InfiniBand network, administration network and hardware management network. Among the three networks, the primary network meant for inter-process communication is the InfiniBand. The three types of networking involved to build the cluster are:

- **Infiniband network** acts as primary network that supports for inter-process communication (IPC). This network is a low latency, high-bandwidth network and being widely used in the HPC arena. The Infiniband switch of Ivy Cluster supports up to 432 FDR 4X ports each with speed of 56Gb/s. InfiniBand Architecture is an industry standard, channel-based, switched fabric, interconnect architecture for servers. It uses low-level Remote Direct Memory Access (RDMA) protocol to reduce the application latency and processor overhead.
- **Administration network** used for cluster management and monitoring the healthiness of the compute nodes. This network is based on 1Gbps Gigabit Ethernet.
- **Hardware management network** meant for hardware remote management and console access of nodes using Intelligent Platform Management Interface (IPMI). This network is based on Gigabit Ethernet. IPMI provides remote access, monitoring, and control functions for hardware without using KVM switches and complex wiring.

Storage Architecture

The storage system of a high-performance computing cluster needs a large capacity of storage, accessible at high speeds in high availability mode. Simultaneous access from multiple nodes and a single global namespace are the other two requirements for the storage. In order to meet these requirements, the storage system of the cluster is configured using a parallel and distributed file system named 'Luster'. Lustre is an open-source file system and supports all the special requirements of storage of the cluster system.

The Ivy Cluster has 500 TB usable storage system configured with Intel sourced Lustre 2.5 Parallel File System (PFS). The storage system architecture of the cluster is depicted in Fig. 2. The Lustre storage architecture of Ivy Cluster consists of two metadata servers (MDS) which manage the names and directories in the file system and four object storage servers (OSS) which provide file input/output (I/O) service. Luster metadata servers are configured in active/passive mode, while object storage servers are deployed in active/active mode. This configuration provides redundancy without extra overhead that improves file system performance, enhances file system recoverability and availability. The recovery feature of Lustre, allows servers to be upgraded without the need to take the system down. The hard disks in storage units are configured in RAID level 6, enabling file system operations to continue in the event of a double disk failure. The object storage servers are connected to storage units using 56 Gbps InifiniBand links and metadata servers are connected to meta-data storage units using 8 Gbps fiber-channel links.

