Kohinoor3 Cluster User/Admin Manual

Intel Xeon 1380 cores HPC Cluster
At
Tata Institute of Fundamental Reasearch (TIFR)
Hydrabad

By

Netweb Technologies India Pvt. Ltd.
Plot No-H1, Pocket- 9,
Faridabad Industrial Town (FIT)
Sector- 57, Faridabad, Ballabgarh,
State – Haryana- 121004, India

Contents

Chapter 1

About Document

1.1 Document History

Version	Release Date	Prepared by
1.0	28 Aug 2016	Netweb Technologies India Pvt Ltd.

1.2 Netweb Technologies India Pvt Ltd 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 homogeneous 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.

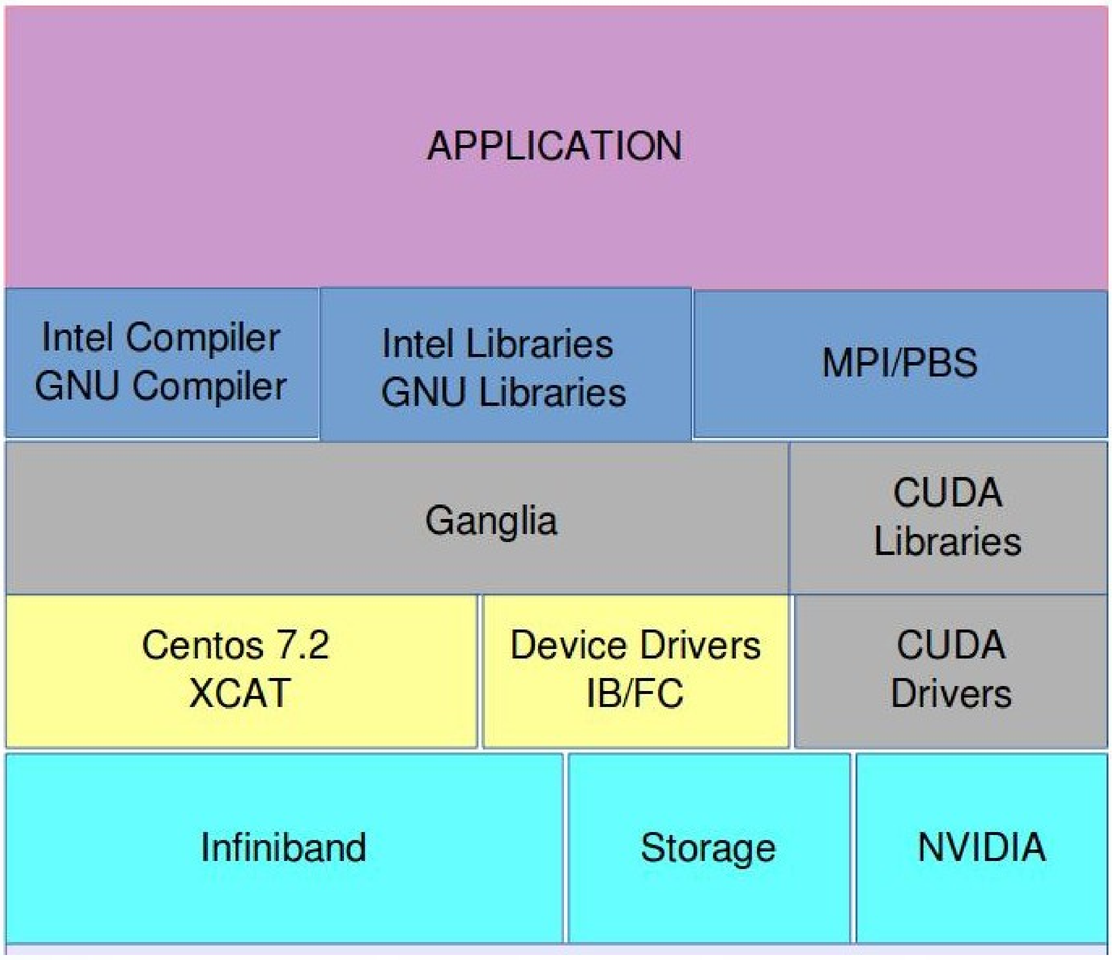
The term is most commonly associated with computing used for scientific research or computational science. A related term, high-performance technical computing (HPTC), generally refers to the engineering applications of cluster-based computing (such as computational fluid dynamics and the building and 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. Below figure shows a brief view about components and its dependency.



HPC Stack

Netweb TECHNOLOGIES

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, connectivitychannels/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 Kohinoor3 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 Kohinoor3 server hardware, XCAT cluster toolkit has been installed with CentOS 7.2 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 7.2 on Kohinoor3, 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(Slurm).

For Kohinoor3 resources management, Simple Linux Utility for Resource Management is a open-source HPC job scheduler. The Simple Linux Utility for Resource Management (Slurm) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. Slurm requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, Slurm has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work.

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 2.0 (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.

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:

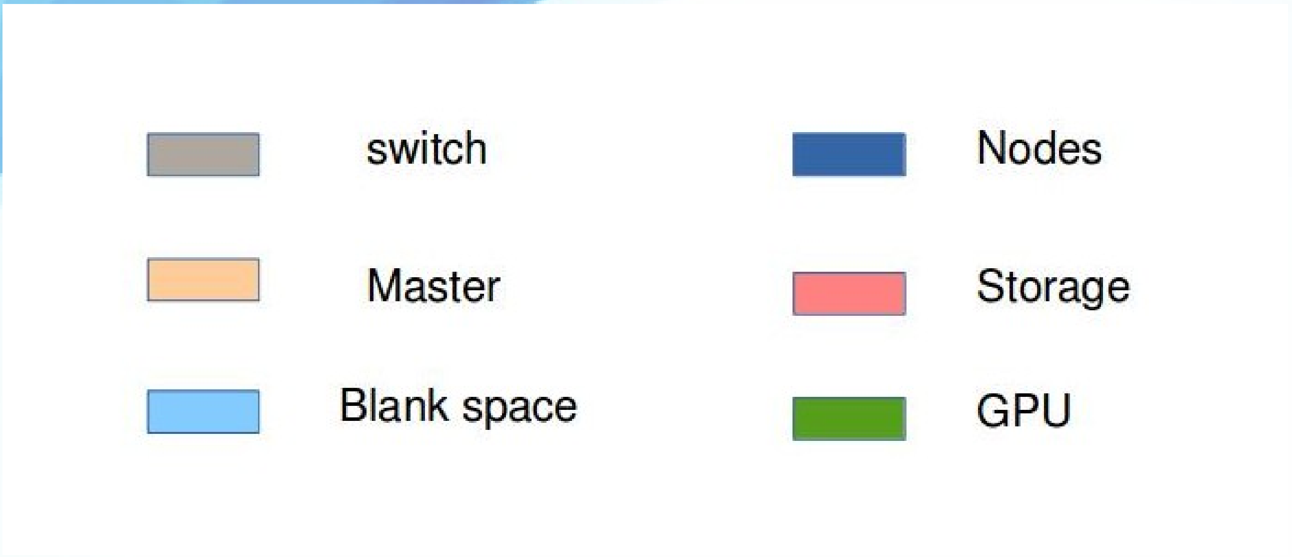
Application program can be any program which can either run on individually on different nodes or on across the node in a parallelized manner. HPC applications are programs which may be designed by end-user or purchased from third party software vendors. Applications may be serial or parallel but this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

Chapter 3

KOHINOOR3 Implementation

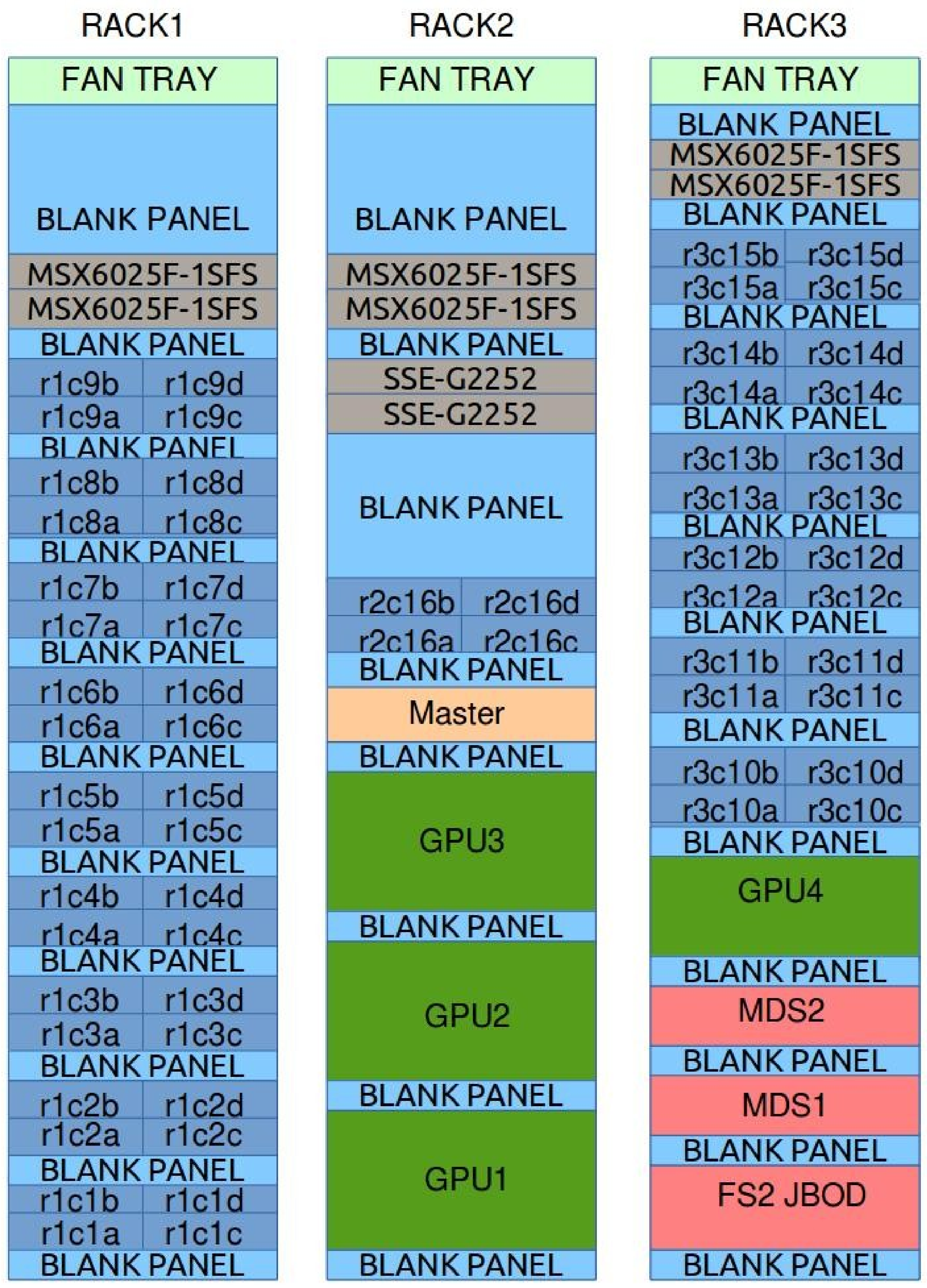
1. IB Switch
2. Ethernet Switch
3. Compute nodes
4. Monitor
5. GPU nodes
6. Master Node
7. Storage

Here we have impelemented KOHINOOR3 diagram with different colour coding which will help you to understand actuall representation of the cluster.



▨	switch	🟦	Nodes
🟧	Master	🟥	Storage
🟦	Blank space	🟩	GPU

Netweb
TECHNOLOGIES



KOHINOOR3 Rack Setup View

3.1. Servers Hardware Details

In KOHINOOR3 cluster, there is a single master node, 32 compute nodes. The following are the hardware details of Servers:

Master Node:

Hardware Details – Supermicro	
Processor	2 x E5-2630 V4
RAM	64 GB
HDD	4x600 GB SAS
Ethernet	2 port
Management Port	1 port
Infiniband	1 port
Optical Drive	DVD Drive
OS Details	
OS	CentOS 7.2
Kernel	3.10.0-327.el7.x86_64
Partition Details	
/	100 GB
Swap	32 GB
/home	200 TB
/apps	300 GB
/boot	1 GB

Compute Node:

Hardware Details – Supermicro	
Processor	2 x E5-2630 V4
RAM	64 GB
HDD	1 TB SATA
Ethernet	2 port
Management Port	1 port
Infiniband	1 port
OS Details	
OS	CentOS 7.2
Kernel	3.10.0-327.el7.x86_64
Partition Details	
/	100 GB
Swap	32 GB
/boot	1 GB

GPU Compute Node:

Hardware Details – Supermicro	
Processor	E5-2630 V4
RAM	64 GB
HDD	1 TB
Ethernet	2 port
Management Port	1 port
GPU	Tesla K40 C
Infiniband	1 port
OS Details	
OS	CentOS 7.2
Kernel	3.10.0-327.el7.x86_64

Partition Details	
/	100 GB
Swap	32 GB
/boot	1 GB

Storage Nodes:

Hardware Details – Supermicro	
Processor	2 x E5-1650 V3
RAM	128 GB
HDD	2 x 80GB SSD
Ethernet	2 port
Management Port	1 port
Infiniband	1 port
Optical Drive	DVD Drive
OS Details	
OS	CentOS 7.2
Kernel	3.10.0-327.el7.x86_64
Partition Details	
/	62 GB
Swap	64 GB
/boot	1 GB

3.2. Switches and Connectivity:

In Kohinoor3 implementation, two types of switches has been used – Ethernet, Infiniband switches.

Ethernet Switch:

In Kohinoor3 implementation, 48 Port Gigabit supermicro Switch has been used to configure the private network and in figure 3.2, Ethernet connectivity has been explained:

SSE-G2252	48 Port Gigabit Switch
MSX6025F-1SFS	36 port Infiniband Switch

Infiniband Switch:

In Kohinoor,3 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 below figure Kohinoor3 Schematic Diagram.

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

SNo	Hostname	Ethernet IP	IB IP	Management IP	Internal Conn.
1	Kohinoor3	192.168.102.254	192.168.12.254	11.11.11.254	172.16.10.24
2	r1c1a	192.168.102.1	192.168.12.1	11.11.11.1	
3	r1c1b	192.168.102.2	192.168.12.2	11.11.11.2	
4	r1c1c	192.168.102.3	192.168.12.3	11.11.11.3	
5	r1c1d	192.168.102.4	192.168.12.4	11.11.11.4	
6	r1c2a	192.168.102.5	192.168.12.5	11.11.11.5	
7	r1c2b	192.168.102.6	192.168.12.6	11.11.11.6	
8	r1c2c	192.168.102.7	192.168.12.7	11.11.11.7	
9	r1c2d	192.168.102.8	192.168.12.8	11.11.11.8	
10	r1c3a	192.168.102.9	192.168.12.9	11.11.11.9	

11	r1c3b	192.168.102.10	192.168.12.10	11.11.11.10	
12	r1c3c	192.168.102.11	192.168.12.11	11.11.11.11	
13	r1c3d	192.168.102.12	192.168.12.12	11.11.11.12	
14	r1c4a	192.168.102.13	192.168.12.13	11.11.11.13	
15	r1c4b	192.168.102.14	192.168.12.14	11.11.11.14	
16	r1c4c	192.168.102.15	192.168.12.15	11.11.11.15	
17	r1c4d	192.168.102.16	192.168.12.16	11.11.11.16	
18	r1c5a	192.168.102.17	192.168.12.17	11.11.11.17	
19	r1c5b	192.168.102.18	192.168.12.18	11.11.11.18	
20	r1c5c	192.168.102.19	192.168.12.19	11.11.11.19	
21	r1c5d	192.168.102.20	192.168.12.20	11.11.11.20	
22	r1c6a	192.168.102.21	192.168.12.21	11.11.11.21	
23	r1c6b	192.168.102.22	192.168.12.22	11.11.11.22	
24	r1c6c	192.168.102.23	192.168.12.23	11.11.11.23	
25	r1c6d	192.168.102.24	192.168.12.24	11.11.11.24	
26	r1c7a	192.168.102.25	192.168.12.25	11.11.11.25	
27	r1c7b	192.168.102.26	192.168.12.26	11.11.11.26	
28	r1c7c	192.168.102.27	192.168.12.27	11.11.11.27	
29	r1c7d	192.168.102.28	192.168.12.28	11.11.11.28	
30	r1c8a	192.168.102.29	192.168.12.29	11.11.11.29	
31	r1c8b	192.168.102.30	192.168.12.30	11.11.11.30	
32	r1c8c	192.168.102.31	192.168.12.31	11.11.11.31	
33	r1c8d	192.168.102.32	192.168.12.32	11.11.11.32	
34	r1c9a	192.168.102.33	192.168.12.33	11.11.11.33	
35	r1c9b	192.168.102.34	192.168.12.34	11.11.11.34	
36	r1c9c	192.168.102.35	192.168.12.35	11.11.11.35	
37	r1c9d	192.168.102.36	192.168.12.36	11.11.11.36	
38	r2c16a	192.168.102.37	192.168.12.37	11.11.11.37	
39	r2c16b	192.168.102.38	192.168.12.38	11.11.11.38	
40	r2c16c	192.168.102.39	192.168.12.39	11.11.11.39	
41	r2c16d	192.168.102.40	192.168.12.40	11.11.11.40	
42	r3c10a	192.168.102.41	192.168.12.41	11.11.11.41	
43	r3c10b	192.168.102.42	192.168.12.42	11.11.11.42	
44	r3c10c	192.168.102.43	192.168.12.43	11.11.11.43	
45	r3c10d	192.168.102.44	192.168.12.44	11.11.11.44	
46	r3c11a	192.168.102.45	192.168.12.45	11.11.11.45	
47	r3c11b	192.168.102.46	192.168.12.46	11.11.11.46	

