

Kohinoor2 Cluster User/Admin Manual

Intel Xeon 512 Cores HPC Cluster
At
Tata Institute of Fundamental Research (TIFR)
Hyderabad



By



Locuz Enterprise Solutions Ltd
401, Krishe Sapphire,
Main Road, Cyberabad,
Madhapur, Hyderabad- 500081.

Contents

1. About Document
 - 1.1. Document History
 - 1.2. Locuz Contact Details
 - 1.3. Target Group
 - 1.4. Typographic Conventions
2. Introduction
 - 2.1. About Clustering
 - 2.2. High-Performance Computing (HPC)
 - 2.3. HPC Stack
3. KOHINOOR2 Implementation
 - 3.1. Servers Hardware Details
 - 3.2. Switches and Connectivity
 - 3.3. Storage
 - 3.4. Operating System - CentOS
 - 3.5. Cluster Management
 - 3.6. Software - Compilers, Libraries, Visualization and Applications
 - 3.7. HPC Scheduler - SGE
 - 3.8. Troubleshooting SGE
 - 3.9. Monitoring Tool - Ganglia
 - 3.10. System Management - server suite
4. Startup & shutdown procedures
 - 4.1. Shutdown sequence
 - 4.2. Startup sequence
5. Support Escalation Matrix

Chapter 1

About Document

1.1. Document History

Version	Release Date	Prepared by
1.0	5 th May 2014	Locuz Enterprise Solution Ltd

1.2. Locuz Contact Details

Please refer page number for “Support Escalation Matrix”

1.3. Target Group

This document is designed for End-Users for reference of technical and usage details. In this document, it has been tried to provide a clear-cut theoretical picture and practical approach to use the cluster in a better way.

1.4. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows:

Arial 16 Bold Centered

Used to specify Chapter Headings

Trebuchet MS 12 Bold

Used to specify side-headings and sub-headings in a chapter with/without sub-heading number

Trebuchet MS 12 Bold

Trebuchet MS 12 Bold

Used to specify the heading that is again a sub-heading of a side-heading

Courier New 12 Bold or `Courier New 12`

Used to specify the system commands and output



Chapter 2

Introduction

2.1. About Clustering

A computer cluster is a group of linked computers, working together closely thus in many respects forming a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Cluster categorizations

a. Compute clusters:

Often clusters are used primarily for computational purposes, rather than handling IO-oriented operations such as web service or databases. For instance, a cluster might support computational simulations of weather or vehicle crashes. The primary distinction within computer clusters is how tightly-coupled the individual nodes are. For instance, a single computer job may require frequent communication among nodes - this implies that the cluster shares a dedicated network, is densely located, and probably has homogenous nodes. This cluster design is usually referred to as Beowulf Cluster. The other extreme is where a computer job uses one or few nodes, and needs little or no inter-node communication. This latter category is sometimes called "Grid" computing. Tightly-coupled compute clusters are designed for work that might traditionally have been called "supercomputing". Middleware such as MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) permits compute clustering programs to be portable to a wide variety of clusters.

2.2. High-Performance Computing (HPC):

High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems. Today, computer systems approaching the teraflops-region are counted as HPC-computers.

HPC integrates systems administration and parallel programming into a multidisciplinary field that combines digital electronics, computer architecture, system software, programming languages, algorithms and computational techniques. HPC technologies are the tools and systems used to implement and create high performance computing systems. Recently, HPC systems have shifted from supercomputing to computing clusters and grids. Because of the need of networking in clusters and grids, High Performance Computing Technologies are being promoted by the use of a collapsed network backbone, because the collapsed backbone

architecture is simple to troubleshoot and upgrades can be applied to a single router as opposed to multiple ones.

The term is most commonly associated with computing used for scientific research or computational science. A related term, high-performance technical computing (HPTC), generally refers to the engineering applications of cluster-based computing (such as computational fluid dynamics and the building and testing of virtual prototypes). Recently, HPC has come to be applied to business uses of cluster-based supercomputers, such as data warehouses, line-of-business (LOB) applications, and transaction processing.

High-performance computing (HPC) is a term that arose after the term "supercomputing." HPC is sometimes used as a synonym for supercomputing; but, in other contexts, "supercomputer" is used to refer to a more powerful subset of "high-performance computers," and the term "supercomputing" becomes a subset of "high-performance computing." The potential for confusion over the use of these terms is apparent.

Because most current applications are not designed for HPC technologies but are retrofitted, they are not designed or tested for scaling to more powerful processors or machines. Since networking clusters and grids use multiple processors and computers, these scaling problems can cripple critical systems in future supercomputing systems. Therefore, either the existing tools do not address the needs of the high performance computing community or the HPC community is unaware of these tools. A few examples of commercial HPC technologies are the simulation of car crashes for structural design, molecular interaction for new drug design and the airflow over automobiles or airplanes. In government and research institutions, scientists are simulating galaxy creation, fusion energy, and global warming, as well as working to create more accurate short- and long-term weather forecasts.

2.3. HPC Stack:

HPC Stack is a cluster components' stack which explains about different components which uses in HPC cluster implementation and its dependencies on one another. Figure 2.1 shows a brief view about components and its dependency.

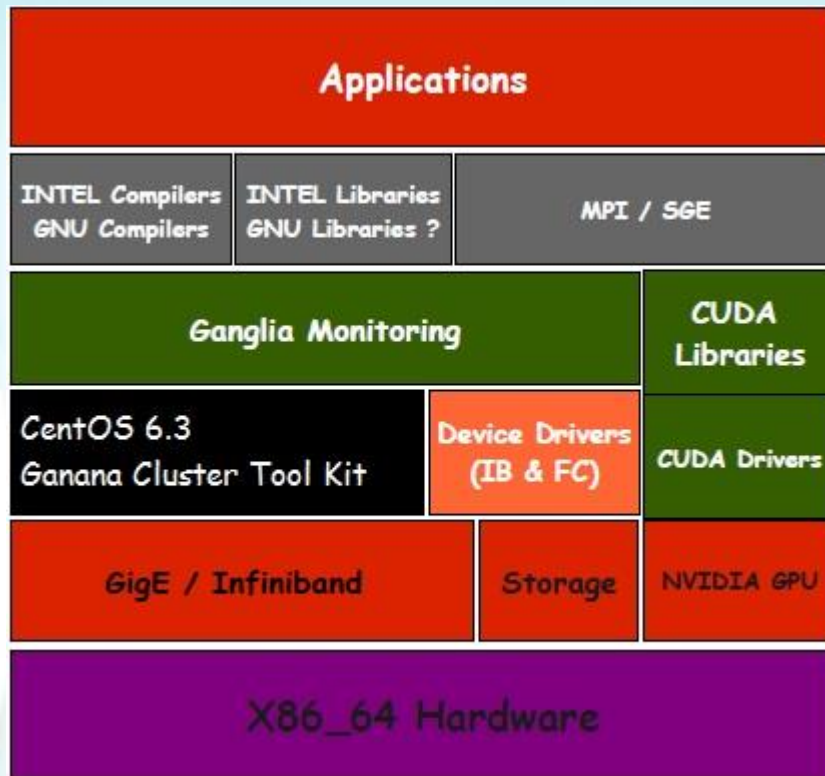


Figure 2.1: HPC Stack

a. *Hardware:*

- Servers:** 32-bit/64-bit servers, which generally have rich hardware resources like Multi-core processors, RAM, Storage, etc.
- Storage:** Storage may be internal, which is attached to Master node internally i.e. internal HDDs or external. Generally external storage can be configured for home directory and backup
- Ethernet:** Ethernet provides a conventional intercommunication among all nodes. This requires Ethernet NIC, Ethernet switch and its cables.
- Infiniband:** Infiniband gives much faster communication than conventional Ethernet communication. It improves the job execution speed and inter-node communication. To configure Infiniband, setup requires three things - Infiniband Cards, Infiniband switch and its cables.

HPC hardware is combination of at least one Master Node, many Compute Nodes, Storage, Network and Storage Switches, connectivity channels/cables, etc. Master node is a special server which plays an administrator role for whole cluster. It provides a centralized facility to

create/delete users, create ACLs, and define roles for different compute nodes, installation of software and many administrative activities. It is mandatory for any HPC cluster that one node should be master node but according to requirement, it can be configuring more than one master node.

In Kohinoor implementation, the cluster has been configured with Single master nodes configuration.

b. *Operating System:*

Operating System plays a great foundational role in cluster configuration. OS manages all resources properly according to specifications and configuration. Apart from hardware management, it is mandatory to create and configure some special things like ssh password-free environment, centralized home directory management, synchronization of user information, facility to execute commands concurrently across the cluster nodes, etc.

On top of Kohinoor2 server hardware, Ganana cluster toolkit has been installed with CentOS 6.3 Operating system.

c. *Libraries:*

Library is a collection of resources used to develop software. These may include pre-written code and subroutines, classes, values or type specifications. Libraries contain code and data that provide services to independent programs. This allows the sharing and changing of code and data in a modular fashion. Some executables are both standalone programs and libraries, but most libraries are not executable. Executables and libraries make references known as links to each other through the process known as linking, which is typically done by a linker.

Here in HPC Stack, libraries mean development libraries both serial and parallel which are associated with compilers and other HPC programs to run jobs.

Originally, only static libraries existed. A static library, also known as an archive, consists of a set of routines which are copied into a target application by the compiler, linker, or binder, producing object files and a stand-alone executable file. This process, and the stand-alone executable file, is known as a static build of the target application. Actual addresses for jumps and other routine calls are stored in a relative or symbolic form which cannot be resolved until all code and libraries are assigned final static addresses.

Dynamic linking involves loading the subroutines of a library into an application program at load time or run-time, rather than linking them in at compile time. Only a minimum amount of work is done at compile time by the linker; it only records what library routines the program needs and the index names or numbers of the routines in the library. The majority of the work of

linking is done at the time the application is loaded (load time) or during execution (runtime).

In addition to identifying static and dynamic loading, computer scientists also often classify libraries according to how they are shared among programs. Dynamic libraries almost always offer some form of sharing, allowing the same library to be used by multiple programs at the same time. Static libraries, by definition, cannot be shared.

With CentOS 6.3 on Kohinoor2, glibc is installed for OS and developmental activities.

The GNU C Library, commonly known as glibc, is the C standard library released by the GNU Project. Originally written by the Free Software Foundation (FSF) for the GNU operating system, the library's development has been overseen by a committee since 2001, with Ulrich Drepper from Red Hat as the lead contributor and maintainer.

d. *Compilers:*

A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form known as object code). The most common reason for wanting to transform source code is to create an executable program.

The GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU tool chain. As well as being the official compiler of the unfinished GNU operating system, GCC has been adopted as the standard compiler by most other modern Unix-like computer operating systems, including Linux, the BSD family and Mac OS X.

Originally named the GNU C Compiler, because it only handled the C programming language, GCC 1.0 was released in 1987, and the compiler was extended to compile C++ in December of that year. Front ends were later developed for FORTRAN, Pascal, Objective-C, Java, and Ada, among others.

e. *Scheduler:*

A job scheduler is a software application that is in charge of unattended background executions, commonly known for historical reasons as batch processing. Synonyms are batch system, Distributed Resource Management System (DRMS), and Distributed Resource Manager (DRM). Today's job schedulers typically provide a graphical user interface and a single point of control for definition and monitoring of background executions in a distributed network of computers. Increasingly job schedulers are required to orchestrate the integration of real-time business activities with traditional background IT

processing, across different operating system platforms and business application environments.