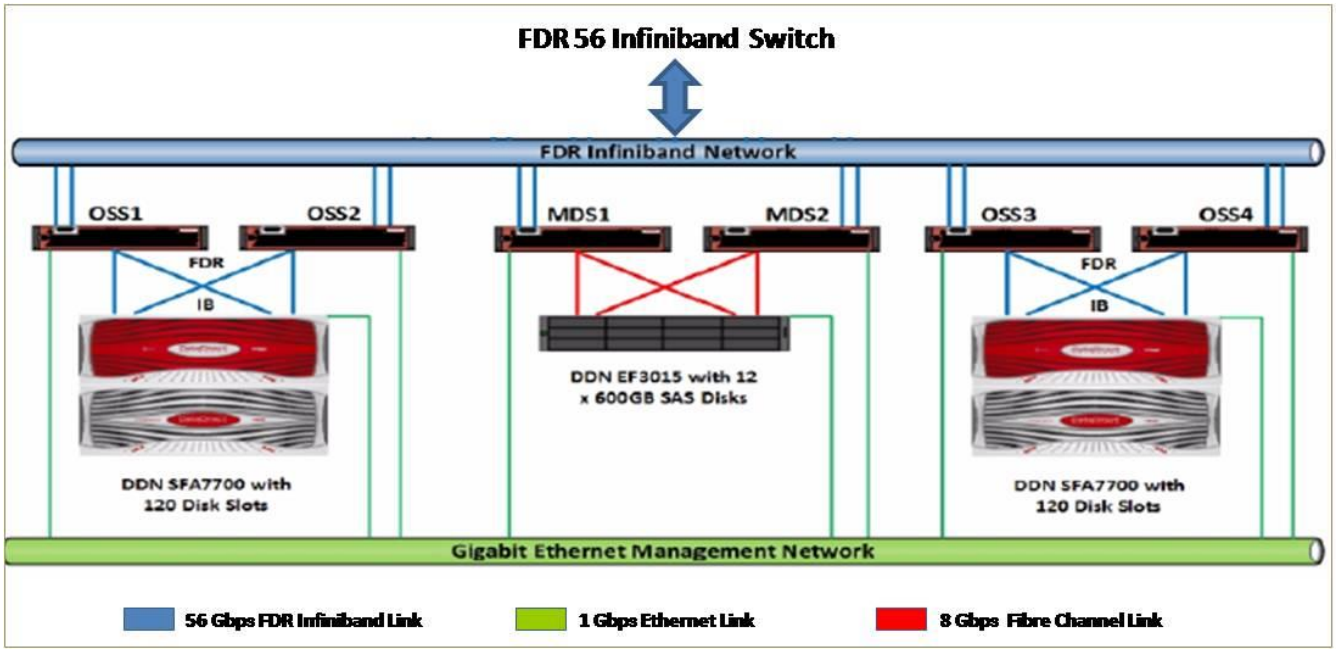

Fig. 2: Architecture layout and connectivity of 500 TeraByte Storage System

3. Ivy Cluster: Lustre File System and User Space

The total usable space of Lustre based storage of Ivy Cluster is divided in to two partitions named 'home' and 'apps'. The home directory of each user account of the cluster is created on the 'home' partition and has the following general format:

/home/<GroupName>/<UserName>

Apart from the home directory, every user is assigned with a relatively large scratch area for running the jobs and saving the intermediate and temporary files. That area is created on 'apps' partition and has the following general format:

/apps/scratch/<GroupName>/<UserName>

For ease of switching between these two directories, aliases are provided as follows:

- To go to the home directory (which is having the general format as /home/<GroupName>/<UserName>) type 'uhm' from any directory
- To go to the scratch area (which is having the general format as /apps/scratch/<GroupName>/<UserName>) type 'shm' from any directory

Quotas are configured for each user for the above two directories, so that every user gets a fair and equal storage space as below:

- Maximum of 500GB on home

- Maximum of 5TB on scratch

Users who are crossing this limit have to delete unwanted files from their respective folder to copy any new content. Apart from the user quota, group quota is also configured such that the space occupied by all the users in one group cannot cross 20TB on scratch area (/apps file system). There is no such group quota restriction on /home though. Therefore, they are free to use full 500GB under home directory. The details about quota setting implemented on the Ivy Cluster are given in Table 1.

Table. 1 : Details about quota setting implemented on the Ivy Cluster

Quota settings	Soft limit	Hard limit	Grace time
'/home' user limit	495GB	500GB	99weeks
'/apps/scratch' user limit	5000GB	5TB	99weeks
'/apps/scratch' group limit	20000GB	20TB	99weeks

Users have to ensure that the used space does not cross the Hard Limit (i.e 500GB user limit on /home and 5TB user limit or 20TB group limit on /apps). In case of specific requirement of more disk space, users can approach the concerned officials of Computer Division. Users can check their quota by running the following commands:

- Type 'uquota' to get current status about user's home area.
- Type 'squota' to get current status about user's scratch area.

4. Ivy Cluster: Software Details

Operating System

Red Hat Enterprise Linux Server version 6.6 is the Operating System installed in all the cluster nodes. It is the world's leading application platform deployed in majority of cluster installations.

Compilers and Tools

The following compilers and development tools are installed on the cluster system.

a) Intel compilers 16.0 C/C++ and Fortran
b) GNU Compilers 4.4.7 for C/C++ and Fortran

Libraries

This cluster is installed with Intel Parallel Studio XE, which consists of set of libraries to support highly optimized algorithmic building blocks for all data analysis stages. They are

a) Intel® Data Analytics Acceleration Library (DAAL)
b) Intel® Math Kernel Library
c) Intel® Integrated Performance Primitives
d) Intel® Threading Building Blocks
e) BLAS 3.6.0
f) scalapack 2.0.2
g) superlu 4.3
h) lapack 3.6.0
i) atlas 3.10.2
j) blocksolve95 3.0
k) FFTW 3.3.4
l) HDF5 1.8.16

MPI Libraries

The list of MPI libraries available in the cluster are as follows:

a) Intel® MPI Library 5.1.2
b) Intel® Trace Analyzer and Collector
c) mvapich2-2.1
d) openmpi-1.8.8
e) mpich2

Ganglia

Ganglia is a web enabled scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It allows the user to remotely view live or historical statistics (such as CPU load averages or network utilization) for all machines that are being monitored.