48	r3c11c	192.168.102.47	192.168.12.47	11.11.11.47	
49	r3c11d	192.168.102.48	192.168.12.48	11.11.11.48	
50	r3c12a	192.168.102.49	192.168.12.49	11.11.11.49	
51	r3c12b	192.168.102.50	192.168.12.50	11.11.11.50	
52	r3c12c	192.168.102.51	192.168.12.51	11.11.11.51	
53	r3c12d	192.168.102.52	192.168.12.52	11.11.11.52	
54	r3c13a	192.168.102.53	192.168.12.53	11.11.11.53	
55	r3c13b	192.168.102.54	192.168.12.54	11.11.11.54	
56	r3c13c	192.168.102.55	192.168.12.55	11.11.11.55	
57	r3c13d	192.168.102.56	192.168.12.56	11.11.11.56	
58	r3c14a	192.168.102.57	192.168.12.57	11.11.11.57	
59	r3c14b	192.168.102.58	192.168.12.58	11.11.11.58	
60	r3c14c	192.168.102.59	192.168.12.59	11.11.11.59	
61	r3c14d	192.168.102.60	192.168.12.60	11.11.11.60	
62	r3c15a	192.168.102.61	192.168.12.61	11.11.11.61	
63	r3c15b	192.168.102.62	192.168.12.62	11.11.11.62	
64	r3c15c	192.168.102.63	192.168.12.63	11.11.11.63	
65	r3c15d	192.168.102.64	192.168.12.64	11.11.11.64	
66	gpu1	192.168.102.65	192.168.12.65	11.11.11.65	
67	gpu2	192.168.102.66	192.168.12.66	11.11.11.66	
68	gpu3	192.168.102.67	192.168.12.67	11.11.11.67	
69	rgpu4	192.168.102.68	192.168.12.68	11.11.11.68	
70	mds1	192.168.102.69	192.168.12.69	11.11.11.69	
71	mds2	192.168.102.70	192.168.12.70	11.11.11.70	

3.3. Storage:

Lustre is a type of parallel distributed file system, generally used for large-scale cluster computing. The name Lustre is a portmanteau word derived from Linux and cluster.Lustre file system software is available under the GNU General Public License (version 2 only) and provides high performance file systems for computer clusters ranging in size from small workgroup clusters to large-scale, multi-site clusters.

The Lustre file system architecture was started as a research project in 1999 by Peter Braam, who was on the staff of Carnegie Mellon University (CMU) at the time. Braam went on to found his own company Cluster File Systems in 2001, starting from work on the InterMezzo file system in the Coda project at CMU. Lustre was developed under the Accelerated Strategic Computing Initiative Path Forward project funded by the United States Department of Energy, which included Hewlett-Packard and Intel. In September 2007, Sun Microsystems acquired the assets of Cluster File Systems Inc. including its intellectual property. Sun included Lustre with its high-performance computing hardware offerings, with the intent to bring Lustre technologies to Sun's ZFS file system and the Solaris operating system. In November 2008, Braam left Sun Microsystems, and Eric Barton and Andreas Dilger took control of the project. In 2010 Oracle Corporation, by way of its acquisition of Sun, began to manage and release Lustre.

In December 2010, Oracle announced they would cease Lustre 2.x development and place Lustre 1.8 into maintenance-only support creating uncertainty around the future development of the file system. Following this announcement, several new organizations sprang up to provide support and development in an open community development model, including Whamcloud, Open Scalable File Systems, Inc. (OpenSFS), EUROPEAN Open File Systems (EOFS) and others. By the end of 2010, most Lustre developers had left Oracle. Braam and several associates joined the hardware-oriented Xyratex when it acquired the assets of ClusterStor, while Barton, Dilger, and others formed software startup Whamcloud, where they continued to work on Lustre.

In August 2011, OpenSFS awarded a contract for Lustre feature development to Whamcloud. This contract covered the completion of features, including improved Single Server

Metadata Performance scaling, which allows Lustre to better take advantage of many-core metadata server; online Lustre distributed filesystem checking (LFSCK), which allows verification of the distributed filesystem state between data and metadata servers while the filesystem is mounted and in use; and Distributed Namespace (DNE), formerly Clustered Metadata (CMD), which allows the Lustre metadata to be distributed across multiple servers. Development also continued on ZFS-based back-end object storage at Lawrence Livermore National Laboratory.These features were in the Lustre 2.2 through 2.4 community release roadmap. In November 2011, a separate contract was awarded to Whamcloud for the maintenance of the Lustre 2.x source code to ensure that the Lustre code would receive sufficient testing and bug fixing while new features were being developed.

In July 2012 Whamcloud was acquired by Intel, after Whamcloud won the FastForward DOE contract to extend Lustre for exascale computing systems in the 2018 timeframe. OpenSFS then transitioned contracts for Lustre development to Intel.

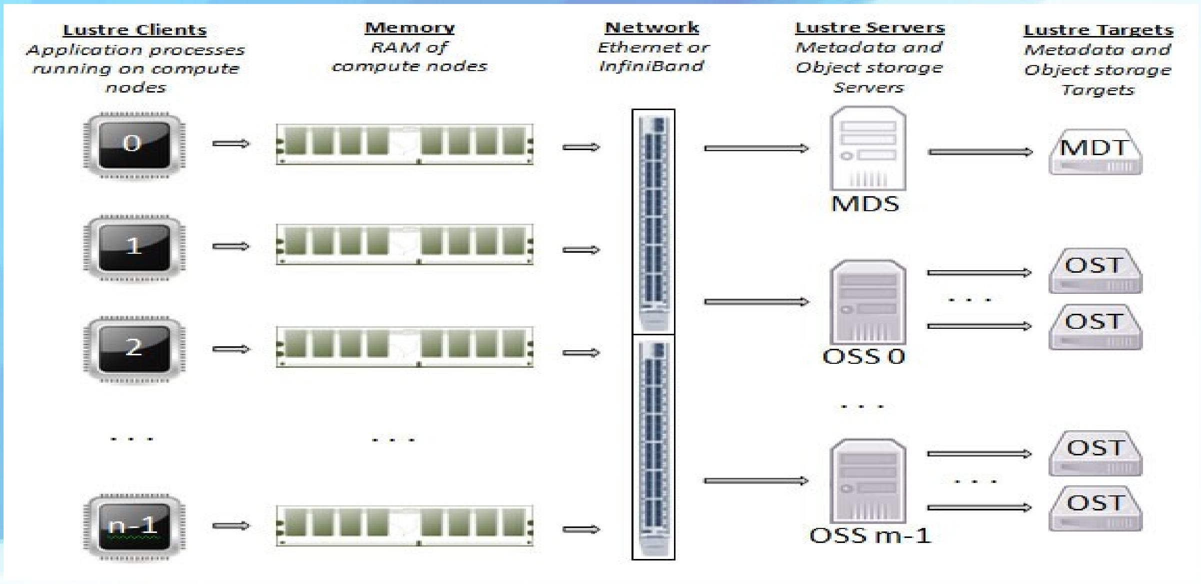
In February 2013, Xyratex Ltd., announced it acquired the original Lustre trademark, logo, website and associated intellectual property from Oracle. In June 2013, Intel began expanding Lustre usage beyond traditional HPC, such as within Hadoop. For 2013 as a whole, OpenSFS announced request for proposals (RFP) to cover Lustre feature development, parallel file system tools, addressing Lustre technical debt, and parallel file system incubators.[30] OpenSFS also established the Lustre Community Portal, a technical site that provides a collection of information and documentation in one area for reference and guidance to support the Lustre open source community. On April 8, 2014, Ken Claffey announced that Xyratex/Seagate is donating the lustre.org domain back to the user community, and was completed in March, 2015.

Lustre Components

The Lustre file system is made up of an underlying set of I/O servers called Object Storage Servers (OSSs) and disks called Object Storage Targets (OSTs). The file metadata is controlled by a Metadata Server (MDS) and stored on a Metadata Target (MDT). A single Lustre file system consists of one MDS and one MDT. The functions of each of these components are described in the following list:

- **Object Storage Servers (OSSs)** manage a small set of OSTs by controlling I/O access and handling network requests to them. OSSs contain some metadata about the files stored on their OSTs. They typically serve between 2 and 8 OSTs, up to 16 TB in size each.
- **Object Storage Targets (OSTs)** are block storage devices that store user file data. An OST may be thought of as a virtual disk, though it often consists of several physical disks, in a RAID configuration for instance. User file data is stored in one or more objects, with each object stored on a separate OST. The number of objects per file is user configurable and can be tuned to optimize performance for a given workload.
- **The Metadata Server (MDS)** is a single service node that assigns and tracks all of the storage locations associated with each file in order to direct file I/O requests to the correct set of OSTs and corresponding OSSs. Once a file is opened, the MDS is not involved with I/O to the file. This is different from many block-based clustered file systems where the MDS controls block allocation, eliminating it as a source of contention for file I/O.
- **The Metadata Target (MDT**) stores metadata (such as filenames, directories, permissions and file layout) on storage

attached to an MDS. Storing the metadata on a MDT provides an efficient division of labour between computing and storage resources. Each file on the MDT contains the layout of the associated data file, including the OST number and object identifier and points to one or more objects associated with the data file.



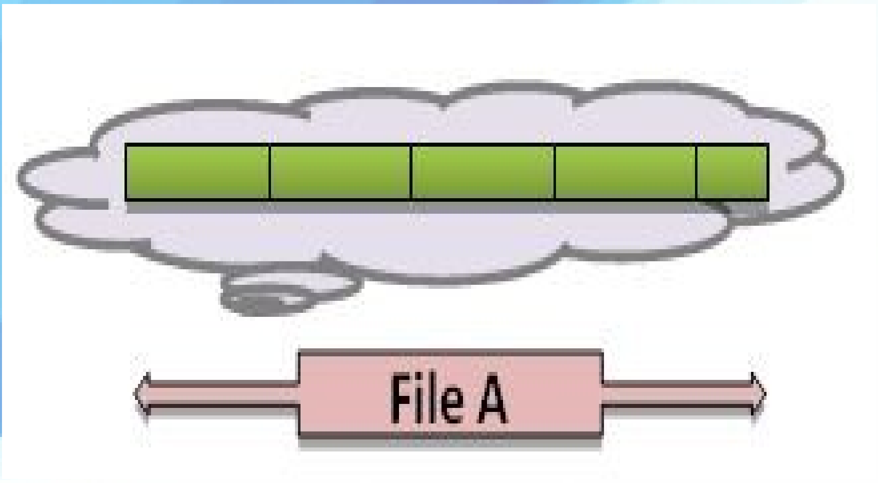
View of the Lustre File System. The route for data movement from application process memory to disk is shown by arrows.

When a compute node needs to create or access a file, it requests the associated storage locations from the MDS and the associated MDT. I/O operations then occur directly with the OSSs and OSTs associated with the file bypassing the MDS. For read operations, file data flows from the OSTs to memory. Each OST and MDT maps to a distinct subset of the RAID devices. The total storage capacity of a Lustre file system is the sum of the capacities provided by the OSTs.

File Striping Basics

A key feature of the Lustre file system is its ability to distribute the segments of a single file across multiple OSTs using a technique called **file striping**. A file is said to be **striped** when its linear sequence of bytes is separated into small chunks, or stripes, so that read and write operations can access multiple OSTs concurrently.

A file is a linear sequence of bytes lined up one after another. Below Figure shows a logical view of a single file, File A, broken into five segments and lined up in sequence.



Logical view of a file.

A physical view of File A striped across four OSTs in five distinct pieces is shown in Figure 3.3.3.

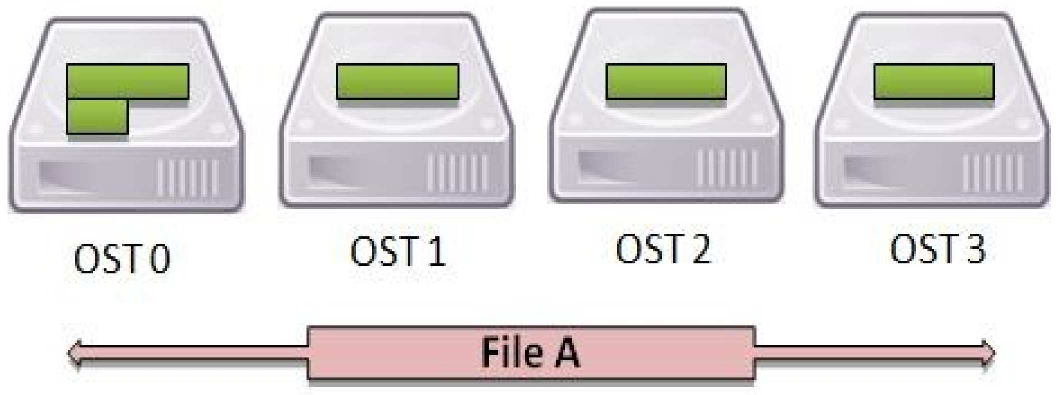


Figure 3.3.3: Physical view of a file.

Storing a single file across multiple OSTs (referred to as striping) offers two benefits: 1) an increase in the bandwidth available when accessing the file and 2) an increase in the available disk space for storing the file. However, striping is not without disadvantages, namely: 1) increased overhead due to network operations and server contention and 2) increased risk of file damage due to hardware malfunction. Given the tradeoffs involved, the Lustre file system allows users to specify the striping policy for each file or directory of files using the lfs utility. The lfs utility usage can be found in the Basic Lustre User Commands section.

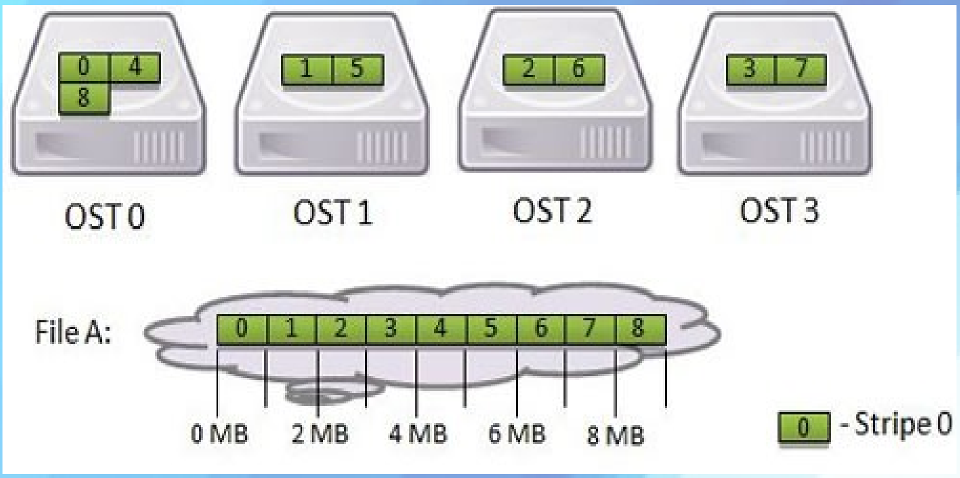
Stripe Alignment

Performance concerns related to file striping include resource contention on the block device (OST) and request contention on the OSS associated with the OST. This contention is minimized when processes (who access the file in parallel) access file locations that reside on different stripes.

Additionally, performance can be improved by minimizing the number of OSTs in which a process must communicate. An effective strategy to accomplish this is to stripe align your I/O requests. Ensure that processes access the file at offsets which correspond to stripe boundaries. Stripe settings should take into account the I/O pattern utilized to access the file.

Aligned Stripes

In below figure we gave an example of a single file spread across four OSTs in five distinct pieces. Now, we add information to that example to show how the stripes are aligned in the logical view of File A. Since the file is spread across 4 OSTs the stripe count is 4. If File A has 9 MB of data and the stripe size is set to 1 MB it can be segmented into 9 equally sized stripes that will be accessed concurrently. The physical and logical views of File A are shown in below figure.



