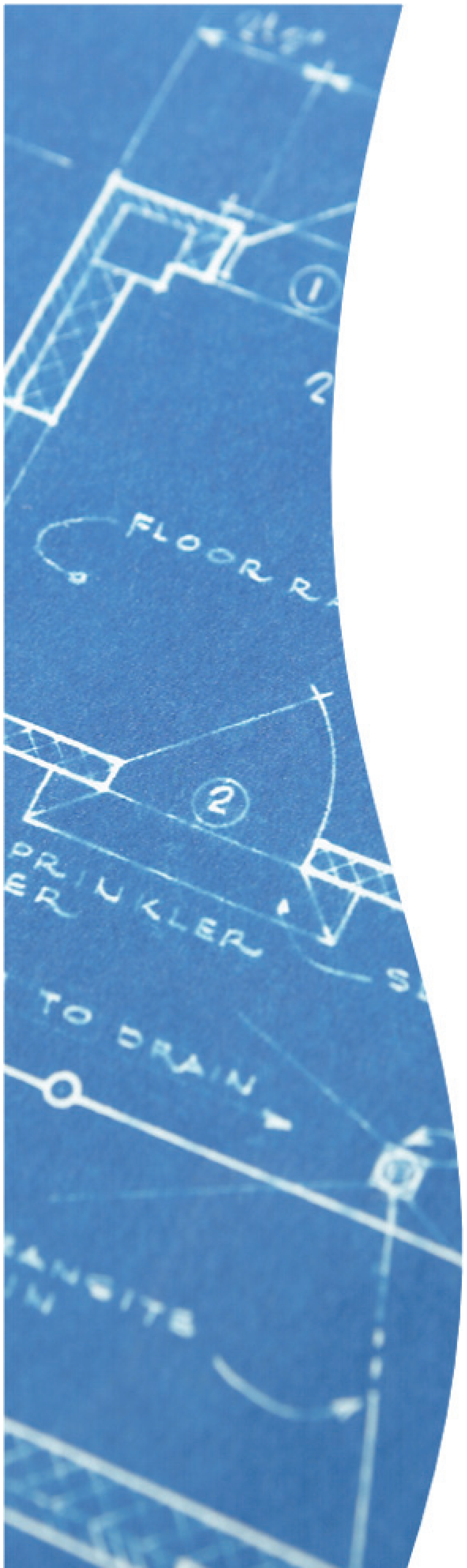

TOKYO TECH TSUBAME GRID STORAGE IMPLEMENTATION

Syuuichi Ihara, Sun Microsystems

Sun BluePrints™ On-Line — May 2007

Part No 820-2187-10
Revision 1.0, 5/22/07
Edition: May 2007

Table of Contents

Chapter 1
Introduction

One of the world's leading technical institutes, the Tokyo Institute of Technology (Tokyo Tech) recently created the fastest supercomputer in Asia, and one of the largest supercomputers outside of the United States. Deploying Sun Fire™ x64 servers and data servers in a grid architecture enabled Tokyo Tech to build a cost-effective, flexible supercomputer to meet the demands of compute- and data-intensive applications. Hundreds of systems in the grid, which Tokyo Tech named TSUBAME, incorporate thousands of processors and terabytes of memory, delivering 47.38 trillion floating-point operations per second (TeraFLOPS) of sustained LINPACK benchmark performance, and is expected to reach 100 TeraFLOPS in the future. TSUBAME also provides 1 petabyte (PB) of storage intended to support users running common off-the-shelf applications.

This Sun BluePrints™ article describes the storage architecture of the Tokyo Tech TSUBAME grid, as well as the steps for installing and configuring the Lustre file system within the storage architecture. This article is one in a series of Sun BluePrints articles on the TSUBAME grid.

Note – The TSUBAME grid, like other high performance computing environments, is constantly growing and changing. The latest configuration information and performance characteristics of the TSUBAME grid can be found on the TOP500 Supercomputer Sites Web site located at *http://www.top500.org*, or the Web site of the Global Scientific Information and Computing Center at the Tokyo Institute of Technology located at *http://www.gsic.titech.ac.jp/index.html.en/*.