In order to get the information about Ivy Cluster through Ganglia, just type the following link from the browser:

http://10.1.2.3/ganglia/

5. Ivy Cluster: SLURM Resource Management System

A resource management system manages the processing load by preventing jobs from competing with each other for limited compute resources. Typically, a resource management system comprises a resource manager and a job scheduler. The resource management system named 'Slurm' (Simple Linux Utility for Resource Management) is configured for the Ivy Cluster. The details about the Slurm configuration are given in the subsequent section.

Slurm is an open source, fault-tolerant and highly scalable resource management system for Linux clusters. Slurm comprises of a resource manager and a job scheduler (which is built-in to the resource manager itself). The major components and interdependencies of Slurm resource management system are shown in Fig. 3. Slurm relieves the users of the task of finding free resources for job execution in a cluster and helps them in job submission and monitoring. It has following three key functions:

1. Resource Allocation: It allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work.
2. Job Management: It provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes.
3. Job Scheduling: It arbitrates contention for resources by managing a queue of pending work.

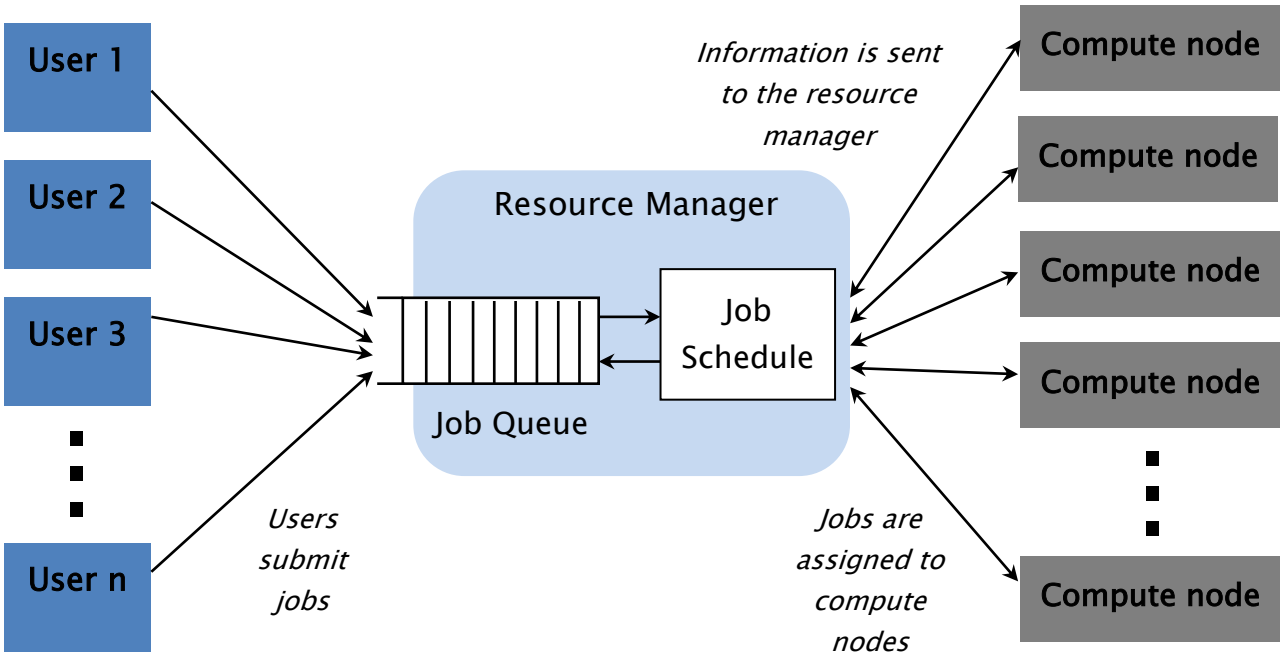

Fig. 3: Slurm Resource Management System: Major Components and interdependencies

Slurm Commands

1. **sinfo** : Displays queue/ partition names and available nodes.