Physical and Logical Views of File A.

In this example, the I/O requests are *stripe aligned*, meaning that the processes access the file at offsets that correspond to stripe boundaries.

Non-aligned Stripes

Next, we give an example where the stripes are not aligned. Four processes write different amounts of data to a single shared File B that is 5 MB in size. The file is striped across 4 OSTs and the stripe size is 1 MB, meaning that the file will require 5 stripes. Each process writes its data as a single contiguous region in File B. No overlaps or gaps between these regions should be present; otherwise the data in the file would be corrupted. The sizes of the four writes and their corresponding offsets are as follows:

- Process 0 writes 0.6 MB starting at offset 0 MB
- Process 1 writes 1.8 MB starting at offset 0.6 MB
- Process 2 writes 1.2 MB starting at offset 2.4 MB
- Process 3 writes 1.4 MB starting at offset 3.6 MB

The logical and physical views of File B are shown in Figure 3.3.5.

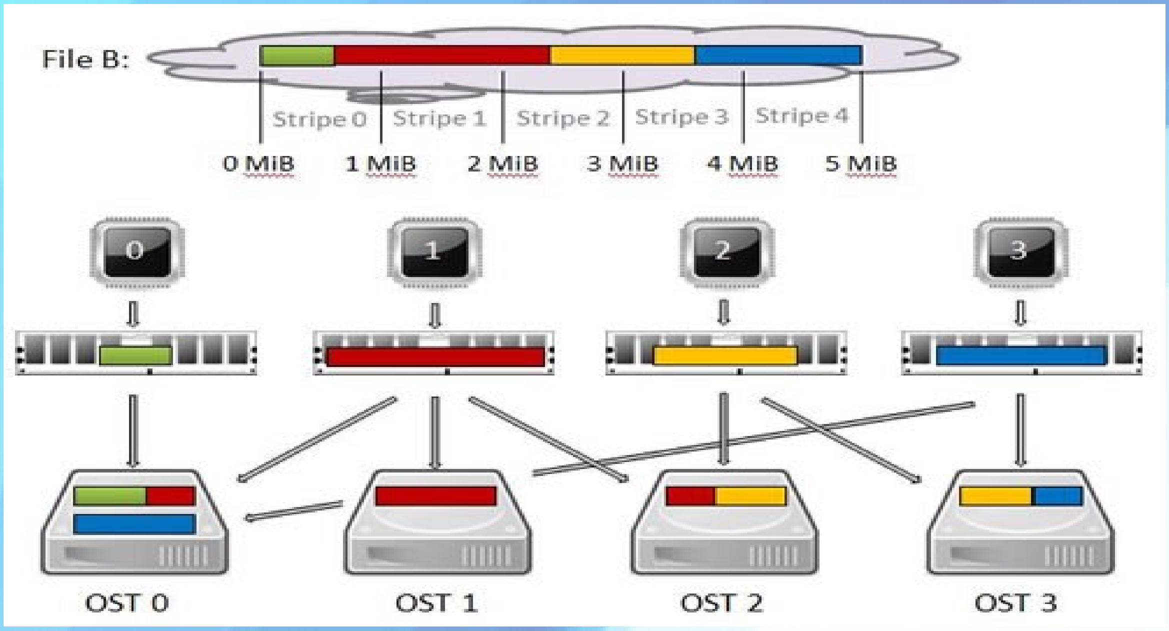


Figure 3.3.5: Logical and Physical Views of File B.

None of the four writes fits the stripe size exactly so Lustre will split each of them into pieces. Since these writes cross an object boundary, they are *not stripe aligned* as in our previous example. When they are not stripe aligned, some of the OSTs are simultaneously receiving data from more than one process. In our non-aligned example, OST 0 is simultaneously receiving data from processes 0, 1 and 3; OST 2 is simultaneously receiving data from processes 1 and 2; and OST 3 is simultaneously receiving data from processes 2 and 3. This creates resource contention on the OST and request contention on the OSS associated with the OST. This contention is a significant performance concern related to striping. It is minimized when processes (that access the file in parallel) access file locations that reside on different stripes as in our stripe aligned example.

Serial I/O

Serial I/O includes those application I/O patterns in which one process performs I/O operations to one or more files. In general, serial I/O is not scalable.

File-per-Process

File-per-process is a communication pattern in which each process of a parallel application writes its data to a private file. This pattern creates N or more files for an application run of N processes. The performance of each process's file write is governed by the statements made above for serial I/O. However, this pattern constitutes the simplest implementation of parallel I/O due to the possibility of improved I/O performance from a parallel file system.

Write performance of a file-per-process I/O pattern as a function of number of files/processes. The file size is 128 MB with 32 MB sized write operations. Performance increases as the number of processes/files increases until OST and metadata contention hinder performance improvements.

- Each file is subject to the limitations of serial I/O.
- Improved performance can be obtained from a parallel file system such as Lustre. However, at large process counts (large number of files) metadata operations may hinder overall performance. Additionally, at large process counts (large number of files) OSS and OST contention will hinder overall performance.

Single-shared-file

A single shared file I/O pattern involves multiple application processes which either independently or concurrently share access to the same file. This particular I/O pattern can take advantage of both process and file system parallelism to achieve high levels of performance. However, at large process counts contention for file system resources OSTs can hinder performance gains.

Two possible shared file layouts. The aggregate file size in both cases is 1 and 2 GB depending on which block size is utilized. The major difference in file layouts is the locality of the data from each process. Layout #1 keeps data from a process in a contiguous block, while Layout #2 strides this data throughout the file. Thirty-two (32) processes will concurrently access this shared file.

Write performance utilizing a single shared file accessed by 32 processes. Stripe counts utilized are 32 (1 GB file) and 64 (2 GB file) with stripe sizes of 32 MB and 1 MB. A 1 MB stripe size on Layout #1 results in the lowest performance due to OST contention. Each OST is accessed by every process. Whereas, the highest performance is seen from a 32 MB stripe size on Layout #1. Each OST is accessed by only one process. A 1 MB stripe size gives better performance with Layout #2. Each OST is accessed by only one process. However, the overall performance is lower due to the increased latency in the write (smaller I/O operations). With a stripe count of 64 each process communicates with 2 OSTs.

Write Performance of a single shared file as the number of processes increases. A file size of 32 MB per process is utilized with 32 MB write operations. For each I/O library (Posix, MPI-IO, and HDF5) performance levels off at high core counts.

- The layout of the single shared file and its interaction with Lustre settings is particularly important with respect to performance.
- At large core counts file system contention limits the performance gains of utilizing a single shared file. The major

limitation is the 160 OST limit on the striping of a single file.

Basic Lustre User Commands

Lustre's **lfs** utility provides several options for monitoring and configuring your Lustre environment. In this section, we describe the basic options that enable you to:

- List OSTs in the File System
- Search the Directory Tree
- Check Disk Space Usage
- Get Striping Information
- Set Striping Patterns

For a complete list of available options, type help at the **lfs** prompt.

```
$ lfs help
```

To get more information on a specific option, type help along with the option name.

```
$ lfs help option-name
```

We are using these commands to mount the lustre .

```
#/etc/init.d/lustre_storage start
```

```
#zpool import <device name>
```

```
#mount -t lustre <device name>/<device name> /lustre/<device name>
```

unmount luster

```
#/etc/init.d/lustre_storage stop
```

```
#umount /lustrefs/<device name>
```

```
#zpool export <device name>
```

Recognize situations where file system contention may limit performance

When an I/O pattern is scaled to large core counts performance degradation may occur due to file system contention. This situation arises when many-many more processes than file system resources request I/O nearly simultaneously. Examples include file-per-process I/O patterns which utilize over ten-thousand processes/files and single-shared-file I/O patterns which utilize over five-thousand processes accessing a single file. Potential solutions involve decreasing the number of processes which perform I/O simultaneously. For a file-per-process pattern this may involve allowing only a subset of processes to perform I/O at any particular time. For a single-shared file pattern this solution may involve utilizing more than one shared-file in which a subset of processes perform I/O. Additionally, some I/O libraries such as MPI-IO allow for collective buffering which aggregates I/O from the running processes onto a subset of processes which perform I/O.

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.

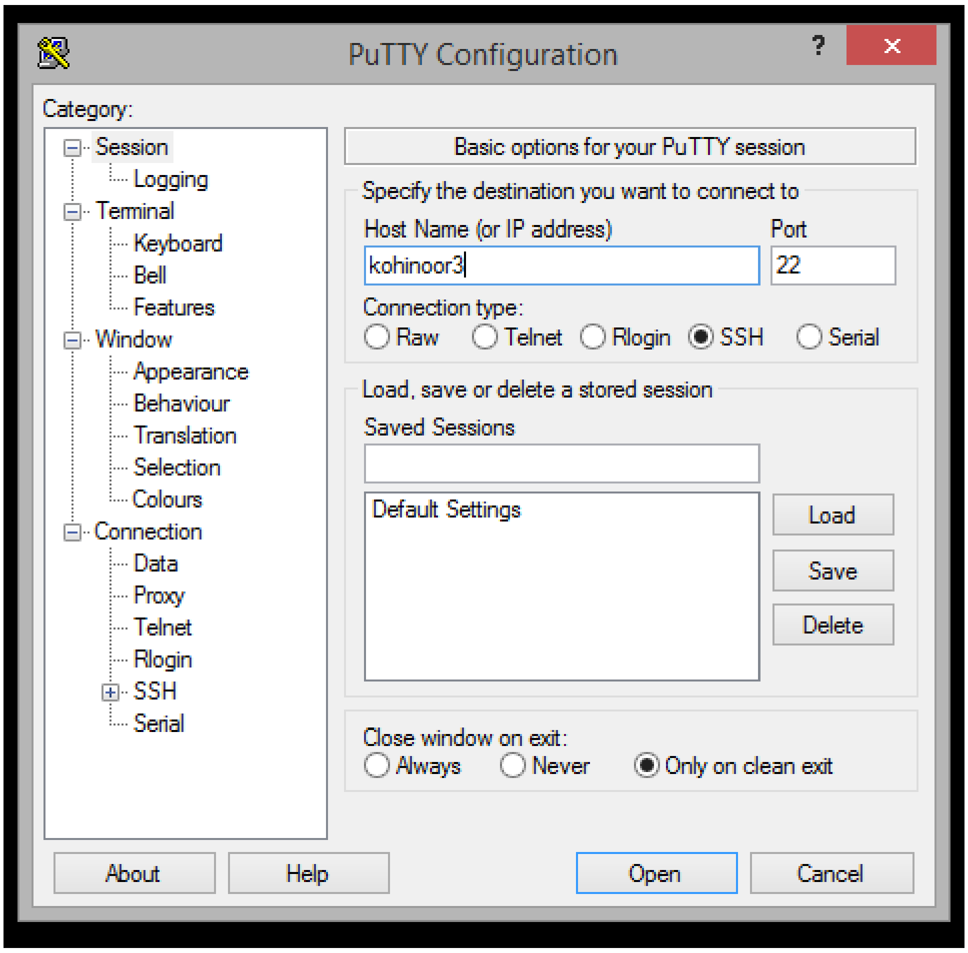
Typically Linux is packaged in a format known as a Linux distribution for desktop and server use. Some popular mainstream

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 <Kohinoor3>



Specify the destination you want to connect to

Host Name (or IP address): kohinoor3

Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

Saved Sessions:

Default Settings

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Login from Linux machine

```
[root@localhost ~] ssh netweb@kohinoor3
Password:
```

Creating User account into the Kohinoor cluster

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

```
# /root/sbin/cluster_useradd <username>
```

It will Ask the Password only one time.

Keep on Pressing Enter key till end.

User deletion:

```
#/root/sbin/cluster_userdel <username>
```

Group Creation:

```
#/root/sbin/cluster_groupadd <groupname>
```

Group deletion:

```
#/root/sbin/cluster_groupdel <groupname>
```

Adding a user to group/groups

```
#/root/sbin/cluster_add_user_to_group <username>
   <group1,group2,etc>
```

Copy a file or directory of every compute node:

```
#module load utils/pdsh
#pdcp -r -a <filename/dictory> <destination directory>
```

Copy a file to defined compute nodes:

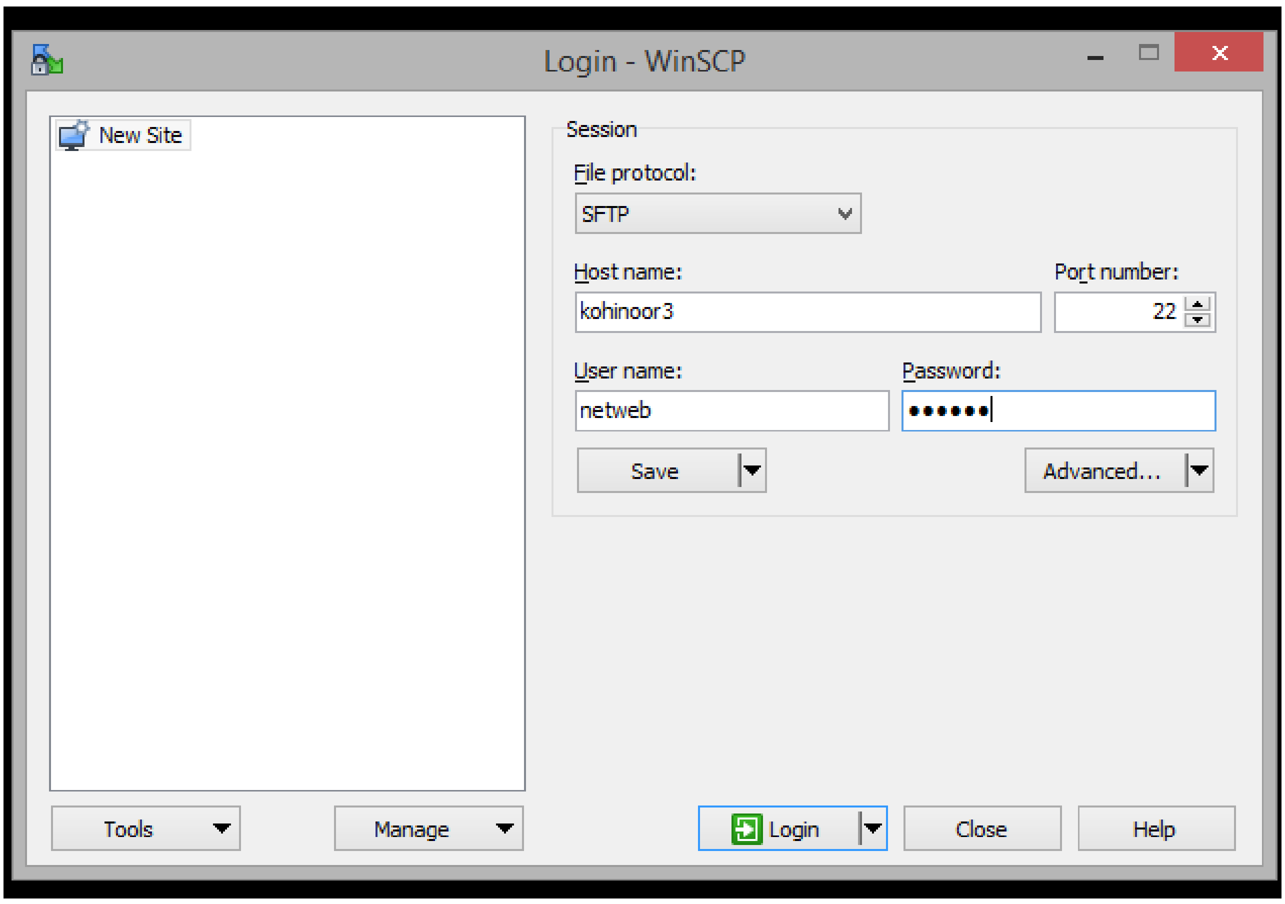
```
# pdcp -r -w node1,node2,etc <filename/dictory> <destination directory>
```

Copy a file to defined compute node:

```
scp -r <filename/folder> <nodename:/path of destination folder>
```

Copy a data from end-user Windows machine to cluster

Step 1:
Install and start WinSCP, then following screen is shown. Click 'Login' button.