This article addresses the following topics:

- "TSUBAME Architecture and Components" on page 2 briefly describes the components in the TSUBAME grid, with particular details on the Lustre file system.
- "Installing Required RPMs" on page 6 details the kernel RPMs required for Lustre and the InfiniBand network.
- "Configuring Storage and Lustre" on page 17 highlights the steps that were performed to configure the storage and Lustre file system.
- "Software RAID and Disk Monitoring" on page 31 provides a brief discussion of monitoring techniques.
- "Summary" on page 33 provides a recap of the purpose of the storage systems in the TSUBAME grid, as well as resources for more information.

Chapter 2
TSUBAME Architecture and Components

The TSUBAME supercomputer grid utilizes standard off-the-shelf hardware components to create a high performance computing (HPC) cluster solution. A single Sun Fire x64 system architecture is used across 655 servers to power three different types of clusters within the grid. All systems in the grid are interconnected through InfiniBand technology and are capable of accessing the entire storage infrastructure in parallel. The TSUBAME grid incorporates technology from ClearSpeed Technology, Cluster File Systems, Inc., and Voltaire, as well as the Sun N1™ System Manager and Sun N1 Grid Engine software. The grid is running Linux.

With a wide range of researchers throughout the university accessing the system, as well as collaborators all over the world, data storage was a key concern. Over a petabyte of physical storage capacity was required, with a Mean Time Between Failure (MTBF) for data loss across the entire system of 1000 years, and an average not ready (ANR) of less then 0.4 percent. A parallel file system with a large aggregate I/O transfer rate was needed to support over 1000 mount points, or equivalent remote file system capabilities, along with fast parallel file systems like Lustre.

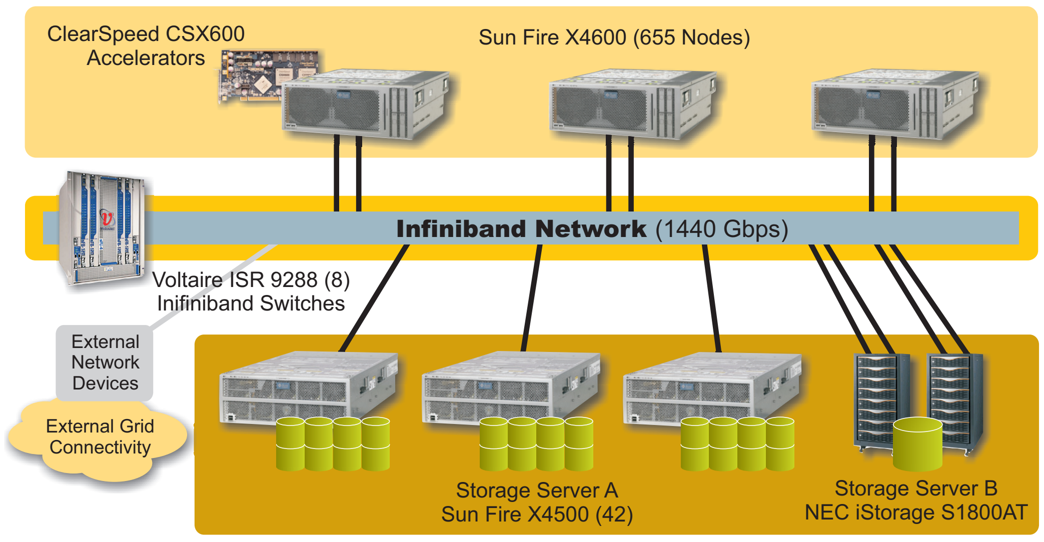

Figure 1. TSUBAME grid architecture

Compute Servers — Sun Fire X4600 Servers

In total, the TSUBAME grid consists of 655 Sun Fire X4600 servers running SUSE Linux Enterprise Server 9 SP3 configured into capacity, capability, and shared memory clusters. The clusters provide users access to 10,368 high-performance, Second Generation dual-core AMD Opteron™ processors and 21 TB of memory. Each Sun Fire X4600 server incorporates two PCI-Express 4x double data rate (DDR) InfiniBand host channel adapters (HCAs) to connect to the network. In addition, the ClearSpeed Advance accelerator board is configured into 360 of the compute servers to provide increased floating-point performance for HPC algorithms.