Usage:
$sinfo // displays status of the partitions/ queues which the user can access
$sinfo –p testq // displays the status of the queue named testq
$sinfo –a // displays status of all the partitions of the cluster
$ sinfo –s // displays partition state summary without node state details.

Example 1:

```
[prashantcss@head1 ~]$ sinfo -a
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
all.q*       up   infinite      6  alloc node[1-4,6,8]
all.q*       up   infinite    161   idle node[5,7,9-167]
gpuqueue     up   infinite      2  idle* gpunode[1-2]
phiqueue     up   infinite      2  down* phinode[1-2]
chem.q       up   infinite      1  down* node210
chem.q       up   infinite     82   idle node[168-209,211-250]
cmd.q        up   infinite     83   idle node[251-333]
seq.q        up   infinite      4   idle node[334-337]
fem.q        up   infinite      8   idle node[338-345]
pvm.q        up   infinite     17   idle node[346-362]
comp.q       up   infinite      1  down* node376
comp.q       up   infinite      4  alloc node[365-368]
comp.q       up   infinite     33   idle node[363-364,369-375,377-400]
```

The details about the status information of the above example are describes as follows:

- **PARTITION** column lists the available queues specific to the user.
- **AVAIL** column specify if jobs can be allocated nodes or queued in this partition. Possible values are ‘UP’, ‘DOWN’, ‘DRAIN’ and ‘INACTIVE’.
  - *UP:* Designates that new jobs may queued on the partition, and that jobs may be allocated nodes and run from the partition.
  - *DOWN:* Designates that new jobs may be queued on the partition, but queued jobs may not be allocated nodes and run from the partition. Jobs already running on the partition continue to run.
  - *DRAIN:* Designates that no new jobs may be queued on the partition, but jobs already queued on the partition may be allocated nodes and run.
  - *INACTIVE:* Designates that no new jobs may be queued on the partition, and jobs already queued may not be allocated nodes and run.
- **TIMELIMIT** column specifies the maximum time limit for any job on a particular queue/partition. Infinite is used to identify partitions without job time limit.
- **STATE** column identify the state to be assigned to the node. Typically possible node states are ‘ALLOC’, ‘COMP’, ‘DOWN’, ‘DRAIN’, ‘IDLE’ and ‘MIX’.

- IDLE: Identify the nodes available for allocation.
- DOWN state will cause all running and suspended jobs on that node to be terminated.
- DRAIN state identifies nodes that are unavailable for use.
- ALLOC: Identifies nodes that has been allocated and are running job.
- MIX state means the node is in multiple states. For instance some CPUs of the node are ALLOCATED and rests CPUs are IDLE.
- COMP: All the jobs associated with the node in this state are in the process of completing. This node state will be removed when all of the job's processes have terminated.

- **NODELIST**: Names of the nodes in this partition that are in the state correspond to the current entry.

NOTE: Nodes of the queue can be in different states based on the allocation. In the output shown in Example 1, one node of "chem.q" queue i.e. "node210" is in down state and rests are idle. Thus, more than one entry is listed for single queue.

Example 2:

```
[molly@head1 ~]$ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
all.q*       up   infinite      8/159/0/167  node[1-167]
gpuqueue     up   infinite          0/2/0/2  gpunode[1-2]
phiqueue     up   infinite          0/0/2/2  phinode[1-2]
comp.q       up   infinite        0/37/1/38  node[363-400]
```

- **NODES (A/I/O/T)** column specifies number of nodes by state in the format: "A (allocated)/I(idle)/O(other)/T(total)" for each partition.

2. sbatch : Submit a batch job for execution

Usage:

$ *sbatch <batchscript>*	// submit a given batch script
$ *sbatch –n <no_of_processors> <batchscript>*	// submit a given batch script and allocate specified no. of processors
$ *sbatch –p<partition> <batchscript>*	// submit a given batch script to the specified partition

Example 3:

$ *sbatch –n 24 abc.sh*

This command submits the batch script "abc.sh" and allocates 24 processors to the job. Batch script is explained in more detail under the section - procedure of job submission.