File protocol: SFTP

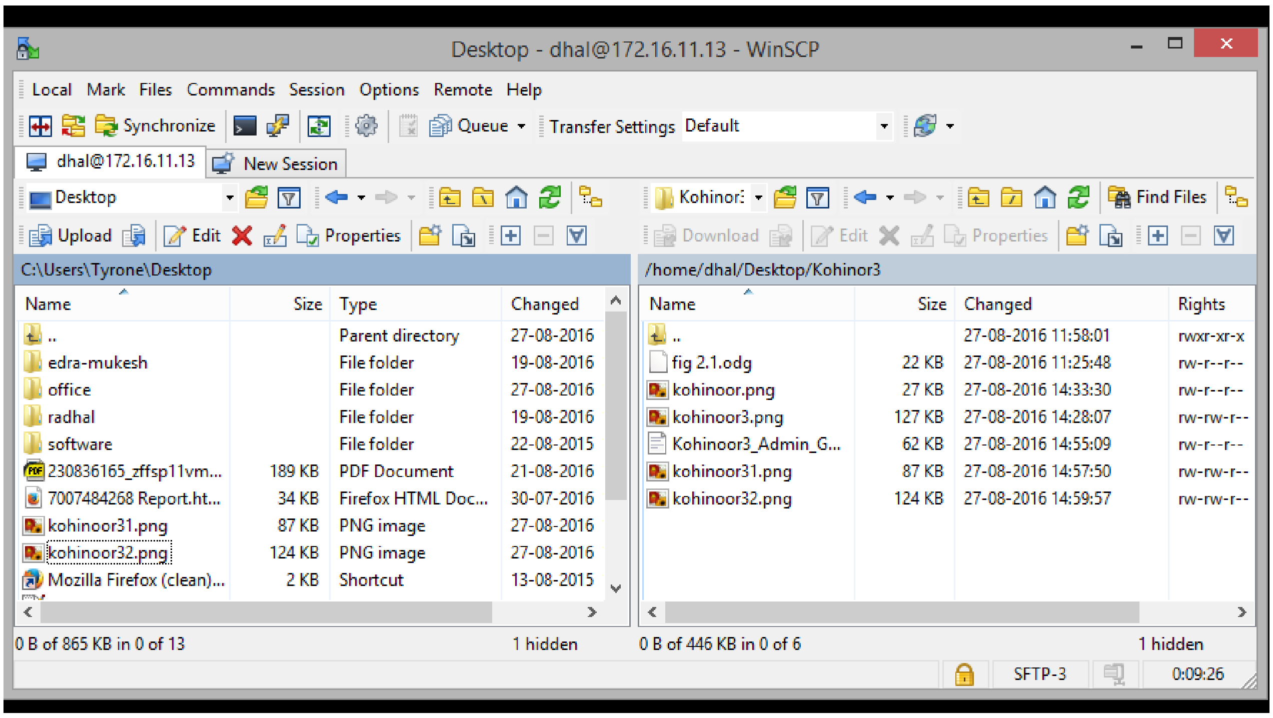
Host name: kohinoor3

Port number: 22

User name: netweb

Password: ••••••

Step 2:
It’s possible to upload or download files simple as drap and drop.



C:\Users\Tyrone\Desktop

Name	Size	Type	Changed
..		Parent directory	27-08-2016
edra-mukesh		File folder	19-08-2016
office		File folder	27-08-2016
radhal		File folder	19-08-2016
software		File folder	22-08-2015
230836165_zffsp11vm...	189 KB	PDF Document	21-08-2016
7007484268 Report.ht...	34 KB	Firefox HTML Doc...	30-07-2016
kohinoor31.png	87 KB	PNG image	27-08-2016
kohinoor32.png	124 KB	PNG image	27-08-2016
Mozilla Firefox (clean)...	2 KB	Shortcut	13-08-2015

0 B of 865 KB in 0 of 13 — 1 hidden

/home/dhal/Desktop/Kohinor3

Name	Size	Changed	Rights
..		27-08-2016 11:58:01	rwxr-xr-x
fig 2.1.odg	22 KB	27-08-2016 11:25:48	rw-r--r--
kohinoor.png	27 KB	27-08-2016 14:33:30	rw-r--r--
kohinoor3.png	127 KB	27-08-2016 14:28:07	rw-rw-r--
Kohinoor3_Admin_G...	62 KB	27-08-2016 14:55:09	rw-r--r--
kohinoor31.png	87 KB	27-08-2016 14:57:50	rw-rw-r--
kohinoor32.png	124 KB	27-08-2016 14:59:57	rw-rw-r--

0 B of 446 KB in 0 of 6 — 1 hidden

Copy a data from end-user Linux machine to cluster

```
[root@localhost ~] scp <filename> support@172.16.10.24
```

Copying a file into user netweb home directory

```
[root@localhost ~] scp –r <directory> support@172.16.10.24
```

Copying a directory into user netweb home directory

Netweb TECHNOLOGIES

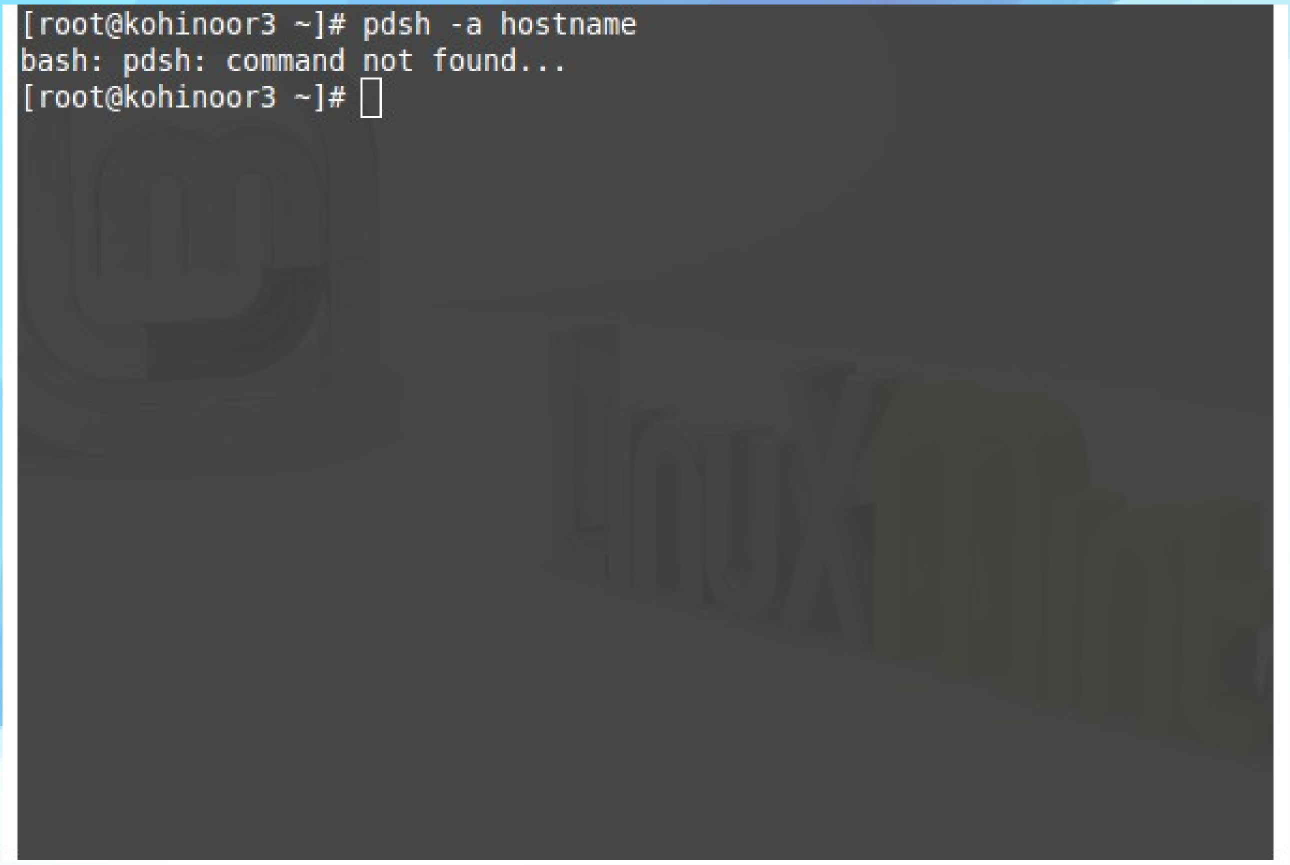
3.6 Module Management:

The Environment Modules package provides for the dynamic modification of a user's environment via modulefiles.

Each modulefile contains the information needed to configure the shell for an application. Once the Modules package is initialized, the environment can be modified on a per-module basis using the module command which interprets modulefiles. Typically modulefiles instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc. modulefiles may be shared by many users on a system and users may have their own collection to supplement or replace the shared modulefiles.

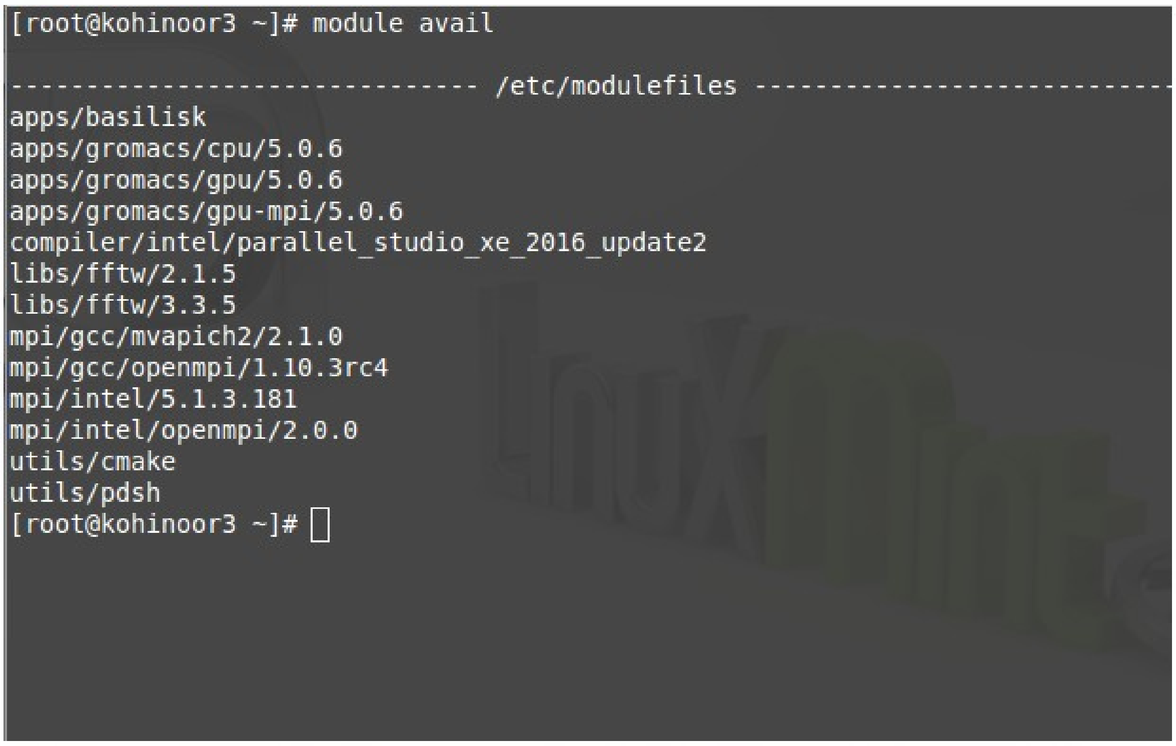
Modules can be **loaded** and **unloaded** dynamically and atomically, in an clean fashion.

Module need to load some as example if you run pdsh command and you are getting following message.



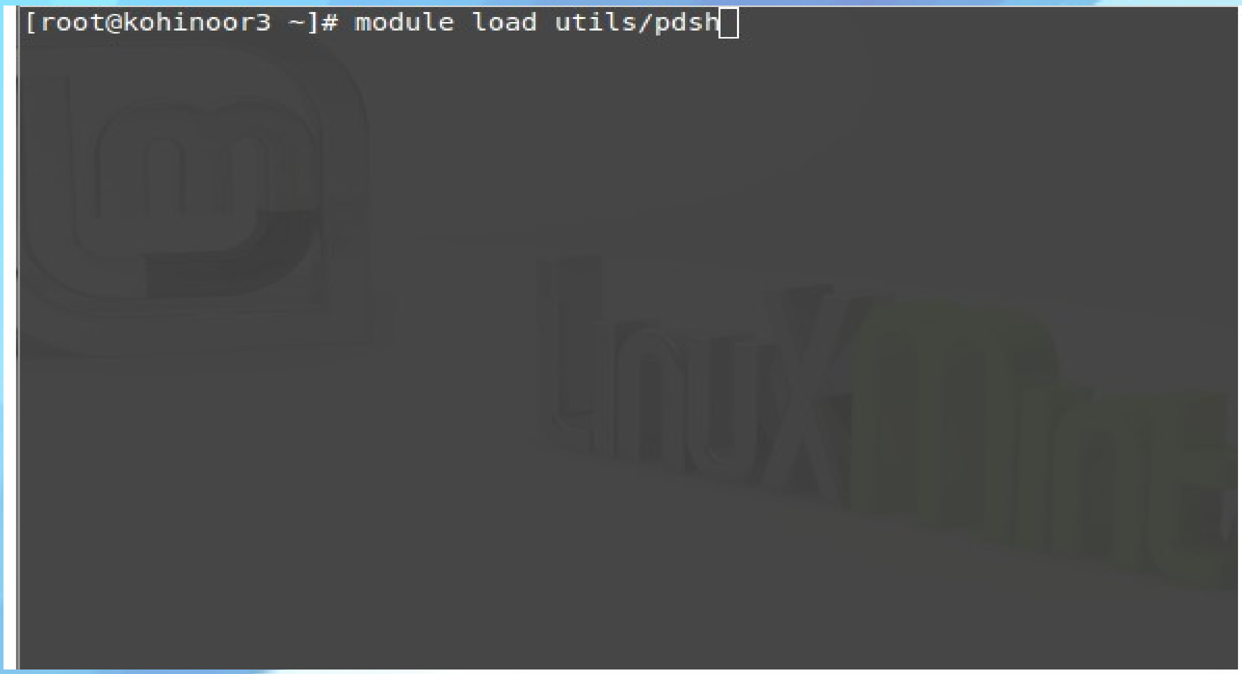
```
[root@kohinoor3 ~]# pdsh -a hostname
bash: pdsh: command not found...
[root@kohinoor3 ~]# 
```

Then you can find that module is avail for that application by the command module avail.



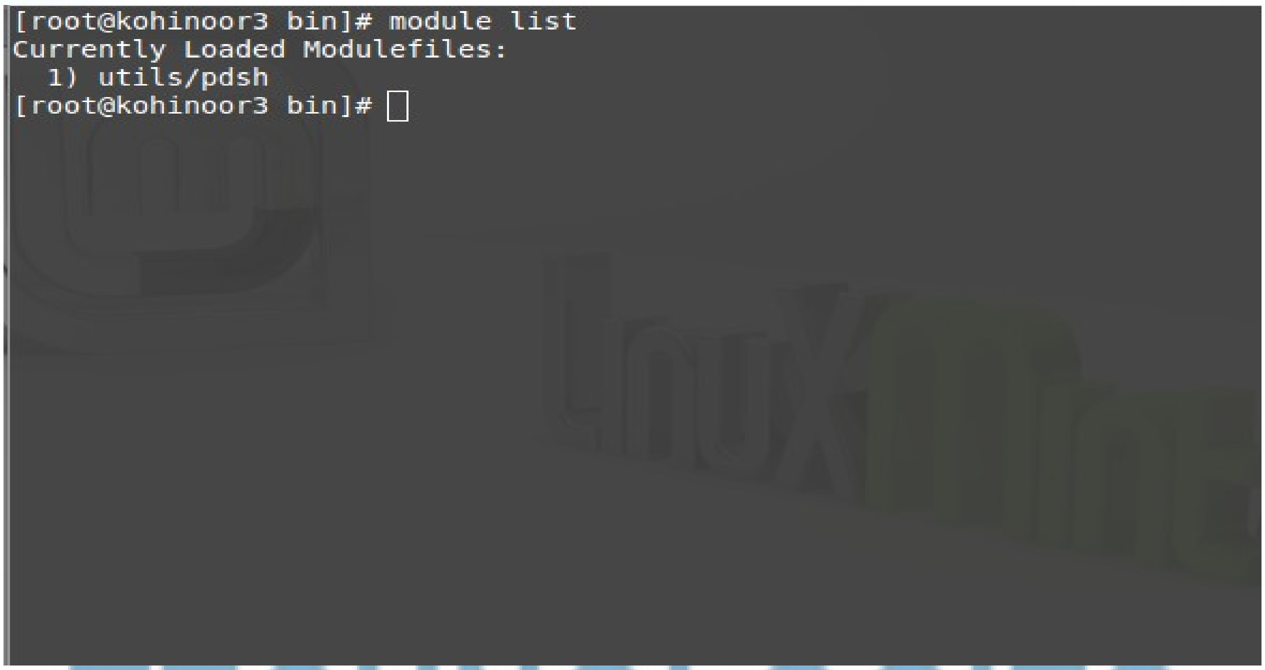
```
[root@kohinoor3 ~]# module avail

------------------------------- /etc/modulefiles ----------------------------
apps/basilisk
apps/gromacs/cpu/5.0.6
apps/gromacs/gpu/5.0.6
apps/gromacs/gpu-mpi/5.0.6
compiler/intel/parallel_studio_xe_2016_update2
libs/fftw/2.1.5
libs/fftw/3.3.5
mpi/gcc/mvapich2/2.1.0
mpi/gcc/openmpi/1.10.3rc4
mpi/intel/5.1.3.181
mpi/intel/openmpi/2.0.0
utils/cmake
utils/pdsh
[root@kohinoor3 ~]# 
```

For loading that module you have to load the module Example: module load utils/pdsh.