Basic features expected of job scheduler software are: Interfaces which help to define workflows and/or job dependencies, automatic submission of executions, interfaces to monitor the executions and priorities and/or queues to control the execution order of unrelated jobs.

An important niche for job schedulers is managing the job queue for a cluster of computers. Typically, the scheduler will schedule jobs from the queue as sufficient resources (cluster nodes) become idle. Some widely used cluster batch systems are Sun Grid Engine, Portable Batch System, Load Leveler, Condor, OAR and Simple Linux Utility for Resource Management.

For Kohinoor2 resources management, Sun Grid Engine GE2011.p11 is an open-source HPC job scheduler. Now it is Oracle Grid Engine, previously known as Sun Grid Engine (SGE) is an open source batch-queuing system, developed and supported by Sun Microsystems. Sun once also sold a commercial product based on SGE, known as N1 Grid Engine (N1GE). SGE is typically used on a computer farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses.

f. *Monitoring Tools:*

A system monitor or monitoring tool is hardware or software-based system used to monitor resources and performance in a computer system.

Ganglia is a scalable distributed system monitor tool 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.

g. *MPI:*

Message Passing Interface (MPI) is an API specification that allows processes to communicate with one another by sending and receiving messages. Besides many other applications, it is a de facto standard for parallel programs running on computer clusters and supercomputers, where the cost of accessing non-local memory is high. MPI was created since 1992 by William Gropp, Ewing Lusk and others, a first standard appeared in 1994.

MPI is a language-independent communications protocol used to program parallel computers. Both point-to-point and collective communication are supported. MPI "is a message-passing application programmer interface,

together with protocol and semantic specifications for how its features must behave in any implementation." MPI's goals are high performance, scalability, and portability. MPI remains the dominant model used in high-performance computing today.

At present, the standard has several popular versions: version 1.3 (shortly called MPI-1), which emphasizes message passing and has a static runtime environment, and MPI-2.2 (MPI-2), which includes new features such as parallel I/O, dynamic process management and remote memory operations.[5] MPI-2's LIS specifies over 500 functions and provides language bindings for ANSI C, ANSI Fortran (Fortran90), and ANSI C++. Object interoperability was also added to allow for easier mixed-language message passing programming. A side-effect of MPI-2 standardization (completed in 1996) was clarification of the MPI-1 standard, creating the MPI-1.2.

Note that MPI-2 is mostly a superset of MPI-1, although some functions have been deprecated. MPI-1.3 programs still work under MPI implementations compliant with the MPI-2 standard.

Open MPI is an Message Passing Interface (MPI) library project combining technologies and resources from several other projects (FT-MPI, LA-MPI, LAM/MPI, and PACX-MPI). It is used by many TOP500 supercomputers including Roadrunner, which was the world's fastest supercomputer from June 2008 to November 2009, and K computer, the fastest supercomputer since June 2011.

h. *Applications:*

Application program can be any program which can either run on individually on different nodes or on across the node in a parallelized manner. HPC applications are programs which may be designed by end-user or purchased from third party software vendors. Applications may be serial or parallel but this is all depends on the end-user and his requirement which decides.

LOCUZ

Chapter 3

KOHINOOR2 Implementation



1. IB Switch
2. Ethernet Switch
3. Compute nodes
4. Monitor
5. GPU nodes
6. Master Node
7. Storage

Figure 3.1: KOHINOOR2 Rack Setup View

3.1. Servers Hardware Details

In KOHINOOR2 cluster, there is a single master node, 32 compute nodes. The following are the hardware details of Servers:

Master Node:

Hardware Details - Fujitsu RX300 S7	
Processor	2 x Intel Xeon 8-core 2.7 GHz
RAM	64 GB DDR3
HDD	2 x 1 GB - 7200 RPM
Ethernet	4 ports
Serial Management	1 port
Net Management	1 port
Infiniband	1 port
FC	2 ports
Optical Drive	DVD Drive
OS Details	
OS	CentOS 6.3
Kernel	2.6.32-279.el6.x86_64
Partition Details	
/	493 GB
Swap	32 GB
/hpcapps	393 GB
/home	22 TB
/data	24 TB
/scratch	5.4 TB

Compute Nodes: 32 Compute nodes.

Hardware Details - IBM iDataplex DX 360 M4	
Processor	2 x Intel Xeon 8-core 2.4 GHz
RAM	32 GB DDR3
HDD	1 x 500 GB - 7200 RPM
Ethernet	2 ports
Net Management	1 port
InfiniBand	1 port
OS Details	
OS	CentOS 5.8
Kernel	2.6.18-308.4.1.el5
Partition Details	
/	419GB

Swap	32 GB
/boot	1GB
/export/data	190GB (nfs from master)
/scratch	12TB (nfs from master)
/scratch2	13TB (nfs from master)

3.2. Switches and Connectivity:

In Kohinoor implementation, two types of switches has been used - Ethernet, Infiniband switches.

Ethernet Switch:

In Kohinoor implementation, 24 Port Gigabit Dlink Switch has been used to configure the private network and in figure 3.2, Ethernet connectivity has been explained:

Dlink DGS-1210-24	
Type	Gigabit Switch
Speed	10/100/1000 Mbps
No.of Ports	24
Connectivity	Nodes are connected using CAT 6

Infiniband Switch:

In Kohinoor, one Infiniband switch with 36 ports for HPC Application/MPI communication. All the nodes including master have been IBSwitch.

The complete connectivity diagram among all hardware components has been explained by a schematic diagram in figure 3.3

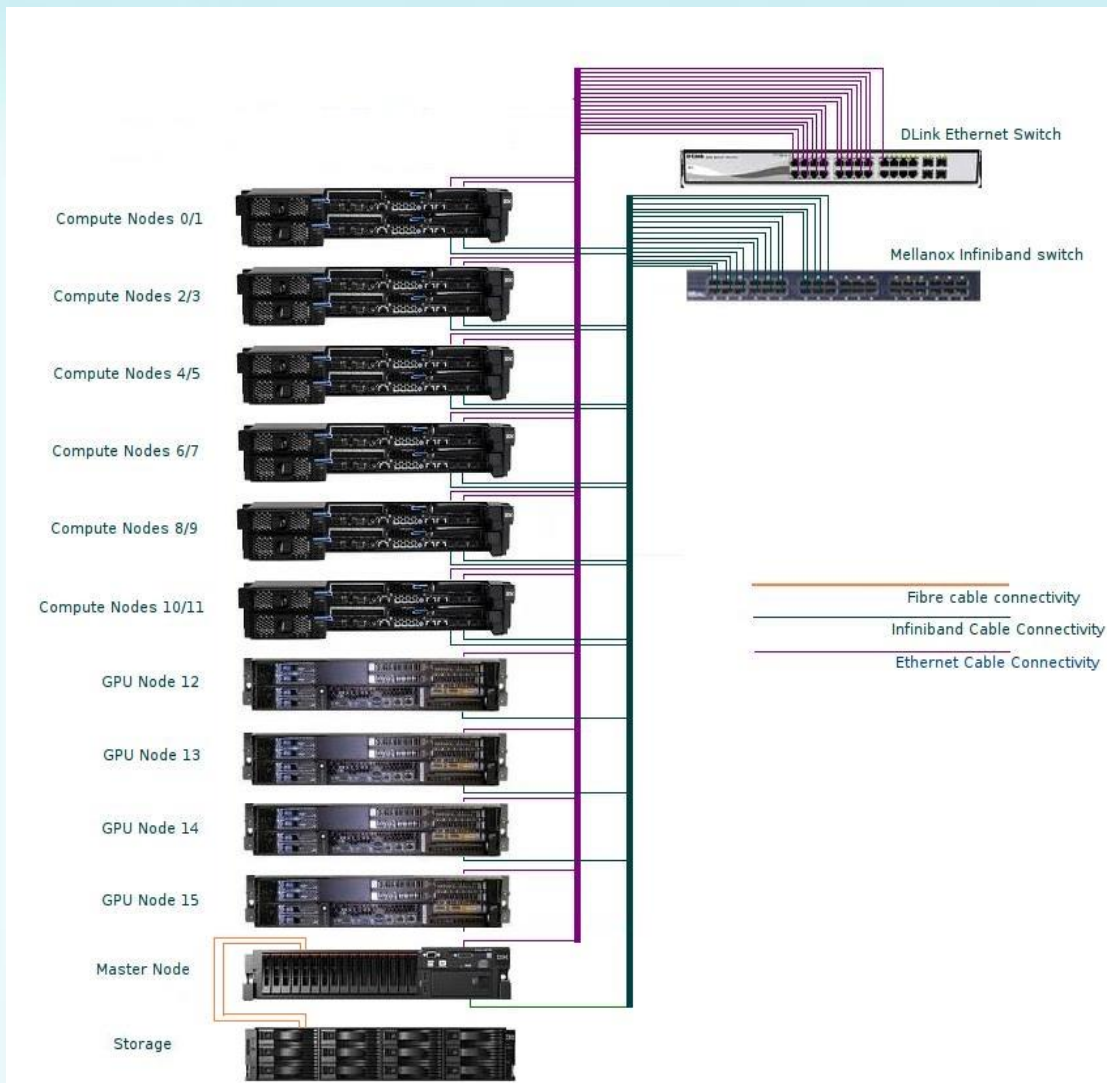


Figure 3.3: Kohinoor2 Schematic Diagram

The connectivity as explained above, the IP configuration details are:

SNo	Host Name	Gigabit Eth IP	IB IP	Mgmt IP	TIFR LAN IP
1	kohinoor2	192.168.10.1	192.168.20.1	192.168.10.101	172.16.10.22
2.	node1	192.168.10.21	192.168.20.21	192.168.10.121	
3	node2	192.168.10.22	192.168.20.22	192.168.10.122	
4	node3	192.168.10.23	192.168.20.23	192.168.10.123	
5	node4	192.168.10.24	192.168.20.24	192.168.10.124	
6	node5	192.168.10.25	192.168.20.25	192.168.10.125	

7	node6	192.168.10.26	192.168.20.26	192.168.10.126	
8	node7	192.168.10.27	192.168.20.27	192.168.10.127	
9	node8	192.168.10.28	192.168.20.28	192.168.10.128	
10	node9	192.168.10.29	192.168.20.29	192.168.10.129	
11	node10	192.168.10.30	192.168.20.30	192.168.10.130	
12	node11	192.168.10.31	192.168.20.31	192.168.10.131	
13	node12	192.168.10.32	192.168.20.32	192.168.10.132	
14	node13	192.168.10.33	192.168.20.33	192.168.10.133	
15	node14	192.168.10.34	192.168.20.34	192.168.10.134	
16	node15	192.168.10.35	192.168.20.35	192.168.10.135	
17	node16	192.168.10.36	192.168.20.36	192.168.10.136	
18	node17	192.168.10.37	192.168.20.37	192.168.10.137	
19	node18	192.168.10.38	192.168.20.38	192.168.10.138	
20	node19	192.168.10.39	192.168.20.39	192.168.10.139	
21	node20	192.168.10.40	192.168.20.40	192.168.10.140	
22	node21	192.168.10.41	192.168.20.41	192.168.10.141	
23	node22	192.168.10.42	192.168.20.42	192.168.10.142	
24	node23	192.168.10.43	192.168.20.43	192.168.10.143	
25	node24	192.168.10.44	192.168.20.44	192.168.10.144	
26	node25	192.168.10.45	192.168.20.45	192.168.10.145	
27	node26	192.168.10.46	192.168.20.46	192.168.10.146	
28	node27	192.168.10.47	192.168.20.47	192.168.10.147	
29	node28	192.168.10.48	192.168.20.48	192.168.10.148	
30	node29	192.168.10.49	192.168.20.49	192.168.10.149	
31	node30	192.168.10.50	192.168.20.50	192.168.10.150	
32	node31	192.168.10.51	192.168.20.51	192.168.10.151	
33	node32	192.168.10.52	192.168.20.52	192.168.10.152	
33	Storage			192.168.10.6	

3.3. Storage:

Fujitsu Eternus DX 60 S2 Storage is a Dual Controller Storage has been configured. 192.168.10.5 is the storage IP assigned.