3. **squeue**: Displays the status of the submitted job

Usage:

$*squeue*	// displays status of all jobs in the queue
$*squeue –u <username>*	// displays status of given user's jobs in the queue
$*squeue –u <username> –l*	// displays long output of user's job in the queue

4. **scancel**: Cancel a pending or running job

Usage:

$ *scancel <jobid>*	//Cancel a job with the given jobid

Example 4:

$ *scancel 452*

Here, 452 is the job id generated while submitting the job with sbatch, also the job id can be found with squeue command.

6. Ivy Cluster: How to Login

For getting access to the Ivy Cluster, users have to give the 'User Account Requisition' form available in the service portal of Computer Division (http://csport/). After getting the required permissions, users can access the Ivy Cluster using any ssh client like 'PuTTY' from their desktop using the following details:

IP Address : 10.1.2.3

7. Ivy Cluster: Compilation of Sequential and Parallel Programs

The users can use the available compilers including GNU and Intel compilers for C, C++ and Fortran. As an example, to compile a sequential C program using GNU compiler, user can give the following command:

$ gcc test.c -o test

Similarly, to compile a parallel program using intel MPI compiler, user can give the following command:

$ mpicc test.c -o testMPI

8. Ivy Cluster: Procedure for Parallel Job Submission

Slurm can be used to submit sequential and parallel jobs to the queue. The executables submitted as jobs to slurm can either be user created executables or existing software executables like Computational Fluid Dynamic code, Molecular Dynamics code etc.

Example 5: Consider the user has *hpc_programs* as a sub-directory in his home directory. Also, consider *hpc_programs* directory contains the input files needed by the executable named *paraller_program*. The procedure for running this *parallel_program* as a parallel *slurm job* is as follows:

1. **Creating the slurm script**

   A typical batch script template for the job submission is available at */home/slurm_templet/slurmScript.sh* of the Ivy Cluster. The content of the script is furnished below:

```
#!/bin/bash
#SBATCH --ntasks-per-node 24
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out
#SBATCH -p all.q

INP_FILES="input_file1 input_file2"   #Line No. 7
OUT_FILES="output_file1 output_file2" #Line No. 8

JOB_DIR="$SLURM_JOBID"_$SLURM_JOB_NAME
SCRATCH_DIR=/apps/scratch/`id -gn`/$USER/$JOB_DIR
mkdir $SCRATCH_DIR
```

```
if [ -n INP_FILES ]; then
  cp $INP_FILES $SCRATCH_DIR;
fi
cd $SCRATCH_DIR

mpirun -np $SLURM_NPROCS  <path to executable/executable_name> #Line No. 19

if [ -n OUT_FILES ]; then
  mkdir $JOB_DIR
  cp $OUT_FILES $SLURM_SUBMIT_DIR/$JOB_DIR;
fi
```

Copy that script template "*slurmScript.sh*" from */home/slurm_templet/* to the directory where the input files are located. In this example (Example 5), it is *hpc_programs*.

```
[user@head1 ~]$ cd hpc_programs
[user@head1 hpc_programs]$ cp /home/slurm_templet/slurmScript.sh myParallelJob.sh
```

Edit this newly copied *myParallelJob.sh* script with any text editor and change the following lines:

a. Line no. 7 of myParallelJob.sh:

```
INP_FILES="input_file1 input_file2"   #Line No. 7
```

INP_FILES should contain the list of space separated file names that are used by the *parallel_program* as input. Therefore, the user may modify the file names inside the double quotes as per the program's requirements. For example, if *parallel_program* requires three input files named *points.inp*, *delta* and *tables*, then the line should look like below:

```
INP_FILES="points.inp delta tables" #Line No. 7
```

If no input files are needed for the program, the user may use a null string without any file name or blank space like INP_FILES=""

b. Line no. 8 of myParallelJob.sh:

```
OUT_FILES="output_file1 output_file2" #Line No. 8
```

OUT_FILES should contain the list of space-separated file names that are the output files of the *parallel_program* which the user prefers to copy back. See the subsection "4. Finding the output of your job" below for more details. For example, if the user wants to give only one output file named *finals* that has to be copied back, the line will be:

```
OUT_FILES="finals" #Line No. 8
```