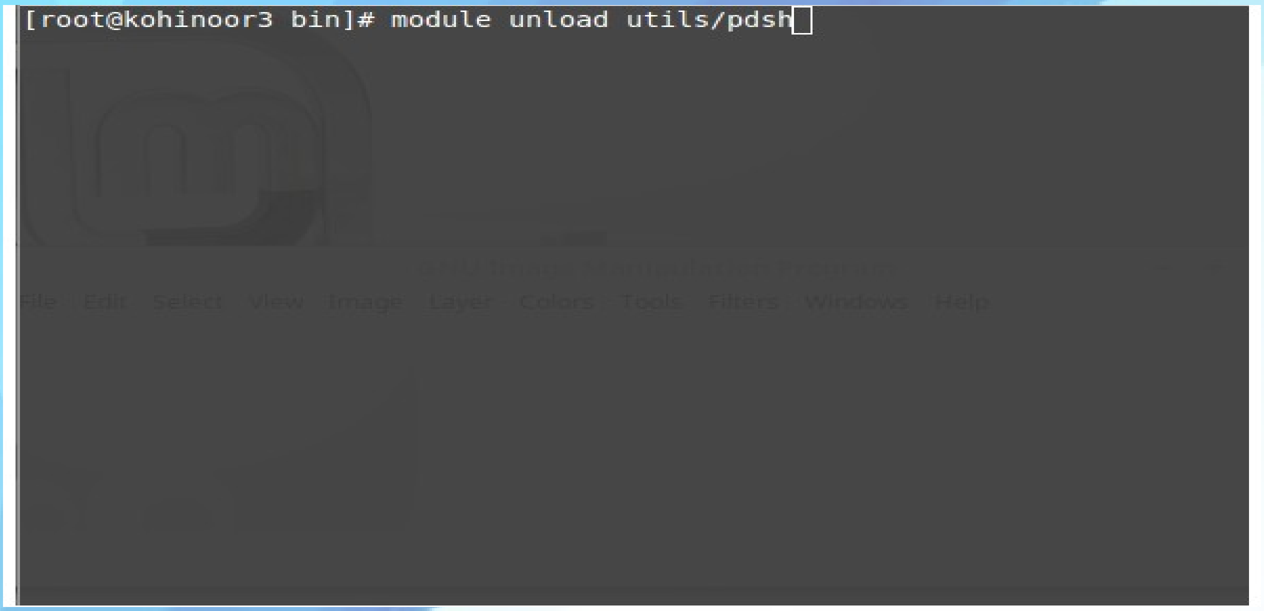
```
[root@kohinoor3 ~]# module load utils/pdsh
```

We can check which modules are loaded by the command module list.



```
[root@kohinoor3 bin]# module list
Currently Loaded Modulefiles:
  1) utils/pdsh
[root@kohinoor3 bin]# 
```

We can unload the module as well by the command module unload.



```
[root@kohinoor3 bin]# module unload utils/pdsh
```

3.7. 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 = /apps/mpi/mvapich2-2.1-gcc/bin
mpic++ = /apps/mpi/mvapich2-2.1-gcc/bin
mpif77 = /apps/mpi/mvapich2-2.1-gcc/bin
mpif90 = /apps/mpi/mvapich2-2.1-gcc/bin

Applications:

List of Application

Application	CPU
Gromacs	YES
LAMPS	YES
FFTW	YES

GROMACS
CPU:
Location = /apps/gromacs
Executable file = /apps/gromacs/5.0.6-cpu/bin/GMXRC
GPU:
Location = /apps/gromacs
Executable file = /apps/gromacs/5.0.6-gpu/bin/GMXRC

LAMPS

CPU:
Location = /apps/lamps
Executable file = /apps/lamps/lmp_icc_openmpi

FFTW(2.1.5)

Location = /apps/libs/fftw-2.1.5/

FFTW(3.3.5)

Location = /apps/libs/fftw-3.3.5/

3.8. Scheduler & Distributed Resource Manager (SLURM)

The Simple Linux Utility for Resource Management (Slurm) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. Slurm requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, Slurm has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes.

Commands

Man pages exist for all Slurm daemons, commands, and API functions. The command option --help also provides a brief summary of options. Note that the command options are all case sensitive.

sacct is used to report job or job step accounting information about active or completed jobs.

salloc is used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.

sattach is used to attach standard input, output, and error plus signal capabilities to a currently running job or job step. One can attach to and detach from jobs multiple times.

sbatch is used to submit a job script for later execution. The script will typically contain one or more srun commands to launch parallel tasks.

sbcast is used to transfer a file from local disk to local disk on the nodes allocated to a job. This can be used to effectively use diskless compute nodes or provide improved performance relative to a shared file system.

scancel is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

scontrol is the administrative tool used to view and/or modify Slurm state. Note that many scontrol commands can only be executed as user root.

sinfo reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

smap reports state information for jobs, partitions, and nodes managed by Slurm, but graphically displays the information to reflect network topology.

squeue reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.

srun is used to submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared resources within the job's node allocation.

strigger is used to set, get or view event triggers. Event triggers include things such as nodes going down or jobs approaching their time limit.

sview is a graphical user interface to get and update state information for jobs, partitions, and nodes managed by Slurm.

Example

The sinfo command has many options to easily let you view the information of interest to you in whatever format you prefer. See the man page for more information.

```
# sinfo
PARTITION AVAIL  TIMELIMIT NODES  STATE NODELIST
debug*       up       30:00         2          down*  node[1-2]
debug*        up       30:00         3          idle       node[3-5]
batch          up       30:00         3         down*   node[6,13,15]
batch          up        30:00         3         alloc     node[7-8,14]
batch          up        30:00         4         idle        node[9-12]
```

Next we determine what jobs exist on the system using the squeue command. The ST field is job state. Two jobs are in a running state (R is an abbreviation for Running) while one job is in a pending state (PD is an abbreviation for Pending). The TIME field shows how long the jobs have run for using the format days-hours:minutes:seconds. The NODELIST(REASON) field indicates where the job is running or the reason it is still pending. Typical reasons for pending jobs are Resources (waiting for resources to become available) and Priority (queued behind a higher priority job). The squeue command has many options to easily let you view the information of interest to you in whatever format you prefer. See the man page for more information.

```
# squeue
JOBID PARTITION  NAME  USER ST  TIME NODES NODELIST(REASON)
65646     batch  chem  mike  R 24:19     2 node[7-8]
65647     batch   bio  joan  R  0:09     1 node14
65648     batch  math  phil PD  0:00     6 (Resources)
```

One common mode of operation is to submit a script for later execution. In this example the script name is *my.script* and we explicitly use the nodes node9 and node10 (*-w "node[9-10]"*, note the use of a node range expression). We also explicitly state that the subsequent job steps will spawn four tasks each, which will insure that our allocation contains at least four processors (one processor per task to be launched). The output will appear in the file my.stdout ("-o my.stdout"). This script contains a timelimit for the job embedded within itself. Other options can be supplied as desired by using a prefix of "#SBATCH" followed by the option at the beginning of the script (before any commands to be executed in the script). Options supplied on the command line would override any options specified within the script. Note that my.script contains the command /bin/hostname that executed on the first node in the allocation (where the script runs) plus two job steps initiated using the srun command and executed sequentially.

```
# cat my.script
#!/bin/sh
#SBATCH --time=1
/bin/hostname
srun -l /bin/hostname
srun -l /bin/pwd

# sbatch -n4 -w "node[9-10]" -o my.stdout my.script
sbatch: Submitted batch job 469

node1: cat my.stdout
node9
0: node9
1: node9
2: node10
3: node10
```

The final mode of operation is to create a resource allocation and spawn job steps within that allocation. The salloc command is used to create a resource allocation and typically start a shell within that allocation. One or more job steps would typically be executed within that allocation using the srun command to launch the tasks . Finally the shell created by salloc would be terminated using the *exit* command. Slurm does not automatically migrate executable or data files to the nodes allocated to a job. Either the files must exists on local disk or in some global file system (e.g. NFS or Lustre). We provide the tool sbcast to transfer files to local storage on allocated nodes using Slurm's hierarchical communications. In this example we use sbcast to transfer the executable program *a.out* to */tmp/joe.a.out* on local storage of the allocated nodes. After executing the program, we delete it from local storage.

```
# salloc -N1024 bash

$ sbcast a.out /tmp/joe.a.out

Granted job allocation 471

$ srun /tmp/joe.a.out

Result is 3.14159

$ srun rm /tmp/joe.a.out

$ exit

salloc: Relinquishing job allocation 471
```

In this example, we submit a batch job, get its status, and cancel it.

```
node1: sbatch test

srun: jobid 473 submitted
```

```
node1: squeue
JOBID    PARTITION    NAME    USER    ST    TIME        NODES
NODELIST(REASON)
  473 batch      test  jill   R  00:00   1    node9
node1: scancel 473
node1: squeue
JOBID PARTITION NAME USER ST TIME  NODES NODELIST(REASON)
```

SLURM Script on Kohinoor3

We have given submit.sh in so you can use that script as as demo.

```
 #!/bin/bash -l
#SBATCH -p compute
### -p  partition Name
#SBATCH -N 2
### -N number of nodes
#SBATCH -n 40
### -n Number of tasks
#SBATCH -c 1
### -c number of CPUs per task
#SBATCH -t 03:00:00
## -t  time allocate

module load mpi/gcc/openmpi/2.0.1
# mpicc  -o cpi  cpi.c
#export OMP_NUM_THREADS=8
# thereads are needed in the program .. then export omp tnum
threads
mpirun ./cpi
```

3.9. Troubleshooting SLURM

This guide is meant as a tool to help system administrators or operators troubleshoot Slurm failures and restore services.

- Slurm is not responding
- Jobs are not getting scheduled
- Jobs and nodes are stuck in COMPLETING state
- Notes are getting set to a DOWN state
- Networking and configuration problems

Slurm is not responding

1. Execute "scontrol ping" to determine if the primary and backup controllers are responding.
2. If it responds for you, this could be a networking or configuration problem specific to some user or node in the cluster.
3. If not responding, directly login to the machine and try again to rule out network and configuration problems.
4. If still not responding, check if there is an active slurmctld daemon by executing "ps -el | grep slurmctld".
5. If slurmctld is not running, restart it (typically as user root using the command "/etc/init.d/slurm start"). You should check the log file (SlurmctldLog in the slurm.conf file) for an indication of why it failed. If it keeps failing, you should Contact your Administrator.
6. If slurmctld is running but not responding (a very rare

situation), then kill and restart it (typically as user root using the commands "/etc/init.d/slurm stop" and then "/etc/init.d/slurm start").

7. If it hangs again, increase the verbosity of debug messages (increase SlurmctldDebug in the slurm.conf file) and restart. Again check the log file for an indication of why it failed. At this point, you should Contact your Administrator.

8. If it continues to fail without an indication as to the failure mode, restart without preserving state (typically as user root using the commands "/etc/init.d/slurm stop" and then "/etc/init.d/slurm startclean"). Note: All running jobs and other state information will be lost.

Jobs are not getting scheduled

This is dependent upon the scheduler used by Slurm. Executing the command "scontrol show config | grep SchedulerType" to determine this. For any scheduler, you can check priorities of jobs using the command "scontrol show job".

- If the scheduler type is builtin, then jobs will be executed in the order of submission for a given partition. Even if resources are available to initiate jobs immediately, it will be deferred until no previously submitted job is pending.
- If the scheduler type is backfill, then jobs will generally be executed in the order of submission for a given partition with one exception. Otherwise Contact your Administrator.

Jobs and nodes are stuck in COMPLETING state

This is typically due to non-killable processes associated with the job. Slurm will continue to attempt terminating the processes with SIGKILL, but some jobs may be stuck performing I/O and non-killable. This is typically due to a file system problem and may be addressed in a couple of ways.

1. Fix the file system and/or reboot the node. -OR-
2. Set the node to a DOWN state and then return it to service ("scontrol update NodeName=<node> State=down Reason=hung_proc" and "scontrol update NodeName=<node> State=resume"). This permits other jobs to use the node, but leaves the non-killable process in place. If the process should ever complete the I/O, the pending SIGKILL should terminate it immediately. -OR-
3. Use the UnkillableStepProgram and UnkillableStepTimeout configuration parameters to automatically respond to processes which can not be killed, by sending email or rebooting the node. For more information, see the slurm.conf documentation.

Notes are getting set to a DOWN state

1. Check the reason why the node is down using the command "scontrol show node <name>". This will show the reason why the node was set down and the time when it happened. If there is insufficient disk space, memory space, etc. compared to the parameters specified in the slurm.conf file then either fix the node or change slurm.conf.
2. If the reason is "Not responding", then check communications between the control machine and the DOWN node using the command "ping <address>" being sure to specify the NodeAddr values configured in slurm.conf. If ping fails, then fix the network or addresses in slurm.conf.
3. Next, login to a node tha. Slurm considers to be in a DOWN state and check if the slurmd daemon is running with the command "ps -el | grep slurmd". If slurmd is not running,

restart it (typically as user root using the command "/etc/init.d/slurm start"). You should check the log file (SlurmdLog in the slurm.conf file) for an indication of why it failed. You can get the status of the running slurmd daemon by executing the command "scontrol show slurmd" on the node of interest. Check the value of "Last slurmctld msg time" to determine if the slurmctld is able to communicate with the slurmd. If it keeps failing, you should Contact your Administrator.

4. If slurmd is running but not responding (a very rare situation), then kill and restart it (typically as user root using the commands "/etc/init.d/slurm stop" and then "/etc/init.d/slurm start").
5. If still not responding, try again to rule out network and configuration problems.
6. If still not responding, increase the verbosity of debug messages (increase SlurmdDebug in the slurm.conf file) and restart. Again check the log file for an indication of why it failed. At this point, you should Contact your Administrator.

7. If still not responding without an indication as to the failure mode, restart without preserving state (typically as user root using the commands "/etc/init.d/slurm stop" and then "/etc/init.d/slurm startclean"). Note: All jobs and other state information on that node will be lost.

Networking and configuration problems

1. Check the controller and/or slurmd log files (SlurmctldLog and SlurmdLog in the slurm.conf file) for an indication of why it is failing.

2. Check for consistent slurm.conf and credential files on the node(s) experiencing problems.
3. If this is user-specific problem, check that the user is configured on the controller computer(s) as well as the compute nodes. The user doesn't need to be able to login, but his user ID must exist.
4. Check that compatible versions of Slurm exists on all of the nodes (execute "sinfo -V" or "rpm -qa | grep slurm"). The Slurm version numbers contain three digits, which represent the major, minor and micro release numbers in that order (e.g. 14.11.3 is major=14, minor=11, micro=3). Changes in the RPCs (remote procedure calls) and state files will only be made if the major and/or minor release number changes. Slurm daemons will support RPCs and state files from the two previous minor or releases (e.g. a version 15.08.x SlurmDBD will support slurmctld daemons and commands with a version of 14.03.x or 14.11.x).

3.10. 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.