Open browser and give the storage management ip 192.168.10.6
Enter the password which is same as root password to login in. From there we can access the Array management console by clicking on Manage Storage array option in Enterprise console

Fujitsu Eternus DX 60 S2 Storage is a Dual Controller with 8 GB 4 ports Daughter Card and 4 SFP are attached. It has 1GB of Data cache & 1GB of Cache backup device and 24 number of 3TB each NL SAS Drive.

Created a seven logical array with RAID-5 associated 24 numbers of NL SAS drives, Total Storage capacity is 50TB associated 21 numbers of Data Disk and 3 Parity Disk in a RAID group.

Three luns has been created, its capacity 21.5TB, 24.5 and 4.5TB each and assigned to Master Server. The storage is connected to the master server through two fc cables.

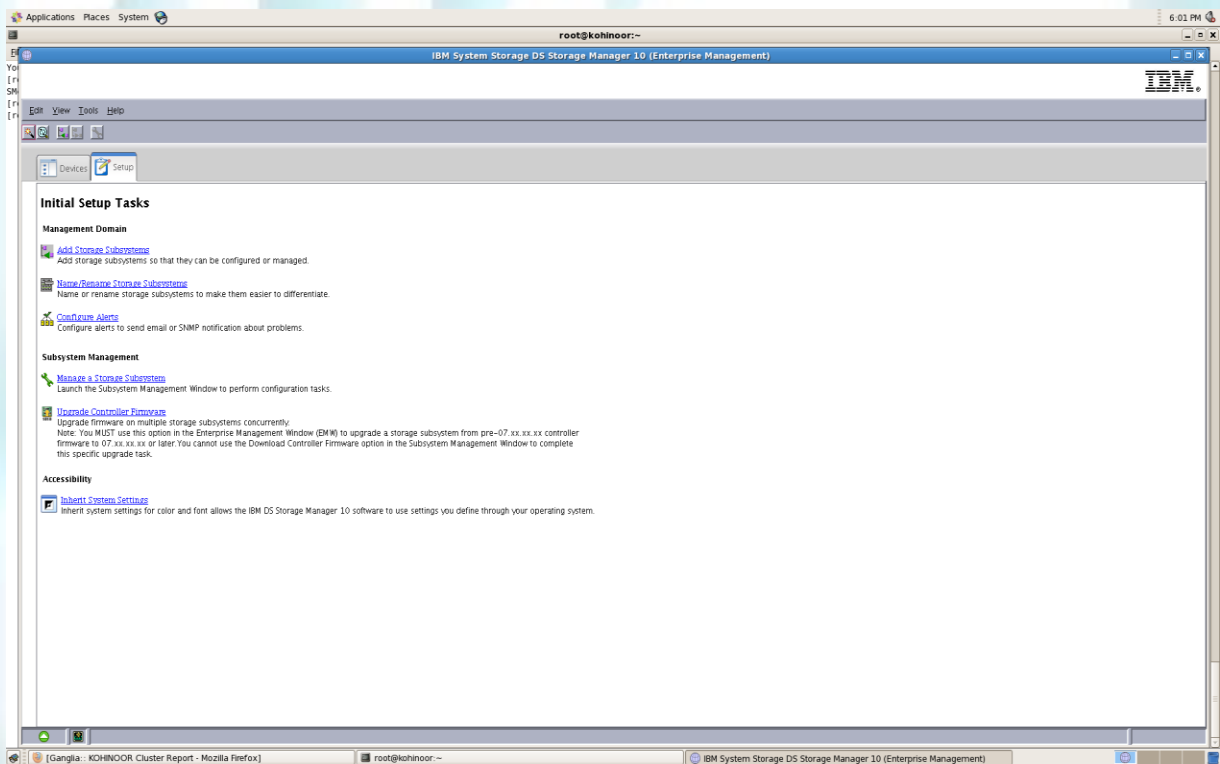


Fig: Enterprise Console Management

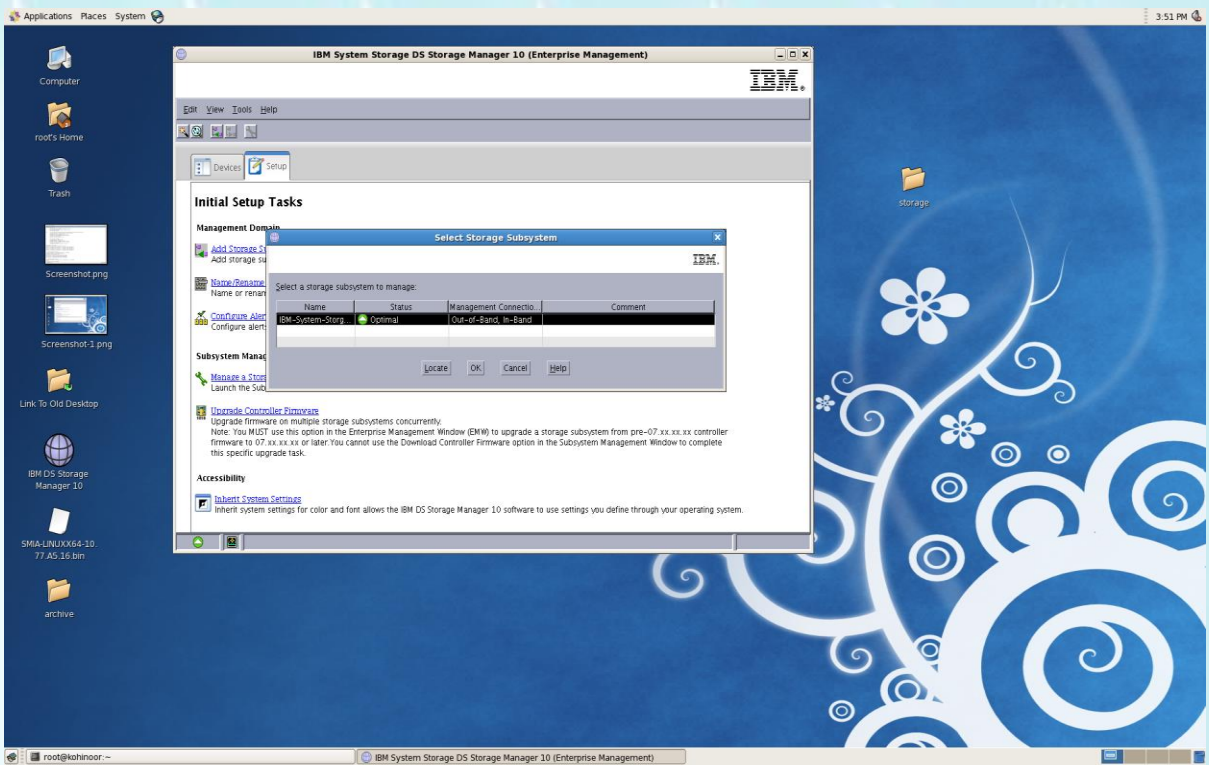


Fig: Array Management console.

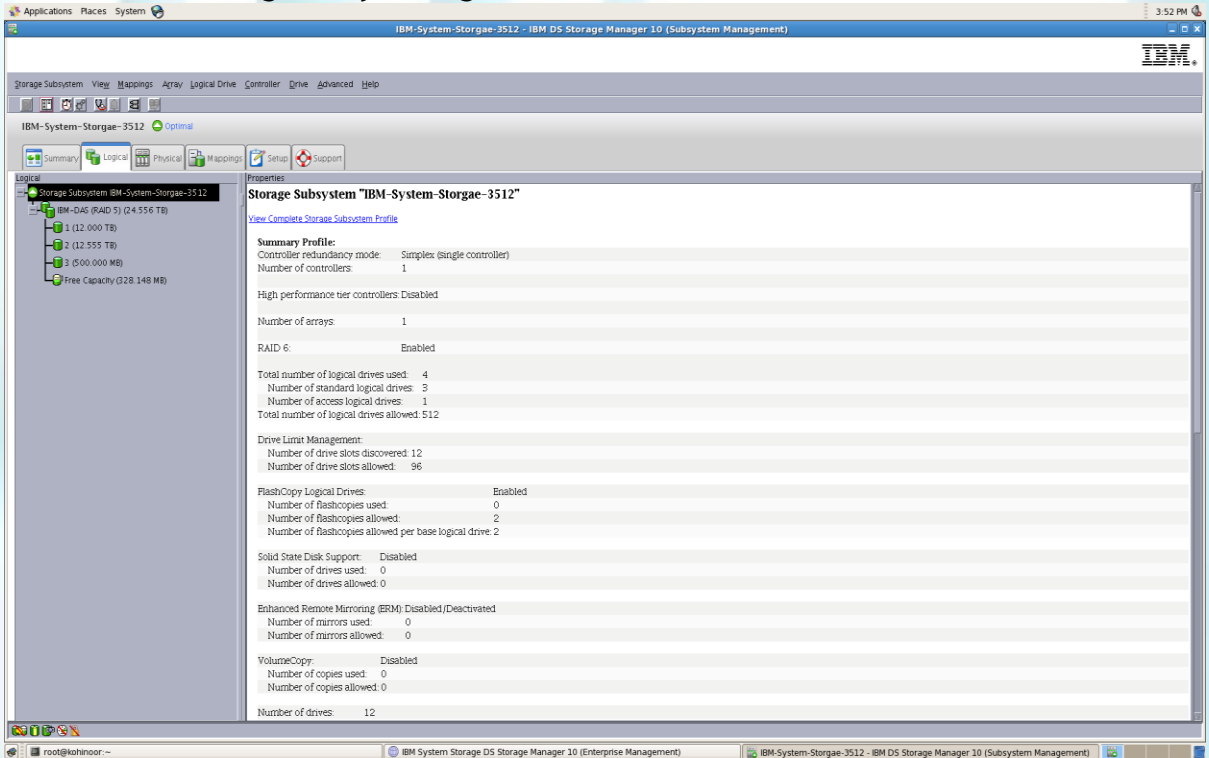


Fig: Logical Volume view of the Storage.