If no output files are expected you may set OUT_FILES=""

c. Line no. 19 of myParallelJob.sh:

```
mpirun -np $SLURM_NPROCS  <path to executable/executable_name> #Line No. 19
```

This line should contain the absolute path of the executable to be run. Users are advised to change this path according to their requirement. In this example, the *paralle_program* is located in the */home/eig/user/hpc_programs/* directory. Therefore, after editing the line, it should look like below:

```
mpirun -np $SLURM_NPROCS /home/eig/user/hpc_programs/parallel_program
```

Note : If the executable path is mentioned using the Linux shell environment variable named 'PATH', then user can avoid mentioning the absolute path of the executable.

d. Lines 2 to 5: SBATCH commands which need not be edited:

```
#SBATCH --ntasks-per-node 24
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out
#SBATCH -p all.q
```

These lines are *sbatch* command parameters and the user can refer manual page of *sbatch* for further details.

Note: All the other lines which are not discussed above should be left as they are. Their purpose is to run the job in the scratch file system allocated to the user and retain only the required files by copying them back to home using OUT_FILES mentioned above.

2. Submitting the slurm job

Now the user can submit the above script (slurm job) using *sbatch* command with –n option to specify the number of processors and –p option to specify the queue (partition), as below:

```
[user@head1 hpc_programs]$ sbatch –n 96 –p comp.q  myParallelJob.sh
Submitted batch job 569
```

NOTE: The command line *sbatch* options take precedence over the options specified in the batch script, so the job will be submitted to "comp.q" (specified with –p option in command line) instead of "all.q" (specified with –p option in the batch script). If no partition is specified either in the command line or in the batch script, the job will be submitted to the default queue i.e. all.q.

3. Finding the status of the slurm job

Status of slurm jobs can be seen by issuing the 'squeue' command as follows:

```
[user@head1 inputs]$ squeue -u user
JOBID    PARTITION    NAME             USER   ST   TIME    NODES   NODELIST(REASON)
569      comp.q       myParallelJob.sh user   R    0:18    4       node[363-366]
```

4. Finding the output of the slurm job

A job submitted to slurm may generate two kinds of output.

a. The standard output and standard error files created by slurm:
   The standard output and standard error messages which usually come to the terminal screen will be redirected to the files named *job.<JobID>.out* and *job.<JobID>.err* and placed in the directory where you submit the job using the *sbatch* command. In the example (Example 5), it is *hpc_programs* directory.
b. Output files those are mentioned by the user in the OUT_FILES part of *myParallelJob.sh* script will be copied to a directory created with the name *<JobID>_<JobName>* where the user has submitted the job.

In this example the JobID is 569, therefore the standard output and standard error files are named as *job.569.out* and *job.569.err* respectively. The output directory is *569_myParallelJob.sh*. These details are listed below:

```
[user@head1 hpc_programs]$ ls
569_myParallelJob.sh    delta          parallel_program
points.inp              job.569.err    job.569.out
tables
```

9. Ivy Cluster: Procedure for Sequential Job Submission

As mentioned earlier, slurm can be used to submit sequential and parallel jobs to the queue. The procedure for sequential job submission is almost similar to that of parallel job submission, described in the previous section. The only difference is in calling the executable from the slurm script. Therefore, the difference is confined to the Line No. 19 of the slurm script given in Example 5. This line should contain the absolute path of the executable to be run. Users are advised to change this path according to their requirement. For example, the executable named *sequential_program* is located in the */home/eig/user/hpc_programs/* directory. Therefore, after editing the line, it should look like below:

```
/home/eig/user/hpc_programs/sequential_program
```

Note: If the executable path is mentioned using the Linux shell environment variable named 'PATH', then user can avoid mentioning the absolute path of the executable.

10. Summary

An overview about the Ivy Cluster installed at Computer Centre of IGCAR is given in the document. The details about storage architecture, parallel file system, software used, and resource management of the Ivy Cluster are given in the document. This document helps the users to familiarize the software environment in order to utilize the cluster system efficiently. It also gives the systematic procedure for the beginners to do job submission and management in the Ivy Cluster.