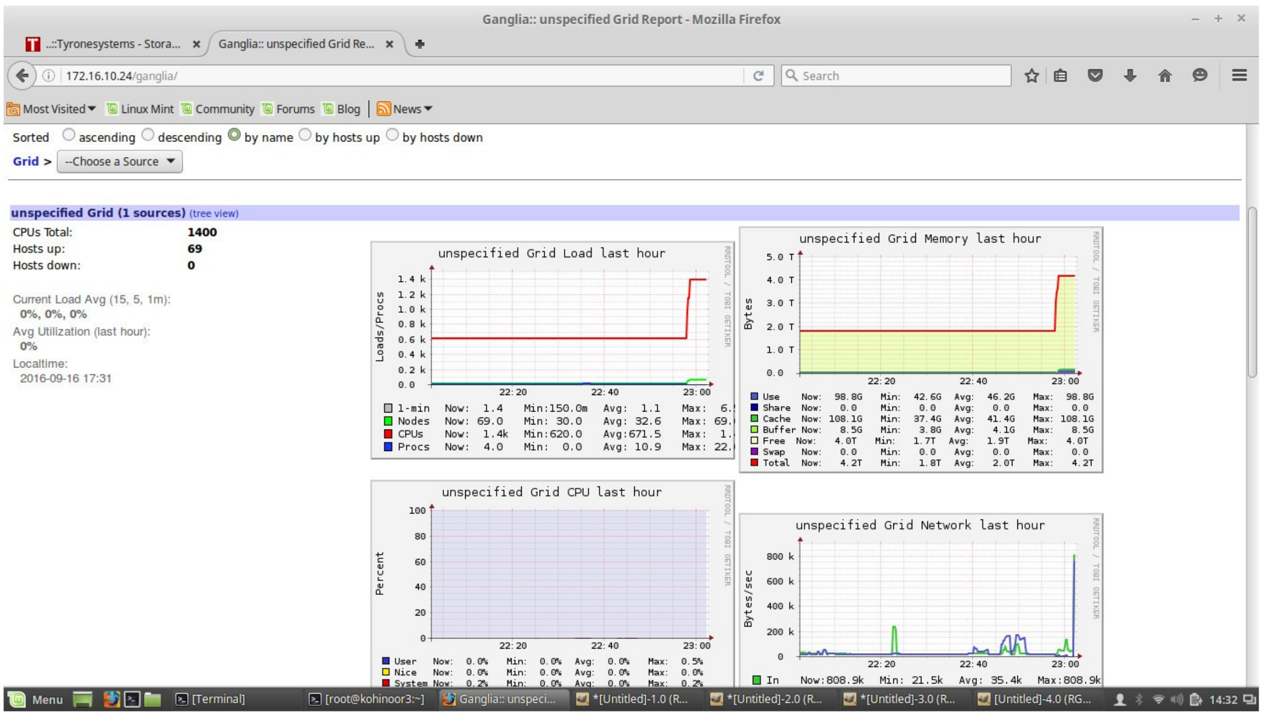
There are three services are mandatory for Ganglia. At server, httpd, gmetad and gmond, and at client, gmond only. If user wants to restart these services, he has to login as root, and restart three services at master node and one at every compute nodes:

```
# systemctl restart httpd
# systemctl restart gmetd
# systemctl restart gmond
```

To open the web interface of Ganglia, type the following URL in browser:

kohinoor3/ganglia

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



Sorted ( ) ascending ( ) descending [✓ by name] ( ) by hosts up ( ) by hosts down

unspecified Grid (1 sources) (tree view)

CPUs Total:	**1400**
Hosts up:	**69**
Hosts down:	**0**

Current Load Avg (15, 5, 1m):
0%, 0%, 0%
Avg Utilization (last hour):
0%
Localtime:
2016-09-16 17:31

```
1-min   Now:  1.4   Min:150.0m   Avg:  1.1   Max:  6.
Nodes   Now: 69.0   Min: 30.0    Avg: 32.6   Max: 69.
CPUs    Now:  1.4k  Min:620.0    Avg:671.5   Max:  1.
Procs   Now:  4.0   Min:  0.0    Avg: 10.9   Max: 22.
```

```
Use    Now:  98.8G   Min:  42.6G   Avg:  46.2G   Max:  98.8G
Share  Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
Cache  Now: 108.1G   Min:  37.4G   Avg:  41.4G   Max: 108.1G
Buffer Now:   8.5G   Min:   3.8G   Avg:   4.1G   Max:   8.5G
Free  Now:   4.0T   Min:   1.7T   Avg:   1.9T   Max:   4.0T
Swap   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
Total  Now:   4.2T   Min:   1.8T   Avg:   2.0T   Max:   4.2T
```

```
User   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.5%
Nice   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
System Now:  0.2%   Min:  0.0%   Avg:  0.0%   Max:  0.2%
```

```
In    Now:808.9k   Min: 21.5k   Avg: 35.4k   Max:808.9k
```

3.11. **System Management:**

The Intelligent Platform Management Interface (IPMI) is a set of computer interface specifications for an autonomous computer subsystem that provides management and monitoring capabilities independently of the host system's CPU, firmware (BIOS or UEFI) and operating system. IPMI defines a set of interfaces used by system administrators for out-of-band management of computer systems and monitoring of their operation. For example, IPMI provides a way to manage a computer that may be powered off or otherwise unresponsive by using a network connection to the hardware rather than to an operating system or login shell.

Functionality:

Using a standardized interface and protocol allows systems-management software based on IPMI to manage multiple disparate servers. As a message-based, hardware-level interface specification, IPMI operates independently of the operating system (OS) to allow administrators to manage a system remotely in the absence of an operating system or of the system management software. Thus IPMI functions can work in any of three scenarios:

- before an OS has booted (allowing, for example, the remote monitoring or changing of BIOS settings)
- when the system is powered down
- after OS or system failure– the key characteristic of IPMI compared with in-band system management such as by remote login to the operating system using SSH

System administrators can use IPMI messaging to monitor platform status (such as system temperatures, voltages, fans, power supplies and chassis intrusion); to query inventory information; to review hardware logs of out-of-range conditions; or to perform recovery procedures such as issuing requests from a remote console through the same connections e.g. system power-down and rebooting, or configuring watchdog timers. The standard also defines an alerting mechanism for the system to send a simple network management protocol (SNMP) platform event trap (PET).

The monitored system may be powered off, but must be connected to a power source and to the monitoring medium, typically a local area network (LAN) connection. IPMI can also function after the operating system has started, and exposes management data and structures to the system management software. IPMI prescribes only the structure and format of the interfaces as a standard, while detailed implementations may vary. An implementation of IPMI version 1.5 can communicate via a direct out-of-band local area network (LAN) or serial connection or via a side-band local area network (LAN) connection to a remote client. The side-band LAN connection utilizes the board network interface controller (NIC). This solution is less expensive than a dedicated LAN connection but also has limited bandwidth.

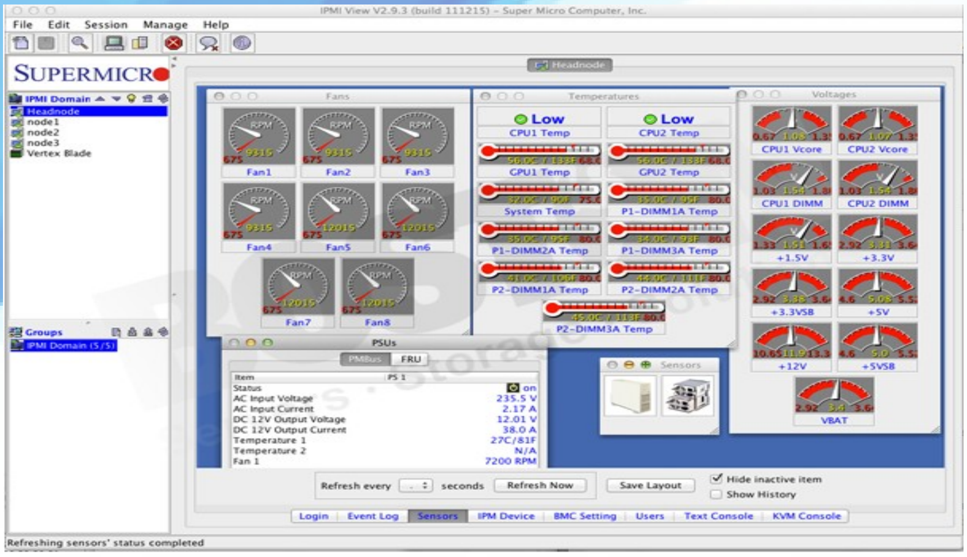
Systems compliant with IPMI version 2.0 can also communicate via serial over LAN, whereby serial console output can be remotely viewed over the LAN. Systems implementing IPMI 2.0 typically also include KVM over IP, remote virtual media and out-of-band embedded web-server interface functionality, although strictly speaking, these lie outside of the scope of the IPMI interface standard.

IPMI runs on a separate hardware subsystem directly attached to a motherboard / server; either hard wired onboard or as an add-in card, this hardware is referred to as a baseboard management controller (BMC).

The BMC functions separately to the motherboard and runs its own independent software stack or firmware to the motherboard it is controlling and monitoring. This enables the administrator to connect to the BMC and control and monitor the system even if it is powered down, crashed or without any O/S.

Typical features of an IPMI BMC are as follows:

- **Hardware monitoring:** CPU / system temperatures, fan speeds / status, power supply status and chassis intrusion can be monitored remotely. In the event of failures or predefined thresholds being exceeded an event is logged and email notifications can be sent to an administrator for immediate action.



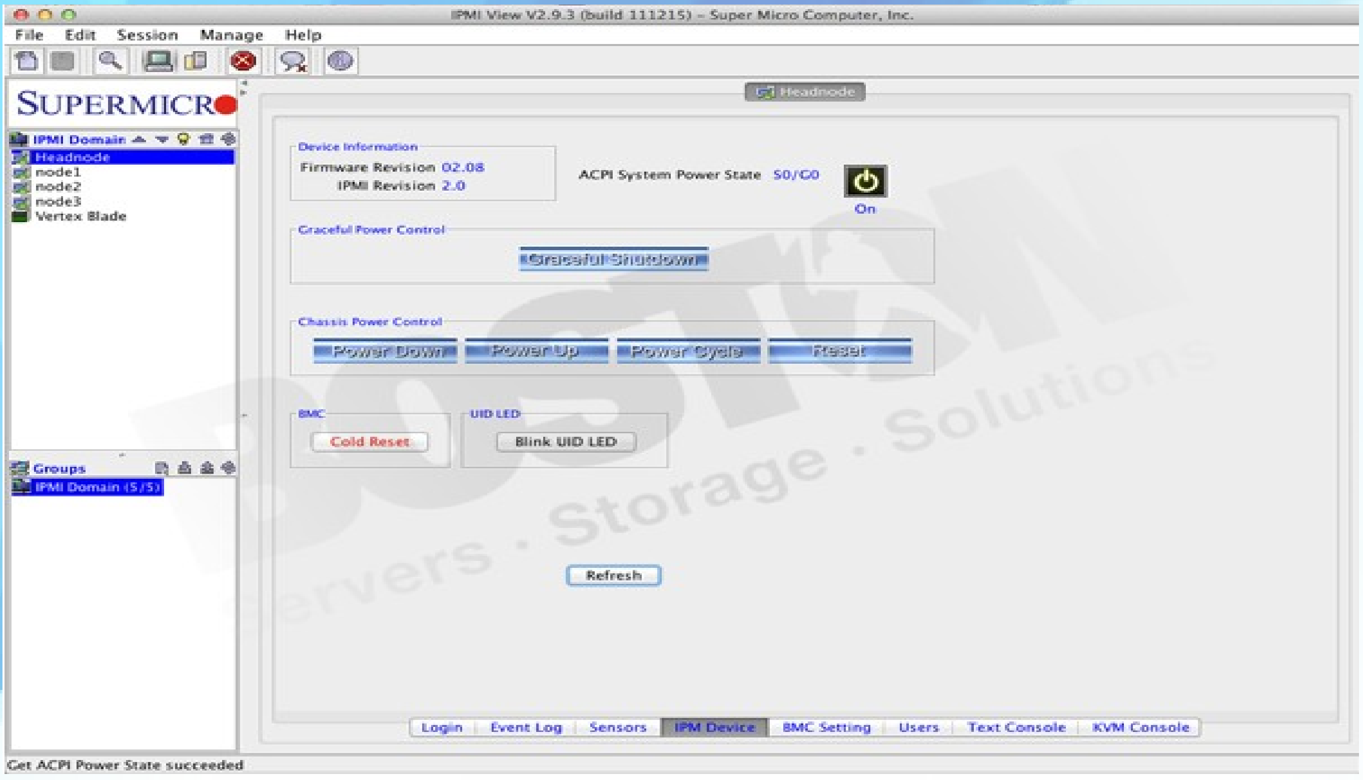
Item	PS 1
Status	on
AC Input Voltage	235.5 V
AC Input Current	2.17 A
DC 12V Output Voltage	12.01 V
DC 12V Output Current	38.0 A
Temperature 1	27C/81F
Temperature 2	N/A
Fan 1	7200 RPM

Refreshing sensors' status completed

Example sensor readout from Supermicro's IPMI View

- **Remote Power Control:**

  Power On, Power Off, Reset & Power Cycle servers remotely. This feature is useful to control power and shut down systems when not in use. In the event of an operating system crash, it's possible to reboot a system and bring it back online. Additionally, in order to identify a system in a crowded data centre there is a UID LED which can be blinked on the front and back of the system to enable technicians to easily identify this.



Device Information
Firmware Revision 02.08
IPMI Revision 2.0

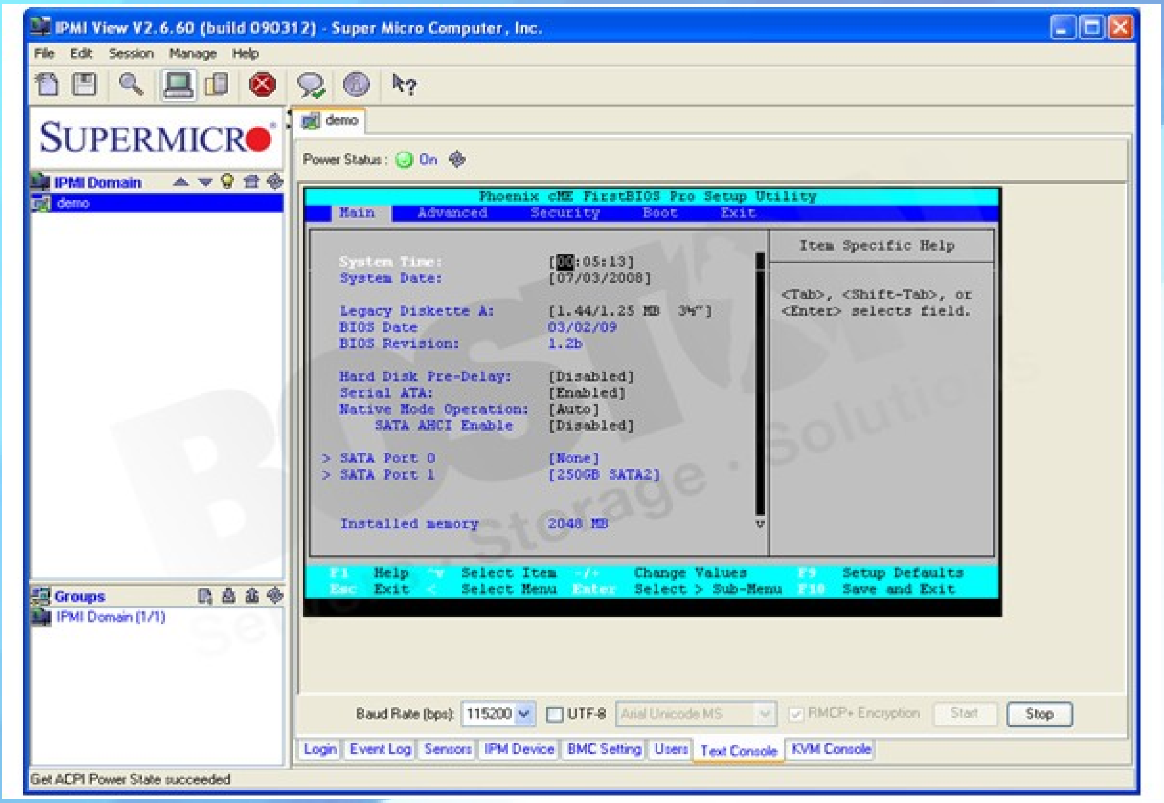
ACPI System Power State S0/G0
On

Get ACPI Power State succeeded

Example power control in Supermicro's IPMI View

- **Remote Control:**

  Serial over LAN (SOL) enables a basic text output of the screen and remote control for diagnostic and administration of CLI based applications and operating systems. This feature is often used by Linux & UNIX administrators and also by some Windows administrators via the EMS (Emergency Management Services) feature.