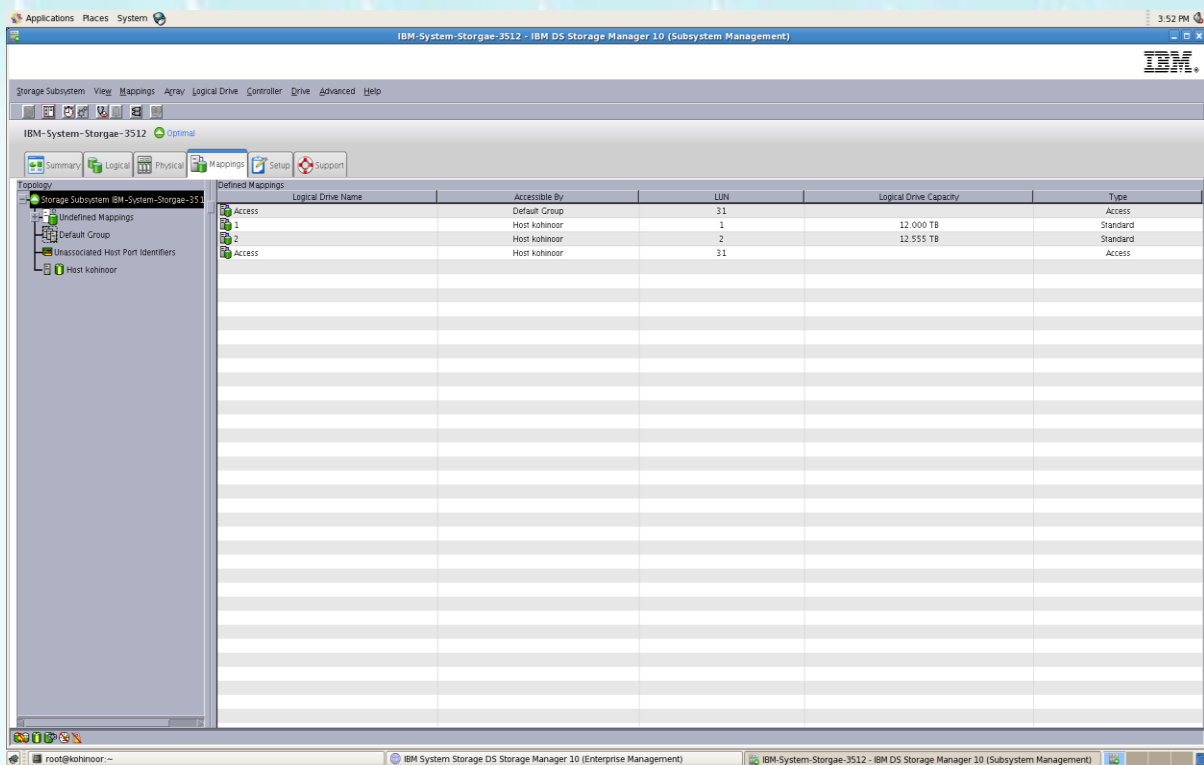


Fig: Logical volume mapping to Host.

3.4. Operating System - CentOS:

An operating system is software, consisting of programs and data that runs on computers manages computer hardware resources, and provides common services for execution of various application software. Operating system is the most important type of system software in a computer system. Without an operating system, a user cannot run an application program on their computer, unless the application program is self booting.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between application programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently call the OS or be interrupted by it. Operating systems are found on almost any device that contains a computer—from cellular phones and video game consoles to supercomputers and web servers.

Examples of popular modern operating systems are: BSD, Linux, Mac OS X, Microsoft Windows and UNIX.

Linux refers to the family of Unix-like computer operating systems using the Linux kernel. Linux can be installed on a wide variety of computer hardware, ranging from mobile phones, tablet computers, routers, and video game consoles,

to mainframes and supercomputers. Linux is a leading server operating system, and runs the 10 fastest supercomputers in the world.

The development of Linux is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed, both commercially and non-commercially, by anyone under licenses such as the GNU General Public License. Typically Linux is packaged in a format known as a Linux distribution for desktop and server use. Some popular mainstream Linux distributions include Debian (and its derivatives such as Ubuntu), Fedora and openSUSE. Linux distributions include the Linux kernel and supporting utilities and libraries to fulfill the distribution's intended use.

3.5. Cluster Management

Login from Windows Machine

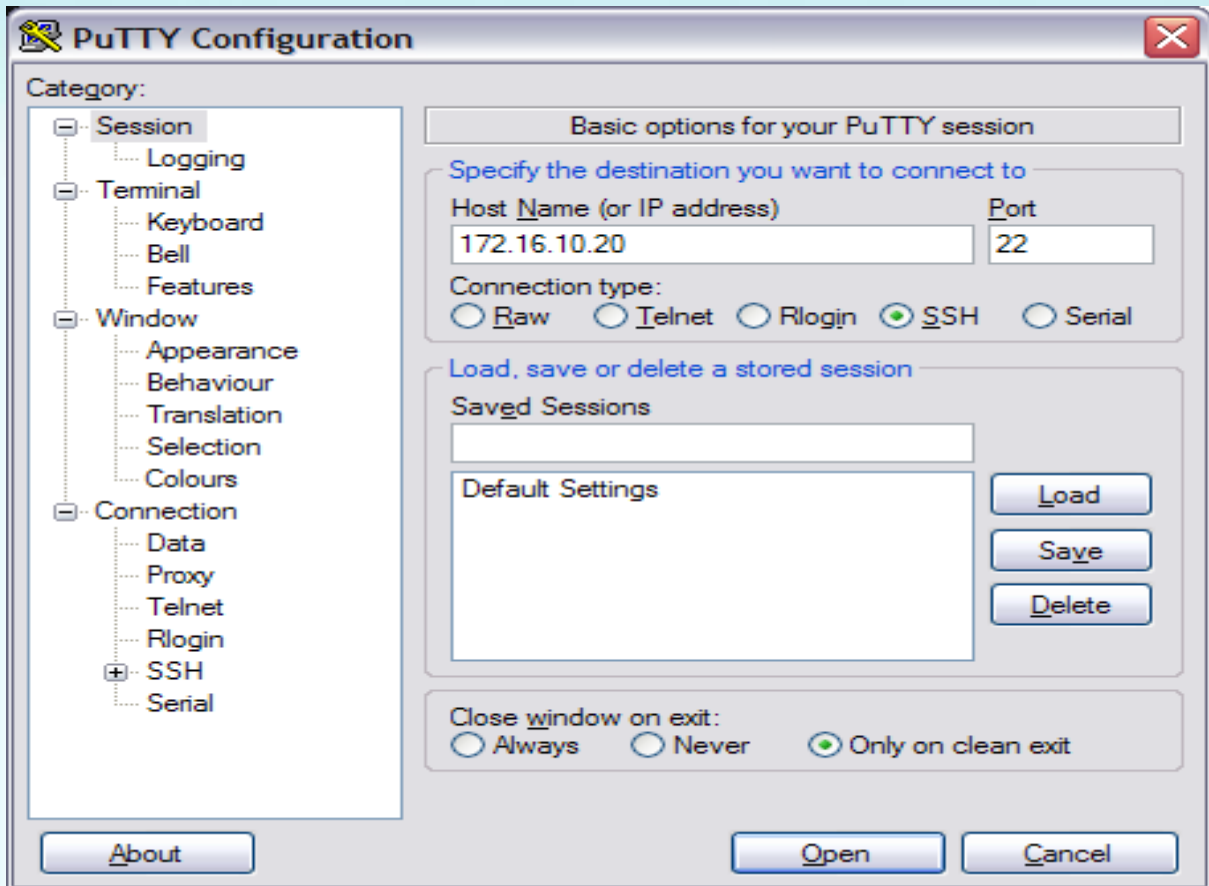
Step1:

Get the latest version of [putty](#)

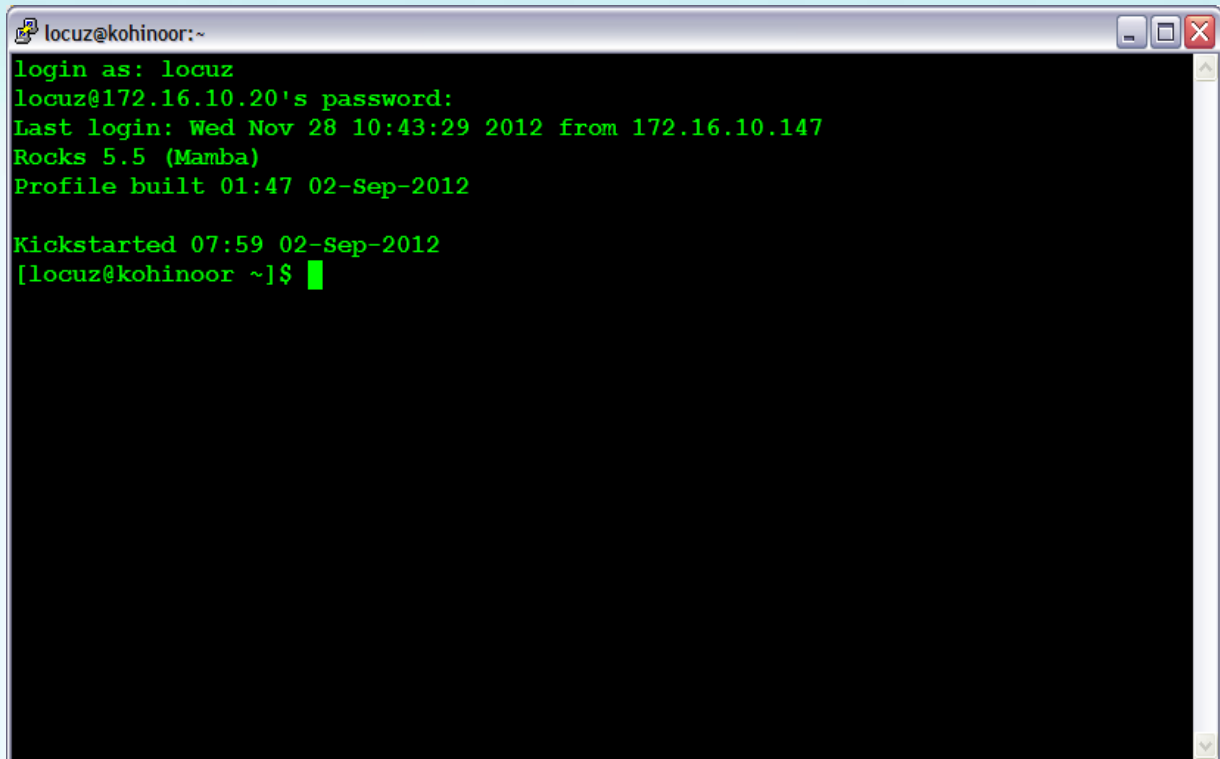
Step2:

Run putty and try to ssh connection to <cluster public IP Address>

LOCUZ



LOCUIZ



```
locuz@kohinoor:~  
login as: locuz  
locuz@172.16.10.20's password:  
Last login: Wed Nov 28 10:43:29 2012 from 172.16.10.147  
Rocks 5.5 (Mamba)  
Profile built 01:47 02-Sep-2012  
  
Kickstarted 07:59 02-Sep-2012  
[locuz@kohinoor ~]$
```

Logged in as a user locuz

Login from Linux machine

```
[root@localhost ~] ssh locuz@172.16.10.22
```

Password:

```
[locuz@kohinoor ~]$
```

Creating User account into the Kohinoor cluster

User Creation:

Create a user account and propagate the information to the compute nodes with:

```
# useradd <username>  
# passwd <username>  
New password:  
Re-enter password:
```

It creates a user's home directory at location `/home/$USER`

```
# make /var/yp
```

Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.) on all known hosts. Also, restart autofs on all known hosts.

User deletion:

```
# userdel <username>
```

Then run the

```
# make /var/yp
```

Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.)

Copy a file or directory of every compute node:

```
$ cpush <file> /tmp
```

Copies file to the /tmp directory of every compute node

```
$ cget <file> /tmp
```

Copy a directory to the /tmp directory of every compute node

Copy a file to defined compute node:

```
$ scp <filename> <compute node name>:/tmp
```

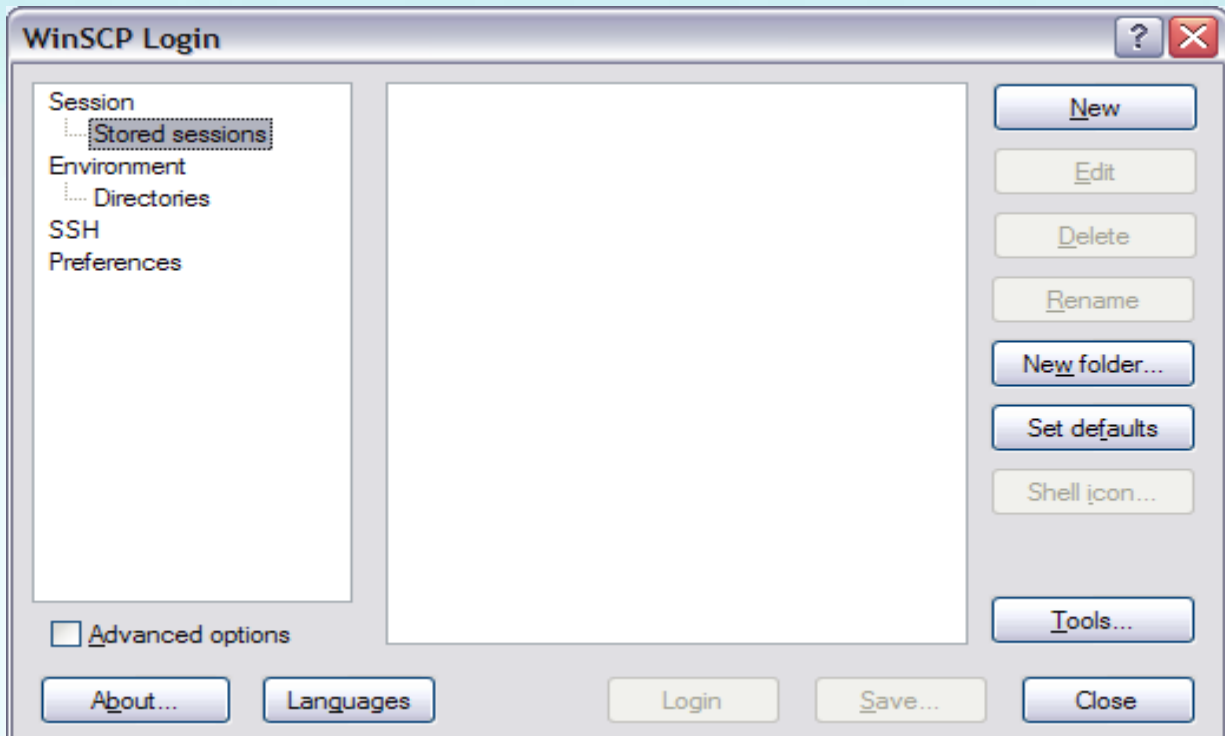
```
$ scp -r <directory> <compute node name>:/tmp
```

Copy a data from end-user Windows machine to cluster

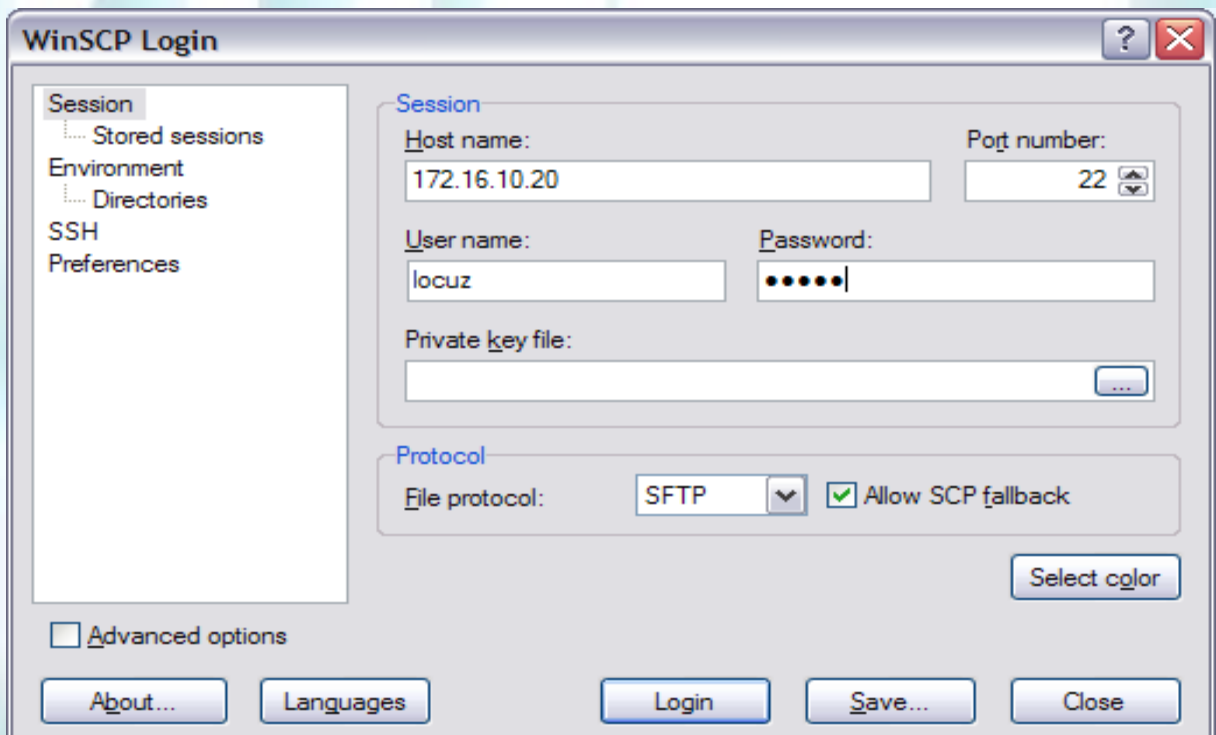
Step 1:

Install and start WinSCP, then following screen is shown. Click 'New' button.

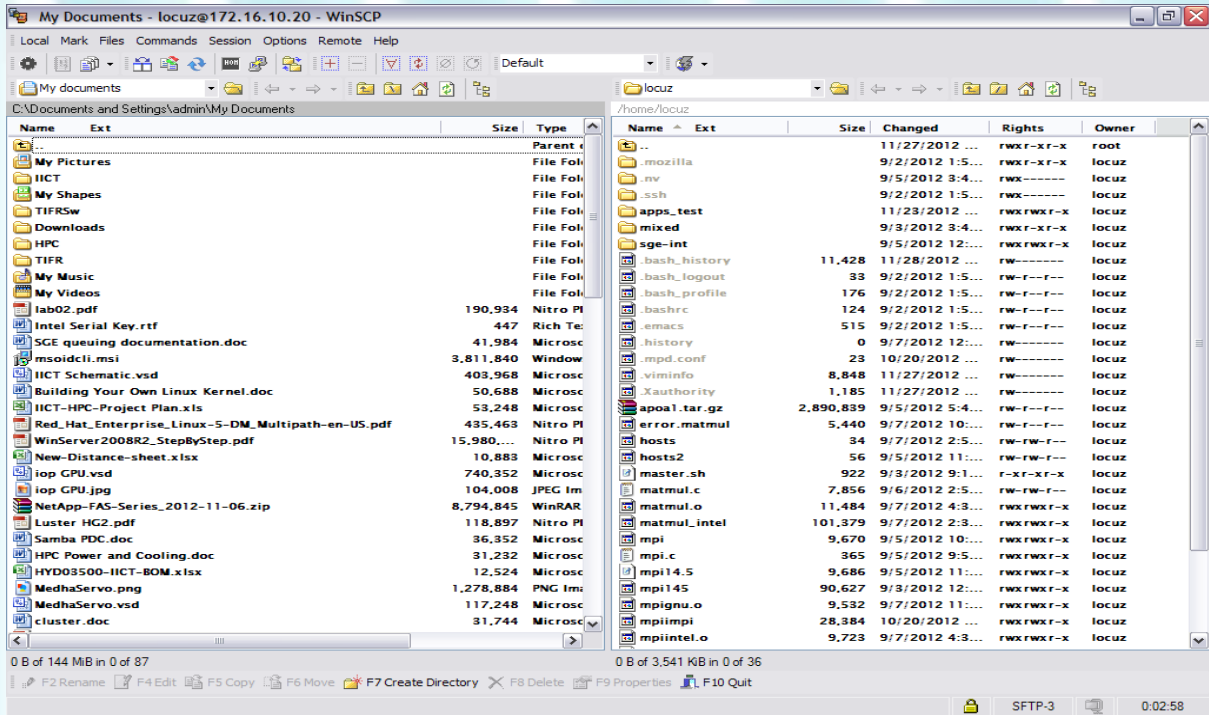
LOCUZ

**Step 2:**

Following screen is shown. Input information's to login like below.



Step 3: It's possible to upload or download files



Copy a data from end-user Linux machine to cluster

```
[root@localhost ~] scp <filename> locuz@172.16.10.22
```

Coping a file into user locuz home directory

```
[root@localhost ~] scp -r <directory> locuz@172.16.10.22
```

Coping a directory into user locus home directory

Rocks run host:

Used to run any command on compute node of the cluster

```
$ cexec "<command>"
```

Run the particular command on all nodes

3.6. Software - Compilers, Libraries, Visualization & Applications

Compilers:

a. GNU compilers

C = /usr/bin/gcc
 C++ = /usr/bin/g++
 FORTRAN = /usr/bin/gfortran, /usr/bin/f95

b. MPI (Parallel) compilers

mpicc = /hpcapps/mpi/openmpi/gcc/1.6.5/bin/mpicc
 mpic++ = /hpcapps/mpi/openmpi/gcc/1.6.5/bin/mpic++
 mpif77 = /hpcapps/mpi/openmpi/gcc/1.6.5/bin/mpif77
 mpif90 = /hpcapps/mpi/openmpi/gcc/1.6.5/bin/mpif90

Applications:

List of Application

Sl.No	Application Name	CPU	
A	LAMMPS	Yes	
B	NAMD	Yes	
C	GROMACS	Yes	
D	HOOMD	Yes	
E	CHARMM	Yes	

A. LAMMPS

CPU:

Location = /export/data/apps/lammps-cpu
Executable file = /export/data/apps/lammps-cpu/30Aug12/
bin/lmp_openmpi

GPU:

Location = /export/data/apps/lammps-cuda
Executable file = /export/data/apps/lammps-cuda/bin/lmp_cuda

B. NAMD

CPU:

Location = /export/data/apps/NAMD_2.9_Linux-x86_64-
ibverbs
Executable file = /export/data/apps/NAMD_2.9_Linux-x86_64-
ibverbs/namd2

GPU:

Location = /export/data/apps/NAMD_2.9_Linux-x86_64-
multicore-CUDA
Executable file = /export/data/apps/NAMD_2.9_Linux-x86_64-
multicore-CUDA/namd2

C. GROMACS

CPU:

Location = /export/data/apps/gromacs
Executable file = /export/data/apps/gromacs/4.5.5/intel/
bin/mdrun_mpi

GPU:

Location = /export/data/apps/gromacs-gpu
Executable file = /export/data/apps/gromacs-gpu/mdrun-openmm

D. CPMD

CPU:

Location = /export/data/apps/cpmd
Executable file = /export/data/apps/cpmd/3.15.3/intel-
ompi/cpmd.x

E. QUANTUM ESPRESSO

CPU:

Location = /export/data/apps/q-espresso
Executable file = /export/data/apps/q-espresso/5.0/intel-
ompi/bin/pw.x

F. SIESTA

CPU:

Location = /export/data/apps/siesta
Executable file = /export/data/apps/siesta/3.1/intel-
ompi/bin/siesta

G. ABINIT

CPU:

Location = /export/data/apps/abinit
Executable file = /export/data/apps/abinit/6.12.3/intel-
ompi/bin/abinit

H. RUMD

GPU:

Location=
Executable file=

I. HOOMD

CPU:

Location = /usr/bin
Executable file = /usr/bin/hoomd

GPU:

Location = /usr/bin
Executable file = /usr/bin/hoomd

J. VASP

3.7. Scheduler & Distributed Resource Manager (SGE)

Oracle Grid Engine, previously known as Sun Grid Engine (SGE) is an open source batch-queuing system, developed and supported by Sun Microsystems. Sun once also sold a commercial product based on SGE, known as N1 Grid Engine (N1GE).