IPMI View V2.6.60 (build 090312) - Super Micro Computer, Inc.
File Edit Session Manage Help
SUPERMICRO
IPMI Domain
demo
Groups
IPMI Domain (1/1)
demo
Baud Rate (bps): 115200
UTF-8
Arial Unicode MS
RMCP+ Encryption
Start
Stop
Login Event Log Sensors IPM Device BMC Setting Users Text Console KVM Console
Get ACPI Power State succeeded

```
Power Status : On

                    Phoenix cME FirstBIOS Pro Setup Utility
   Main     Advanced     Security     Boot     Exit

     System Time:               [00:05:13]               Item Specific Help
     System Date:               [07/03/2008]
                                                        <Tab>, <Shift-Tab>, or
     Legacy Diskette A:         [1.44/1.25 MB  3½"]     <Enter> selects field.
     BIOS Date                  03/02/09
     BIOS Revision:             1.2b

     Hard Disk Pre-Delay:       [Disabled]
     Serial ATA:                [Enabled]
     Native Mode Operation:     [Auto]
         SATA AHCI Enable       [Disabled]

   > SATA Port 0                [None]
   > SATA Port 1                [250GB SATA2]


     Installed memory           2048 MB

   F1   Help   ↑↓  Select Item   -/+    Change Values        F9   Setup Defaults
   Esc  Exit   ←→  Select Menu   Enter  Select > Sub-Menu    F10  Save and Exit
```

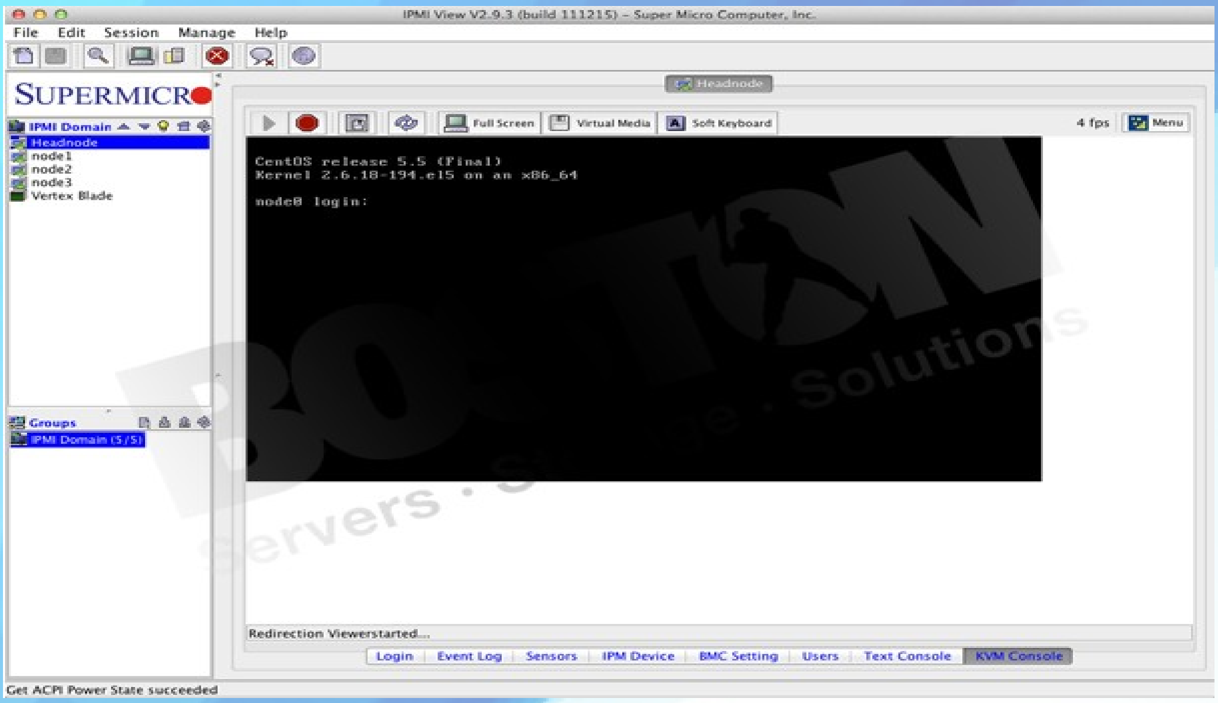
Example SOL Output from Supermicro's IPMI View

IPMI +

Additionally to these features many system vendors build on this base and create a more complete remote management solution. Supermicro for example support the following on almost all of their X8 and X9 based motherboards.

- **KVM over IP Support:**

  Using a Java based console it's possible to gain full graphical KVM access to a system over an IP network. This allows access at all times, even before an O/S has booted, this means that you can gain access to the BIOS or DOS applications, It's even possible to perform installations of Linux and Windows remotely from this console.



```
CentOS release 5.5 (Final)
Kernel 2.6.18-194.el5 on an x86_64

node0 login:
```

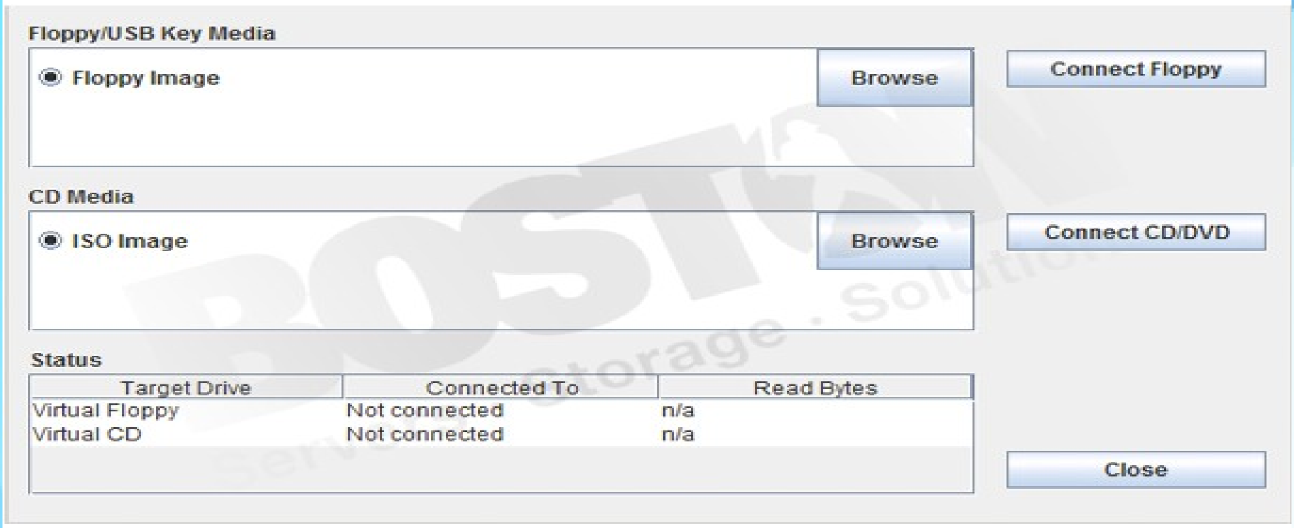
Redirection Viewerstarted...

Get ACPI Power State succeeded

Example KVM over IP output from Supermicro's IPMI View

- **Remote Media Redirection:**

  Typically integrated into the KVM over IP support this feature enables the administrator to physically attach USB storage devices to the remote controlled system. This is in the format of a local physical drive such as a USB pen drive attached to the administrators system or an image file of a DVD/CD in ISO format or even a floppy raw disk image. This means that when combined with KVM over IP O/S installs and firmware / BIOS updates are possible remotely without any need for local hands on support.


Floppy/USB Key Media

● Floppy Image

CD Media

● ISO Image

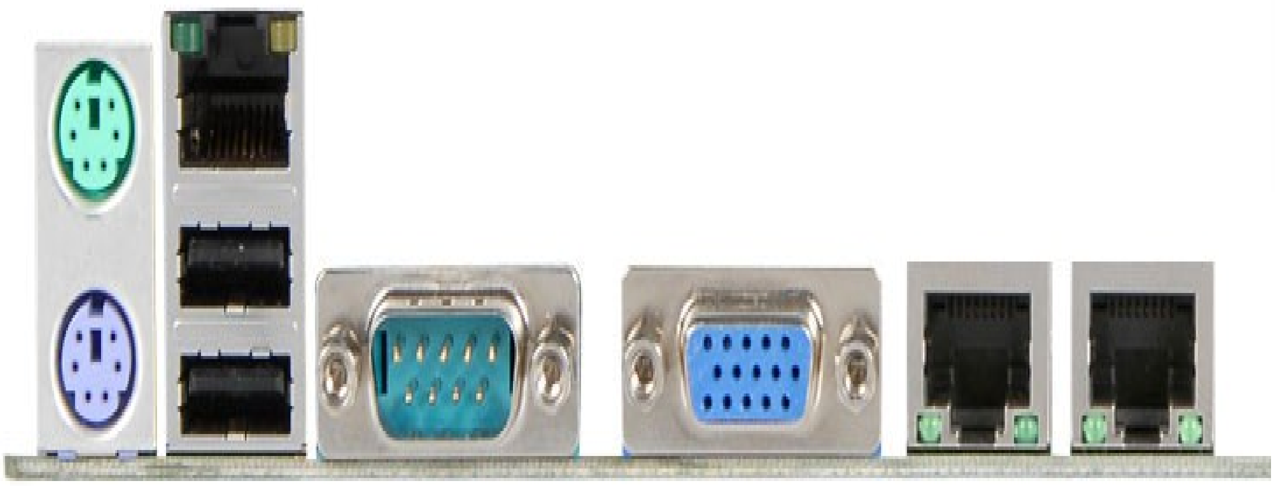
Status

Target Drive	Connected To	Read Bytes
Virtual Floppy	Not connected	n/a
Virtual CD	Not connected	n/a

Example media redirection function of Supermicro's KVM over IP interface

IPMI Connectivity

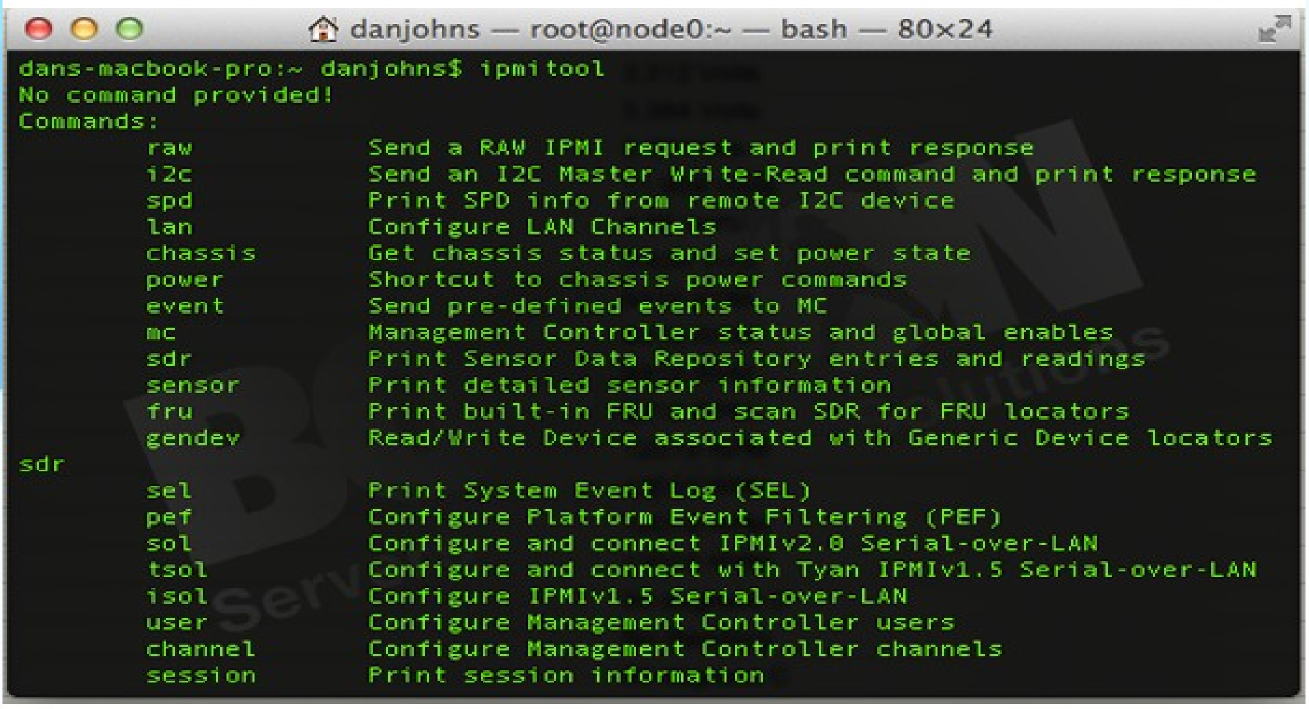
All these features are delivered remotely over a standard IP network port on the managed system. Most systems have both a dedicated port for IPMI traffic or if preferred it's possible to run this traffic over the 1st LAN port on the motherboard (eth0) alongside the systems standard IP traffic.



The dedicated IPMI LAN is above the USB ports on this Supermicro *X8STI-F motherboard*

Open IPMI Tools

For CLI control of IPMI there are several open source clients which can be used to connect to and control IPMI BMC's. The most common example of which is the Open IPMI package; this offers excellent functionality and can be easily scripted. These tools and user guides are available to download directly from sourceforge here: http://openipmi.sourceforge.net/



```
danjohns — root@node0:~ — bash — 80×24
dans-macbook-pro:~ danjohns$ ipmitool
No command provided!
Commands:
        raw           Send a RAW IPMI request and print response
        i2c           Send an I2C Master Write-Read command and print response
        spd           Print SPD info from remote I2C device
        lan           Configure LAN Channels
        chassis       Get chassis status and set power state
        power         Shortcut to chassis power commands
        event         Send pre-defined events to MC
        mc            Management Controller status and global enables
        sdr           Print Sensor Data Repository entries and readings
        sensor        Print detailed sensor information
        fru           Print built-in FRU and scan SDR for FRU locators
        gendev        Read/Write Device associated with Generic Device locators
sdr
        sel           Print System Event Log (SEL)
        pef           Configure Platform Event Filtering (PEF)
        sol           Configure and connect IPMIv2.0 Serial-over-LAN
        tsol          Configure and connect with Tyan IPMIv1.5 Serial-over-LAN
        isol          Configure IPMIv1.5 Serial-over-LAN
        user          Configure Management Controller users
        channel       Configure Management Controller channels
        session       Print session information
```

Example ipmitool output with features

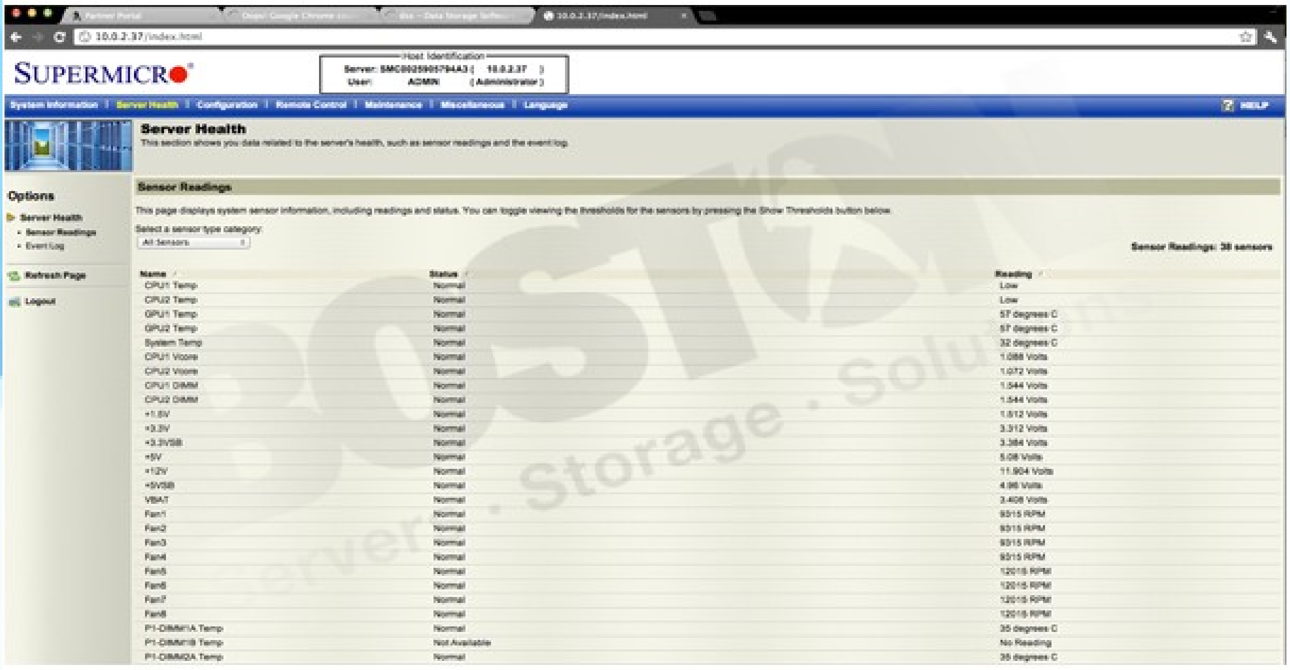
Supermicro IPMI Tool CLI

Supermicro has extended the functionality of the Open IPMI toolset with their own tool -SMCIPMITool which enables support for specific Supermicro features such as controlling their blade enclosure allowing monitoring and management over IPMI.

This tool is available for download directly from Supermicro's ftp here: ftp://ftp.supermicro.com/utility/SMCIPMItool/

Supermicro Web GUI

For casual use Supermicro provide a web GUI which is available directly on the IP address of the BMC module. This interface enables the administrator to take advantage of the full range of functions including KVM over IP and the media redirection using a Java applet without the requirement to install or load additional software.