SGE is typically used on a computer farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses.

Features new in version:

- Advance reservation
- Array job interdependencies
- Rule-based Resource Quota control
- Enhanced remote execution (without using external rshd/rlogind/sshd processes)
- Multi-clustering
- Daemons managed by the Service Management Facility on Solaris
- Pseudo TTY (pty) support for interactive jobs
- Job Submission Verifier (client-side and server-side job verification)
- GUI Installer and SGE Inspect
- Topology-aware scheduling and thread binding

- Hadoop integration, Amazon EC2 integration for cloud computing

Other features of SGE include:

- Multiple advanced scheduling algorithms allow powerful policy-based resource allocation Cluster queues
- Job and scheduler fault tolerance - Grid Engine continues to operate as long as there is one or more hosts available
- Job checkpointing
- Job arrays and job tasks
- DRMAA (Job API)
- Resource reservation
- XML status reporting (qstat and qhost), and the xml-qstat web interface
- Parallel jobs (MPI, PVM, OpenMP), and scalable parallel job startup with qsh[10]
- Usage accounting
- Accounting and Reporting COnsole (ARCO)
- parallel make: distmake, dmake (Sun Studio), and SGE's own qmake
- FLEXlm integration and multi-cluster software license management with LicenseJuggler

Use the qsub Command to Submit a Job:

The `qsub` command is used to submit jobs to SGE. The syntax of the `qsub` command is:

```
$ qsub [-cwd] [-v SOME_VAR] [-o path] [-e path] \
      [-M mail_address][-m mail_options] [-l resources] script
```

Where:

`-cwd`

Directs SGE to run the job in the same directory from which we submitted it. Alternatively, we can specify this flag in the SGE command file for the job.

`-v SOME_VAR`

Passes environment variable `SOME_VAR` to the job. Alternatively, we can specify this flag in the SGE command file for the job.

`-o path`

Redirects stdout from the SGE script. The default is home directory. Specify `/dev/null` to discard SGE messages. Alternatively, we can specify this flag in the SGE command file for the job.

-e path

Redirects stderr from the SGE script. The default is home directory. Specify `/dev/null` to disregard SGE error messages. Alternatively, we can specify this flag in the SGE command file for the job.

-M mail_address

where *mail_address* is user's email address. It is always `login_id@mail on Hoffman2`.

-m mail_options

Specifies the circumstances under which mail is to be sent to the job owner defined by `-M` option. For example options "bea" mean mail is sent at the beginning, end, and at abort time (if it happens) of the job. Option "n" means no mail will be sent.

-l resources

Specifies a list of resources required for the job, for example memory and time per core:

```
-l h_data=1024M,h_rt=24:00:00
```

script

Either the SGE command file or the script that starts up the job.

The `qsub` command line switches and options can also be used as active comments or embedded directives in an SGE command file that we submit with the `qsub` command. Advantages of this approach are: we have a record of what options were used to run our job; we can easily resubmit jobs; and we can use one command file as the basis for creating other similar command files. For example, if the file `myjob.cmd` contains:

```
#!/bin/csh
/path/to/executable
```

and the `qsub` command used to submit it is:

```
$ qsub -cwd -o path -M login_id@mail -m bea \
      -l h_data=1024M,h_rt=24:00:00 myjob.cmd
```

then the same result could be achieved by adding the following lines to the `myjob.cmd` file before the `/path/to/executable` line:

```
## -cwd
## -o path
## -M login_id@mail
```

```
## $ -m bea
## $ -l h_data=1024M,h_rt=24:00:00
```

and submitting the myjob.cmd script with:

```
$ qsub myjob.cmd
```

After submitting a job with qsub, SGE will respond with something like:

```
Your job 624556 ("myjob.cmd") has been submitted
```

where 624556 is the job number assigned by SGE to the job.

Use the qstat Command to Determine the Status of a Job:

The **qstat** command displays information about the jobs in the SGE queues, both running and waiting to run. The syntax of the qstat command is:

```
$ qstat [-f] [-j job_number] [-U login_id] [-u login_id]
```

where:

(qstat alone with no arguments)

Displays a list of all running and waiting jobs.

-f

Displays summary information on each queue as well as the job list.

-j *job_number*

Displays the status of the job whose job number is *job_number*

-U *login_id*

Displays a list of running and waiting jobs for those queues which *login_id* can access. Or use the **groupjobs** script for this information; enter *groupjobs -help* for usage information.

-u *login_id*

Displays a list of *login_id*'s running and waiting jobs. Or use the **myjobs** script for this information for our own *login_id*.

Use the qhost Command to Display Node Information:

The **qhost** command displays information about compute nodes: their architectures, number of processors, load, etc. The syntax of the qhost command is:

```
$ qhost [-j] [-q]
```

where:

(qhost alone with no arguments)

Displays a table of information about the compute nodes.

-j

Adds information about the specific jobs that are running on each compute node.

-q

Shows the queues each compute node accepts.

Use the qdel Command to Cancel a Job:

The **qdel** command is used to cancel a job either while it is waiting to execute or while it is running. The syntax of the qdel command is:

```
$ qdel job_number
```

If a running job does not get cancelled right away, enter:

```
$ qdel -f job_number
```

to force it to be cancelled. Jobs in the "dr" state (disabled running) cannot be cancelled by the job owner. They must be cancelled by a system administrator. "dr" state jobs usually indicate a system hardware problem.

Queue list

- A. **gpu.q** -- This queue is for gpu/cuda jobs, with no time limit.
- B. **long.q** -- This queue is for long parallel jobs, with 15 days time limit.
- C. **short.q** -- This queue is for short parallel jobs, with 5 days time limit.
- D. **serial.q** -- This queue is for serial jobs (single core), with no time limit.

a. ABINIT

Version=6.12.3

PATH=/export/data/apps/abinit/6.12.3

Job Script

The following job script is adequate for running of ABINIT application

```
$vi abinit_cpu_sge.sh
#!/bin/bash
```



```
#$ -N ABINIT
#$ -cwd
#$ -S /bin/bash
#$ -q short.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 16

/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/abinit/6.12.3/intel-mpi/bin/abinit
```

Job Submission

```
$qsub abinit_cpu_sge.sh
```

b. CPMD

Version=3.15.3

PATH=/export/data/apps/cpmd/3.15.3

Job Script

```
#!/bin/bash
#$ -N CPMD
#$ -cwd
#$ -S /bin/bash
#$ -q short.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 16

/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/cpmd/3.15.3/intel-mpi/cpmd.x
```

Job Submission

```
$qsub cpmd_cpu_sge.sh
```

c. GROMACS

CPU

Version=4.5.5

PATH=/export/data/apps/gromacs/4.5.5

Job Scripts

```
#!/bin/bash
#$ -N Gromacs
#$ -cwd
#$ -S /bin/bash
```

```

#$ -q serial.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID

/export/data/apps/gromacs/4.5.5/intel/bin/grompp_mpi
/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/gromacs/4.5.5/intel/bin/mdrun_mpi

```

Job Submission

```
$qsub gromacs_cpu_sge.sh
```

GPU**Version=****PATH=/export/data/apps/gromacs-gpu****Job scripts**

```

#!/bin/bash
#$ -N GMX-GPU
#$ -cwd
#$ -S /bin/bash
#$ -q gpu.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -l gpu=1

# /export/data/apps/gromacs/4.5.5/intel-ompi/bin/grompp_mpi
#$ -v LD_LIBRARY_PATH=/export/data/apps/gromacs-
gpu:/export/data/apps/gromacs-gpu/cuda23/lib64
#$ -v OPENMM_PLUGIN_DIR=/export/data/apps/gromacs-gpu
#$ -v OPENMM_PLATFORM=Cuda,CudaDevice=1
export LD_LIBRARY_PATH=/export/data/apps/gromacs-
gpu:/export/data/apps/gromacs-gpu/cuda23/lib64
export OPENMM_PLUGIN_DIR=/export/data/apps/gromacs-gpu
export OPENMM_PLATFORM=Cuda,CudaDevice=0
/export/data/apps/gromacs-gpu/mdrun-openmm

```

Job Submission

```
$qsub gromacs_gpu_sge.sh
```

d. HOOMD**CPU****Version=0.11.0****PATH=****Job script**

```

#!/bin/bash
#$ -N hoomd-cpu

```

```
#$ -cwd
#$ -S /bin/bash
#$ -q serial.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -v OMP_NUM_THREADS=1

export OMP_NUM_THREADS=1

/usr/bin/hoomd lj_liquid_bmark.hoomd
```

Job submission

```
$qsub hoomd_cpu_serial_sge.sh
```

GPU

Version=0.11.0

PATH=

Job script

```
#!/bin/bash
#$ -N hoomdgpu
#$ -cwd
#$ -S /bin/bash
#$ -q gpu.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 1
#$ -l gpu=1
#$ -v OMP_NUM_THREADS=1

export OMP_NUM_THREADS=1

/usr/bin/hoomd lj_liquid_bmark.hoomd
```

Job submission

```
$qsub hoomd_gpu_sge.sh
```

e. LAMMPS

CPU

Version=

PATH= /export/data/apps/lammps-cpu

Job scripts

```
#!/bin/bash
#$ -N LAMMPS-CPU
#$ -cwd
```

```

#$ -S /bin/bash
#$ -q short.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 16

/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/lammps-cpu/30Aug12/bin/lmp_openmpi

```

Job Submission

```
$qsub lammps_cpu_sge.sh
```

GPU**Version=****PATH= /export/data/apps/lammps-cuda****Job scripts**

```

#!/bin/bash
#$ -N Lammps-cuda
#$ -cwd
#$ -S /bin/bash
#$ -q gpu.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -l gpu=1

#$ -v LD_LIBRARY_PATH=/export/data/apps/gromacs-
gpu:/export/data/apps/gromacs-gpu/cuda23/lib64
export LD_LIBRARY_PATH=/export/data/apps/gromacs-
gpu:/export/data/apps/gromacs-gpu/cuda23/lib64
/export/data/mpi/openmpi/1.6.1/intel/bin/mpirun -np 1
/export/data/apps/lammps-cuda/bin/lmp_cuda -sf cuda -v g 1 -v
x 16 -v y 16 -v z 16 -v t 100 < in.lj.cuda

```

Job Submission

```
$qsub lammps_gpu_sge.sh
```

f. NAMD**CPU****Version=2.9****PATH= /export/data/apps/NAMD_2.9_Linux-x86_64-ibverbs****Job script**

```

#!/bin/bash
#$ -N Namd

```

```

#$ -cwd
#$ -S /bin/bash
#$ -q short.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe mpi 32

nodefile=$TMPDIR/namd2.nodelist
echo group main > $nodefile
awk '{ for (i=0;i<$2;++i) {print "host",$1} }' $PE_HOSTFILE >>
$nodefile

cat $nodefile
dir=/export/data/apps/NAMD_2.9_Linux-x86_64-ibverbs
$dir/charmrun ++remote-shell ssh ++nodelist $nodefile +p$NSLOTS
$dir/namd2 apoal.namd

```

Job Submission

```
$qsub namd_cpu_sge.sh
```

GPU**Version=2.9****PATH=/export/data/apps/NAMD_2.9_Linux-x86_64-multicore-CUDA****Job scripts**

```

#!/bin/bash
#$ -N Namd-CUDA
#$ -cwd
#$ -S /bin/bash
#$ -q gpu.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -l gpu=1
#$ -v LD_LIBRARY_PATH=/export/data/apps/NAMD_2.9_Linux-x86_64-
multicore-CUDA:$LD_LIBRARY_PATH

```

```

export LD_LIBRARY_PATH=/export/data/apps/NAMD_2.9_Linux-
x86_64-multicore-CUDA:$LD_LIBRARY_PATH
/export/data/apps/NAMD_2.9_Linux-x86_64-multicore-CUDA/namd2
+idlepoll apoal.namd

```

Job Submission

```
$qsub namd_gpu_sge.sh
```

g. QUANTUM ESPRESSO**Version=5.0**

PATH= /export/data/apps/q-espresso/5.0

Job scripts

```
#!/bin/bash
#$ -N QEspresso
#$ -cwd
#$ -S /bin/bash
#$ -q long.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 16
```

```
/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/q-espresso/5.0/intel-ompi/bin/pw.x -in input
file > file.out
```

Job Submission

```
$qsub qespresso_cpu_sge.sh
```

h. SIESTA

Version=3.1

PATH= /export/data/apps/siesta/3.1

Job script

```
#!/bin/bash
#$ -N SIESTA
#$ -cwd
#$ -S /bin/bash
#$ -q short.q
#$ -e err.$JOB_ID
#$ -o out.$JOB_ID
#$ -pe orte 16
```

```
/export/data/mpi/openmpi/1.4.5/intel/bin/mpirun -np $NSLOTS
/export/data/apps/siesta/3.1/intel-ompi/bin/siesta
```

Job submission

```
$qsub siesta_cpu_sge.sh
```

3.8. Troubleshooting SGE

(a) Submitting Serial Job

Batch or serial jobs can be submitted to SGE in two ways.

Using SGE job submit scripts

This example executes sleep command via job submit script method.

Create a script as follow:

```
[root@kohinoor ~]# vi submit.sh

#!/bin/bash
#$ -N Serial-Job
#$ -cwd
#$ -S /bin/bash
#$ -e err.$JOB_ID.$JOB_NAME
#$ -o out.$JOB_ID.$JOB_NAME

sleep 100
```

Entries which start with `#$` will be treated as SGE options.

- `#$ -cwd` means to execute the job for the current working directory.
- `#$ -S /bin/bash` specifies the interpreting shell for this job to be the Bash shell.
- `#$ -N <job name>` : The name to given to this job
- `#$ -e <file name>` : Mention the name of the file to store the errors
- `#$ -o <file name>` : Mention the name of the file to store the output
- `#$ -q <queue name>` : Mention the name of the queue

To submit this serial job script, you should use the `qsub` command.

```
[root@kohinoor~]# qsub submit.sh
your job 29 ("submit.sh") has been submitted
```

(b) Submitting Parallel Job

Using job submit script:

```
# cd ~
# vi test/mpi-ring.qsub
[root@ ~]# vi submit.sh
```

```
#!/bin/bash
#$ -N Parallel-Job
#$ -cwd
#$ -S /bin/bash
#$ -e err.$JOB_ID.$JOB_NAME
#$ -o out.$JOB_ID.$JOB_NAME
#$ -q all.q
#$ -pe orte 8
/opt/mpi/openmpi/1.3.3/intel/bin/mpirun -np $NSLOTS ./a.out > outputfile
```

To submit the job:
\$ qsub submit.sh

(c) Knowing the status of job

```
[root@Kohinoor~]# qstat
```

```
[root@kohinoor ~]# qstat -j N    N is the job id.
```

(d) Killing a Job

```
# qdel <jobid>
```

jobid is the pid of the job submitted which can be found when you use the qstat command.

Qhost

qhost shows the current status of the available Grid Engine hosts, queues and the jobs associated with the queues. Selection options allow you to get information about specific hosts, queues, jobs or users. Without any option qhost will display a list of all hosts without queue or job information.

```
[root@kohinoor ~]# qhost
```

Note: The scripts for running different applications under Sun Grid Engine, are kept in /JOB_SCRIPTS of the master node. Copy the scripts to your working directory. Edit the

script with number of slots (for parallel jobs) and input file parameters and number of slots to use.

3.9. Monitoring Tool - Ganglia

Ganglia is a scalable distributed system monitor tool 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.

There are three services are mandatory for Ganglia. At server, httpd, gmetd and gmond, and at client, gmond only. If user wants to restart these services, he has to login as root, and restart three services at master node and one at every compute nodes:

```
# service httpd restart
# service gmetd restart
# service gmond restart
```

To open the web interface of Ganglia, type the following URL in browser:
<http://172.16.10.20/ganglia>

In Figure 3.13, there is a screenshot which give a brief picture of ganglia output:





Figure 3.13: Ganglia Monitoring Tool

3.10. System Management:

The integrated management module (IMM) consolidates the service processor functionality, Super I/O, video controller, and remote presence capabilities in a single chip on the server system board.

IMM features:

The IMM provides the following functions:

- Around-the-clock remote access and management of our server
- Remote management independent of the status of the managed server
- Remote control of hardware and operating systems
- Web-based management with standard Web browsers

IMM has the following features:

- Access to critical server settings
- Access to server vital product data (VPD)
- Advanced Predictive Failure Analysis (PFA) support
- Automatic notification and alerts
- Continuous health monitoring and control
- Choice of a dedicated or shared Ethernet connection
- Domain Name System (DNS) server support
- Dynamic Host Configuration Protocol (DHCP) support
- E-mail alerts
- Embedded Dynamic System Analysis (DSA)
- Enhanced user authority levels
- LAN over USB for in-band communications to the IMM
- Event logs that are time stamped, saved on the IMM, and can be attached to e-mail alerts
- Industry-standard interfaces and protocols
- OS watchdogs
- Remote configuration through Advanced Settings Utility (ASU)
- Remote firmware updating
- Remote power control
- Seamless remote accelerated graphics
- SecureWeb server user interface
- Serial over LAN
- Server console redirection
- Simple Network Management Protocol (SNMP) support
- User authentication using a secure connection to a Lightweight Directory Access
- Protocol (LDAP) server
- Remote presence, including the remote control of a server

- Operating-system failure screen capture and display through the Web interface
- Remote disk, which enables the attachment of a diskette drive, CD/DVD drive, USB flash drive, or disk image to a server

The firmware included with this server is an implementation of Unified Extensible Firmware Interface (UEFI). It offers several features, such as UEFI 2.1 compliance, iSCSI compatibility, and enhanced reliability and service capabilities. The Setup utility provides server information, server setup, customization compatibility, and establishes the boot device order.

Web browser and operating-system requirements:

The IMM Web interface requires the Java Plug-in 1.5 or later (for the remote presence feature) and one of the following Web browsers:

- Microsoft Internet Explorer version 6.0 or later with the latest Service Pack
- Mozilla Firefox version 1.5 or later

Accessing the IMM Web interface:

The IMM supports both static and Dynamic Host Configuration Protocol (DHCP) IP addressing. The default static IP address assigned to the IMM is 192.168.70.125. The IMM is initially configured to attempt to obtain an address from a DHCP server, and if it cannot, it uses the static IP address.

The IMM provides the choice of using a dedicated systems-management network connection or one that is shared with the server. The default connection for rack-mounted and tower servers is to use the dedicated systems-management network connector.

Logging in to the IMM:

Important: The IMM is set initially with a user name of USERID and password of PASSWORD (with a zero, not the letter O). This default user setting has Supervisor access.

Change this default password during our initial configuration for enhanced security.

To access the IMM through the IMM Web interface, complete the following steps:

1. Open a Web browser. In the address or URL field, type the IP address or host name of the IMM server to which we want to connect.
2. Type our user name and password in the IMM Login window. If we are using the IMM for the first time, we can obtain our user name and password from our system administrator. All login attempts are documented in the event log. Depending on how our system administrator configured the user ID, we might need to enter a new password.

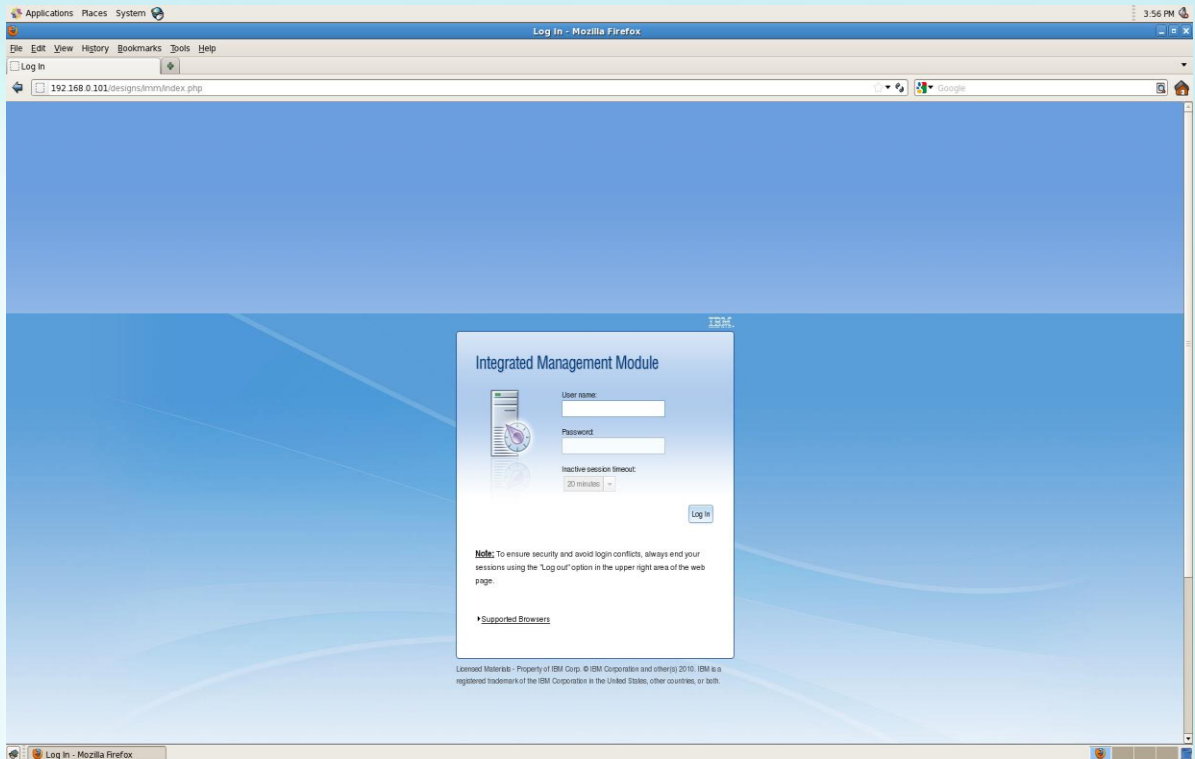


Figure 3.10: IMM Login Screen

3. On the Welcome Web page, select a timeout value from the drop-down list in the field that is provided. If our browser is inactive for that number of minutes, the IMM logs off the Web interface.