10.0.2.37/index.html
SUPERMICRO
System Information | Server Health | Configuration | Remote Control | Maintenance | Miscellaneous | Language
HELP
Options
Server Health
Sensor Readings
Event Log
Refresh Page
Logout

Host Identification
Server: SMC0025905794A3 (10.0.2.37)
User: ADMIN (Administrator)

Server Health

This section shows you data related to the server's health, such as sensor readings and the event log.

Sensor Readings

This page displays system sensor information, including readings and status. You can toggle viewing the thresholds for the sensors by pressing the Show Thresholds button below.

Select a sensor type category: All Sensors

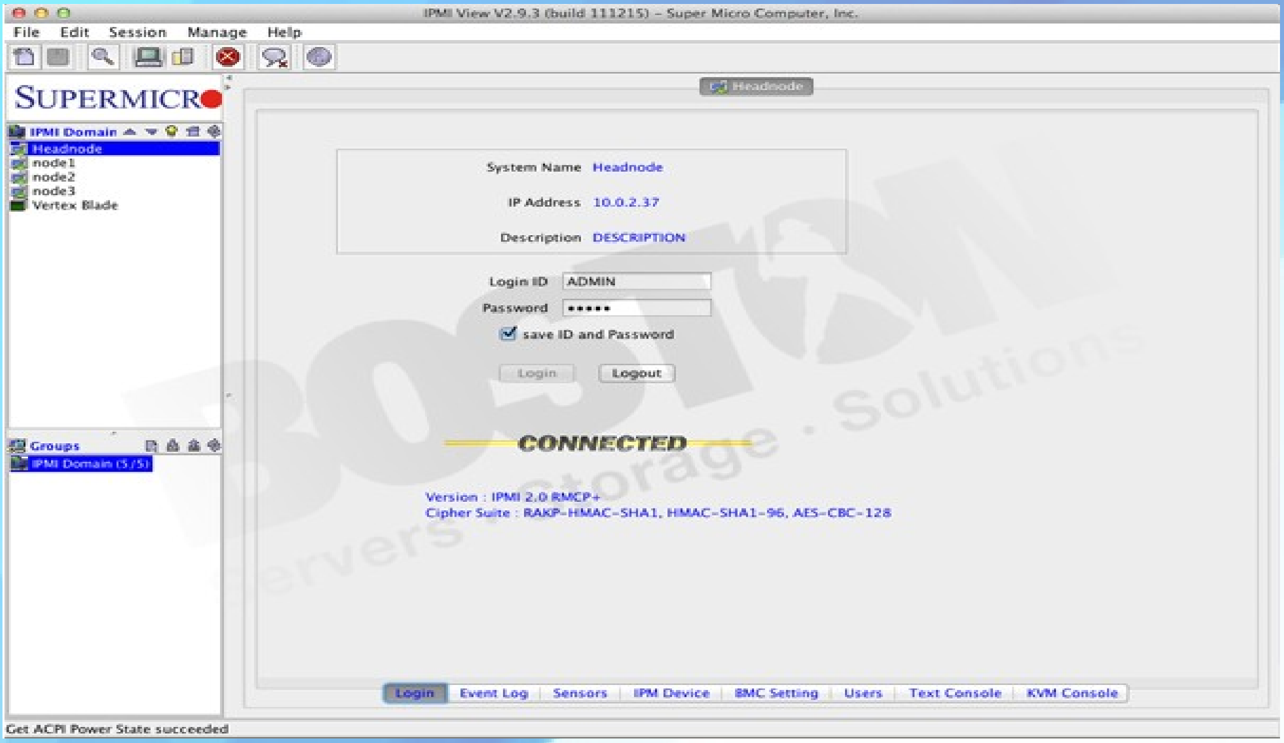
Sensor Readings: 38 sensors

Name	Status	Reading
CPU1 Temp	Normal	Low
CPU2 Temp	Normal	Low
GPU1 Temp	Normal	57 degrees C
GPU2 Temp	Normal	57 degrees C
System Temp	Normal	32 degrees C
CPU1 Vcore	Normal	1.088 Volts
CPU2 Vcore	Normal	1.072 Volts
CPU1 DIMM	Normal	1.544 Volts
CPU2 DIMM	Normal	1.544 Volts
+1.5V	Normal	1.512 Volts
+3.3V	Normal	3.312 Volts
+3.3VSB	Normal	3.384 Volts
+5V	Normal	5.08 Volts
+12V	Normal	11.904 Volts
+5VSB	Normal	4.96 Volts
VBAT	Normal	3.408 Volts
Fan1	Normal	9315 RPM
Fan2	Normal	9315 RPM
Fan3	Normal	9315 RPM
Fan4	Normal	9315 RPM
Fan5	Normal	12015 RPM
Fan6	Normal	12015 RPM
Fan7	Normal	12015 RPM
Fan8	Normal	12015 RPM
P1-DIMM1A Temp	Normal	35 degrees C
P1-DIMM1B Temp	Not Available	No Reading
P1-DIMM2A Temp	Normal	35 degrees C

Example of Supermicro's IPMI web interface showing sensor readouts

Supermicro IPMI View

For managing groups of systems Supermicro provide a Java tool called IPMI View which runs on a variety of platforms. It enables an administrator to keep track of multiple IPMI sessions and if required perform operations on groups of systems with a few simple clicks. IPMI View can be downloaded from the following location: ftp://ftp.supermicro.com/utility/IPMIView/

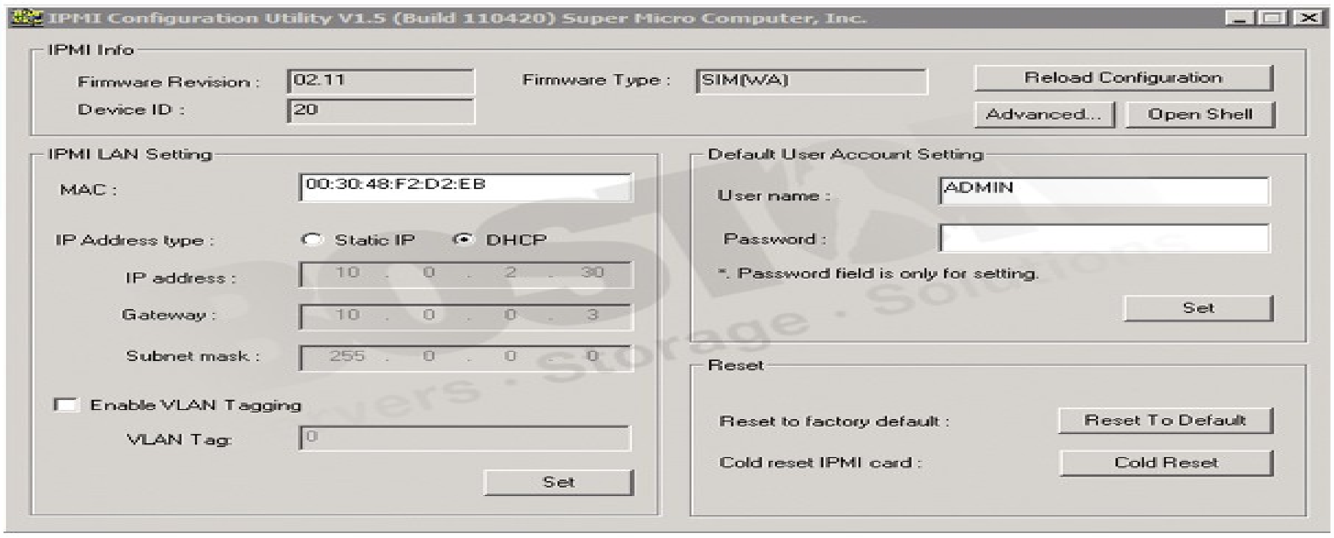


System Name Headnode
IP Address 10.0.2.37
Description DESCRIPTION
Login ID ADMIN
Password •••••
[✓] save ID and Password
CONNECTED
Version : IPMI 2.0 RMCP+
Cipher Suite : RAKP-HMAC-SHA1, HMAC-SHA1-96, AES-CBC-128
Get ACPI Power State succeeded

IPMI View Running on Mac OSX

Initial IPMI Configuration

Configuring IPMI is a simple process; the initial setup is simply to configure an IP address with which to connect to the BMC. This is done either in the motherboard BIOS under the Advanced > IPMI Configuration > LAN Configuration tab or by using the Supermicro IPMICFG tool from your O/S.


IPMI Configuration Utility V1.5 (Build 110420) Super Micro Computer, Inc.
Reload Configuration
Advanced...
Open Shell
Set
Reset To Default
Cold Reset
Set

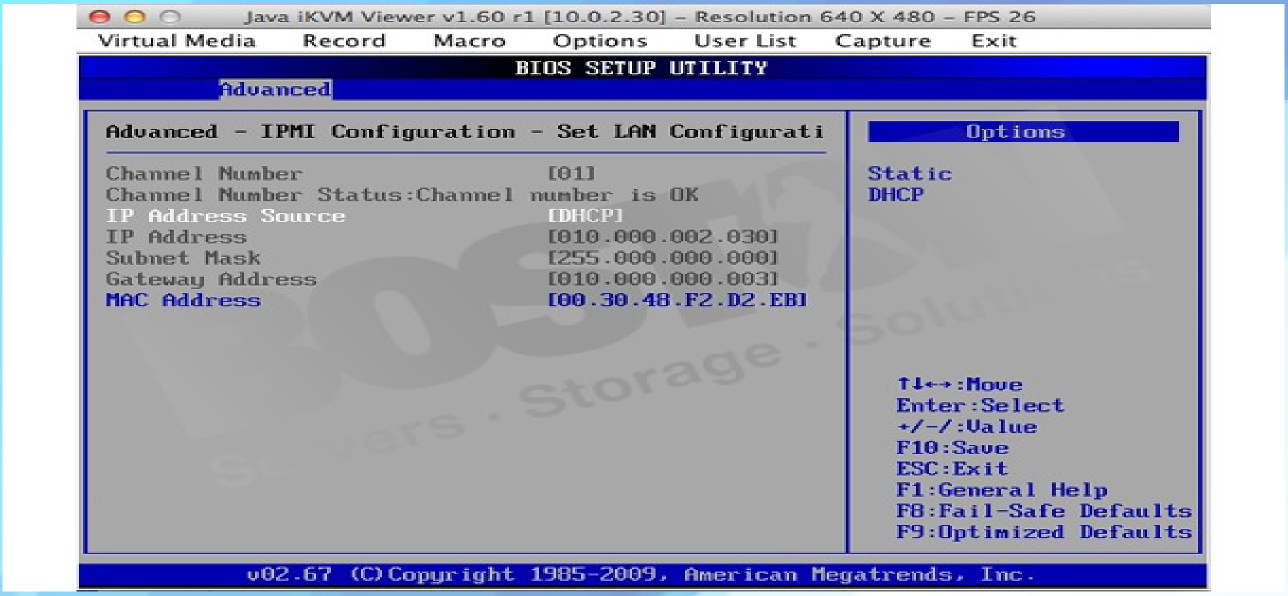
IPMI Info
Firmware Revision : 02.11
Device ID : 20
Firmware Type : SIM(WA)

IPMI LAN Setting
MAC : 00:30:48:F2:D2:EB
IP Address type : Static IP [✓ DHCP]
IP address : 10 . 0 . 2 . 30
Gateway : 10 . 0 . 0 . 3
Subnet mask : 255 . 0 . 0 . 0
☐ Enable VLAN Tagging
VLAN Tag: 0

Default User Account Setting
User name : ADMIN
Password :
*. Password field is only for setting.

Reset
Reset to factory default :
Cold reset IPMI card :

IPMICFG Running in Window



```
Java iKVM Viewer v1.60 r1 [10.0.2.30] - Resolution 640 X 480 - FPS 26
Virtual Media   Record   Macro   Options   User List   Capture   Exit
                         BIOS SETUP UTILITY
        Advanced

Advanced - IPMI Configuration - Set LAN Configurati        Options

Channel Number                         [01]                Static
Channel Number Status:Channel number is OK                 DHCP
IP Address Source                      [DHCP]
IP Address                             [010.000.002.030]
Subnet Mask                            [255.000.000.000]
Gateway Address                        [010.000.000.003]
MAC Address                            [00.30.48.F2.D2.EB]

                                                             ↑↓←→:Move
                                                             Enter:Select
                                                             +/-/:Value
                                                             F10:Save
                                                             ESC:Exit
                                                             F1:General Help
                                                             F8:Fail-Safe Defaults
                                                             F9:Optimized Defaults

        v02.67 (C)Copyright 1985-2009, American Megatrends, Inc.
```

IPMI Configuration in the motherboard BIOS using KVM/IP from Mac OSX

Security

The default username and password for Supemicro's IPMI module is ADMIN in uppercase, however this should be changed immediately in any production environment to avoid any security breaches. It's possible to use local authentication and groups for varying levels or access or even connect to an LDAP or Active Directory service for authentication.

Conclusion

IPMI is an invaluable tool for any administrator, it enables them to monitor systems on a hardware level and perform essential maintenance remotely.

Without it system installation, trouble shooting and monitoring can be a costly time consuming experience with collocated systems – a crash causing a system hang requiring a system reboot can in some situations require a trip to a data centre in another city. Simply probing the KVM to determine the fault and if necessary power cycling the node can be done in seconds resulting in a faster response time.

As all administrators know, if undiscovered, hardware failures can cause slowdown or even the complete halt of critical services. Email notifications of PSU or fan failures and changes in temperature / voltage can ensure that proactive maintenance can take place and avoid costly unscheduled downtime.

All of these factors make IPMI an excellent tool for reducing your TCO and improving your productivity as an administrator and your company's services as a whole.

Netweb
TECHNOLOGIES

Chapter 4
4. Startup and Shutdown Procedures

4.1 Shutdown Sequence

Step1.
(Make sure all jobs are killed or cancel the jobs)
# Module load utils/pdsh
#pdsh -a /etc/init.d/lustre_storage stop
(It should give /storage mount service is stop if not redo the above step)

Step2. On master
/etc/init.d/lustre_storage stop

Step3.
#pdsh -a poweroff

Step4.
#ssh mds2
#umount /lustrefs/ost0002
#umount /lustrefs/ost0003
#umount /lustrefs/ost0004
#umount /lustrefs/ost0005
#luster_rmmod
#zpool export ost0002
#zpool export ost0003
#zpool export ost0004
#zpool export ost0005
#exit

Step5. On master
#ssh mds1
#umount /lustrefs/ost0000
#umount /lustrefs/ost0001
#umount /lustre/mdt
(this will take time)
#lustre_rmmod

```
#zpool export ost0000
#zpool export ost0001
#zpool export mdt
#exit
```

Step6.

```
#ssh mds1
#poweroff (mds1)
#ssh mds2
#poweroff (mds2)
```

Step7.

```
#poweroff (master)
```

Netweb TECHNOLOGIES

4.2 Startup Sequence

Step1.
(Make sure All ethernet switch and IB switches are power on)
Power on JBOD

Step2.
Power on master,mds1 and mds2.

Step3.
Login to master as a root.

```
#systemctl start opensmd
```

Step4.

```
#ssh mds1
#zpool import mdt
#zpool import ost0000
#zpool import ost0001
#modprobe -v lustre
#mount -t lustre mdt/mdt /lustre/mdt
#mount -t lustre ost0000/ost0000 /lustre/ost0000
#mount -t lustre ost0001/ost0001 /lustre/ost0001
#exit
```

Step5.

```
#ssh mds2
#zpool import ost0002
#zpool import ost0003
#zpool import ost0004
#zpool import ost0005
#modprobe -v lustre
#mount -t lustre ost0002/ost0002 /lustre/ost0002
#mount -t lustre ost0003/ost0003 /lustre/ost0003
#mount -t lustre ost0004/ost0004 /lustre/ost0004
#mount -t lustre ost0005/ost0005 /lustre/ost0005
#exit
```

Step6. On master

```
/etc/init.d/lustre_storage start
```

Step7. Power on all compute ,GPU nodes

Step8.

```
#module load utils/pdsh
#pdsh -a /etc/init.d/lustre_storage start
```

Netweb TECHNOLOGIES

Chapter 5
5. EScalation Matrix

Service Desk Timimg : 9:00 AM to 6:00 PM (Monday to Staurday)	
HPC Support	
Level 1	Netweb Hydrabad
	Support Team (Kiran or Manoj Paul)
	04040269635 (office)
	Mr. Kiran Mob. -9703137537
	Mr. Manoj Paul Mob.- 8686038546
	supporthyd@netwebindia.com
Level 2	Peg Netweb Faridabad
	Peg Team (Mr. Rahul or Mr. C. Raja)
	0129- 2310400 (Ext . 460 or 458)
	peg@netwebindia.com
Level 3	Mr. Hemant Agrawal
	CTO
	0129- 2310400 (Ext. 422)
	hemat@netwebindia.com
Top Level	Mr. Sanjay Lodha
	CEO
	0129-2310400(Ext. 430)
	sanjay@netwebindia.com

Netweb
TECHNOLOGIES