Note: Depending on how our system administrator configured the global login settings, the timeout value might be a fixed value.

4. Click Continue to start the session. The browser opens the System Status page, which gives us a quick view of the server status and the server health summary.

LOCUZ

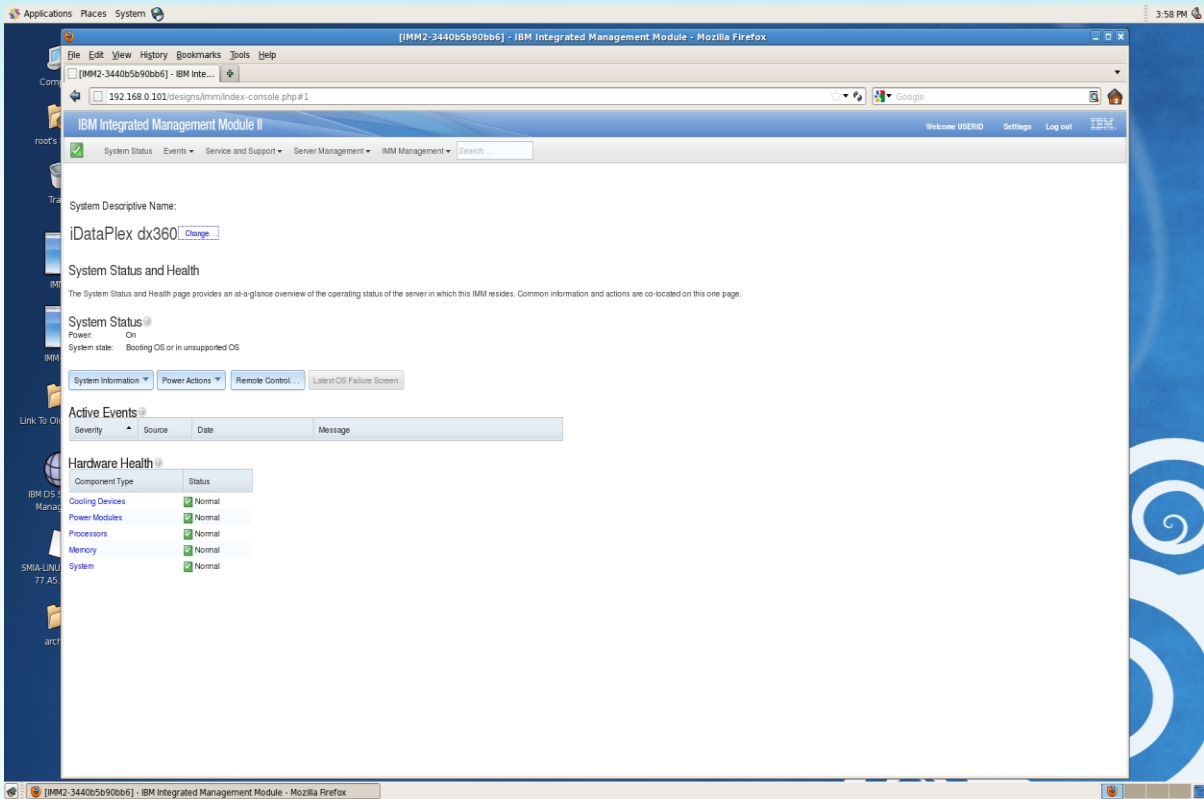


Figure 3.11: System Status Page of IMM

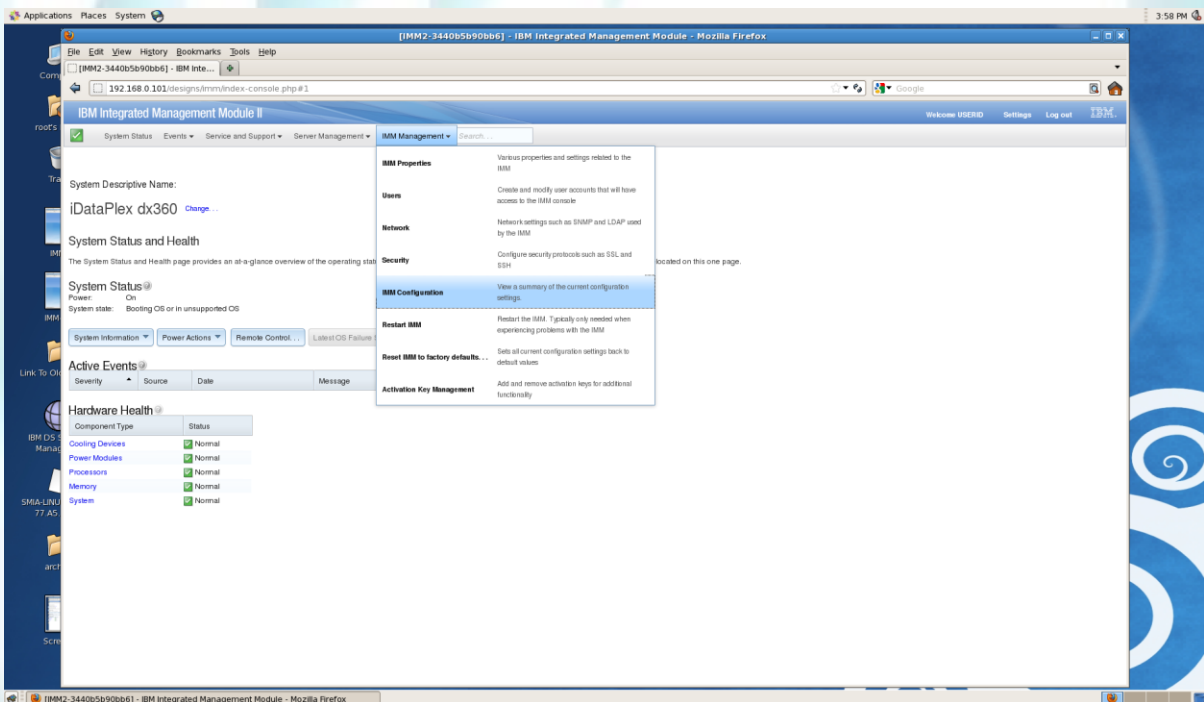


Figure 3.12: IMM Management.

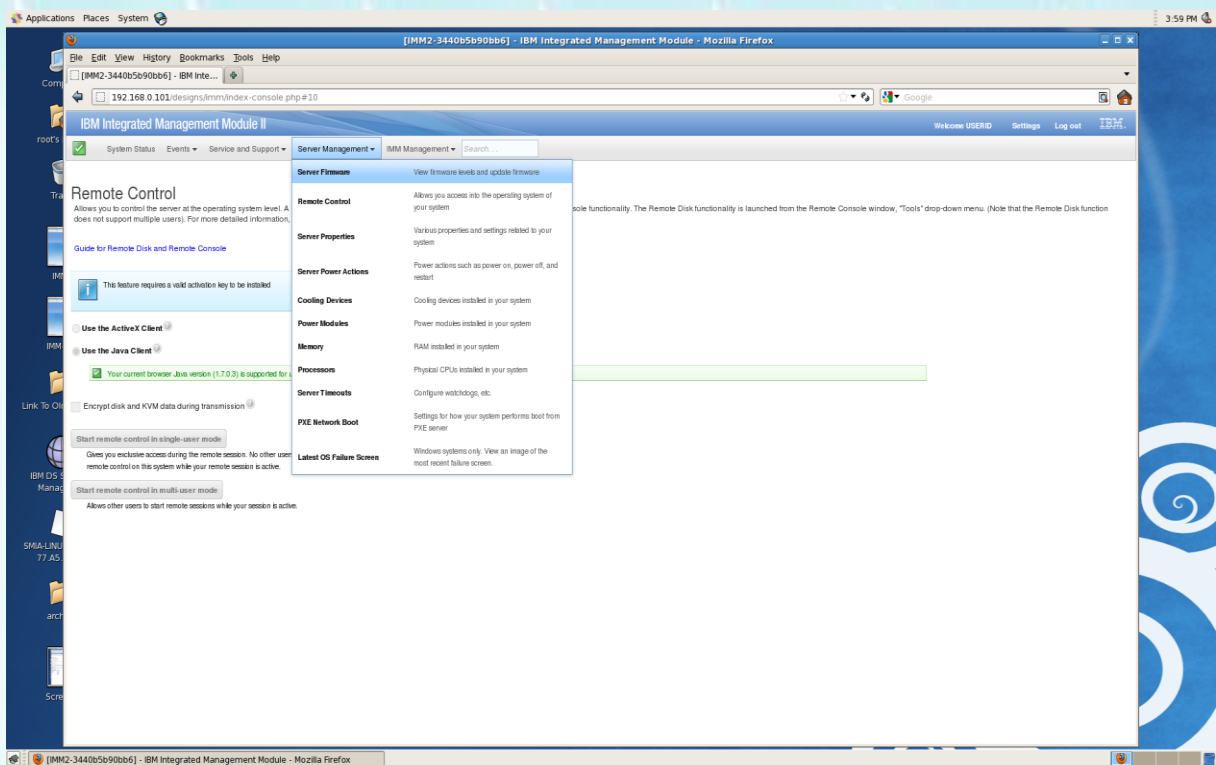


Figure 3.14: IMM Server management.

5. Startup and Shutdown Procedures

Startup Sequence

Step1. Power on the IBM System Storage DS3510. After few minutes check all the led showing the green light.

Step2. Power on the IBM x3650 M4 server using the power button located at the front of the server. Check with the console that it has booted normally.

Step3. Check whether the storage has been mounted on the master server using df -h command.

```
root@Kohinoor ~] df -h
```

Step4. Power on all the compute nodes as well has GPU nodes using the power button in the front.

Step5. Check all the file systems are mounted properly. If not run the command Rocks run host “hostname; mount -a”

```
root@Kohinoor ~] rocks run host “hostname; mount -a”
```

LOCUZ

ShutDown Sequence

Step1. Unmount the storage mounted on all the server using the command

Compute nodes

```
root@Kohinoor ~] rocks run host "hostname; umount /scratch"
```

```
root@Kohinoor ~] rocks run host "hostname; umount /scratch2"
```

```
root@Kohinoor ~] rocks run host "hostname; umount /export/home"
```

Master node

```
root@Kohinoor ~] Umount /scratch
```

```
root@Kohinoor ~] Umount /scratch
```

```
root@Kohinoor ~] Umount /export/home
```

Step2. Shut down the compute and gpu nodes

```
root@Kohinoor ~] Rocks run host "hostname; init 0"
```

Step3. Shut down the master node

```
root@kohinoor ~] Init 0
```

LOCUZ

6. scallation Matrix

Service Desk Timings: 09:15 Hrs to 18:00 (Monday to Friday)	
HPC Support	
Response Time	4 Hrs.
Level 1	Mr. Nitesh
	Helpdesk
	040-30509117
	support@locuz.com
Level 2	
	Mr. Narender gaddam
	Mr. Sangamesh
	Sr. System Engineer
	Sr.Consultant-HPC
	040-30509230
	080-40506868-6820
	91-
	91-9986746471
	Narender.gaddam@locuz.com
	Sangamesh.banappa@locuz.com
Level 3	
	Mr. Srinivas
	Service Delivery Manager
	040-30509117
	91-9885035972
	Srinivas.i@locuz.com
Level 4	
	Srinivas Chada
	GM-Availability Services & Operations
	040-30509277
	91-9949545672
	sreenivas.chada@locuz.com

LOCUZ