Data Servers — Sun Fire X4500 Servers

Forty-two Sun Fire X4500 servers running Red Hat Enterprise Linux 4 provide the storage infrastructure for the TSUBAME grid. Each high-density server houses 48 hot-swappable 500 GB SATA drives, for a total storage capacity of 24 TB or 1 PB for the entire grid. The Sun Fire X4500 servers are also configured with one dual-port PCI-X InfiniBand HCA per server.

The Sun Fire X4500 server features two dual-core AMD Opteron processors, up to 16 GB of memory, and 48 internal hot-swappable hard drives. The Sun Fire X4500 server also includes the following system architecture features:

- Embedded single-channel DDR memory controllers on each CPU. These provide maximum memory capacity and bandwidth scaling, delivering up to 16 GB of capacity and 12.8 GB/sec. of aggregated bandwidth.
- AMD Direct Connect Architecture, which directly connects CPU-to-CPU and CPU-to-I/O delivering 6.4 GB/sec. aggregate bandwidth per link. The architecture also connects CPU-to-memory using the integrated DDR controller.
- Six Marvell 88SX6081 SATA II storage controllers, connecting to 48 high-performance SATA drives.
- Two PCI-X slots to deliver high-performance I/O with over 8.5 Gb/sec. of I/O plug-in bandwidth to the InfiniBand host adapter configured in each server in the grid.
- Two Intel 82546GB Dual Port Gigabit Ethernet Controllers serving four gigabit networks ports.

Voltaire Grid Directory ISR9288

All Sun Fire X4600 and Sun Fire X4500 servers connect to an InfiniBand network through eight Voltaire Grid Director ISR9288 high-speed InfiniBand switches. Each switch provides 20 Gb/sec. bidirectional bandwidth for up to 288 InfiniBand ports in a single 14U chassis, enabling 1352 server and storage links. Up to 11.52 TB/sec. full bidirectional switch bandwidth in a fat-tree architecture is possible, with less than 420 nanoseconds of latency between any two ports. As a result, Voltaire ISR9288 switches can be interconnected to form large clusters consisting of thousands of servers.

Lustre File System

Lustre is a scalable, secure, and highly available cluster file system for Linux operating systems. It is developed and maintained by Cluster File Systems, Inc. The latest Public

Open Source version of Lustre is available from Cluster File Systems, Inc. under the GNU General Public License at: *http://www.clusterfs.com/download.html*.

The components of Lustre are the Meta Data Servers (MDS), Object Storage Servers (OSS), Object Storage Targets (OST), and the Lustre clients. The MDS keeps track of information about the files stored in Lustre. The entire namespace is stored on the MDS, while file data is stored on the OSSs. The MDS is replicated in an active/passive failover with a secondary MDS. At the file system level, Lustre treats files as objects that are located through the MDS. After the MDS identifies the storage location of a file, all subsequent file I/O occurs between the client and the OSSs. The MDS uses the locking modules and features of existing journal file systems.

An Object Storage Server is a server node with one or more network interfaces that is running the Lustre software stack. An Object Storage Target is a software interface to a single exported back-end volume, which contains file system objects rather than a whole namespace. A single OSS can export more than one OST to get around the Linux maximum partition size.

Object Storage Targets are responsible for performing I/O to and from the physical storage devices. The data for each file can be striped over multiple objects. By default, objects are randomly distributed among OSTs to provide I/O load balancing. Lustre is capable of organizing all OSSs in active-active failover pairs. However, the TSUBAME grid does not implement this functionality.

Lustre uses a networking model known as LNET, which provides support for multiple types of networks, such as Gigabit Ethernet, Infiniband, Quadrics Elan, and Myrinet GM for communication between MDS, OSSs, and clients.

The Sun Fire x4500 server is an excellent platform for the Lustre OSSs. One Sun Fire x4500 server can have six to seven OSTs (for a total of 13.5 TB to 17 TB), with each OST configured as a RAID 5 array with Linux software RAID. Figure 2 depicts a partial illustration of the TSUBAME storage architecture.

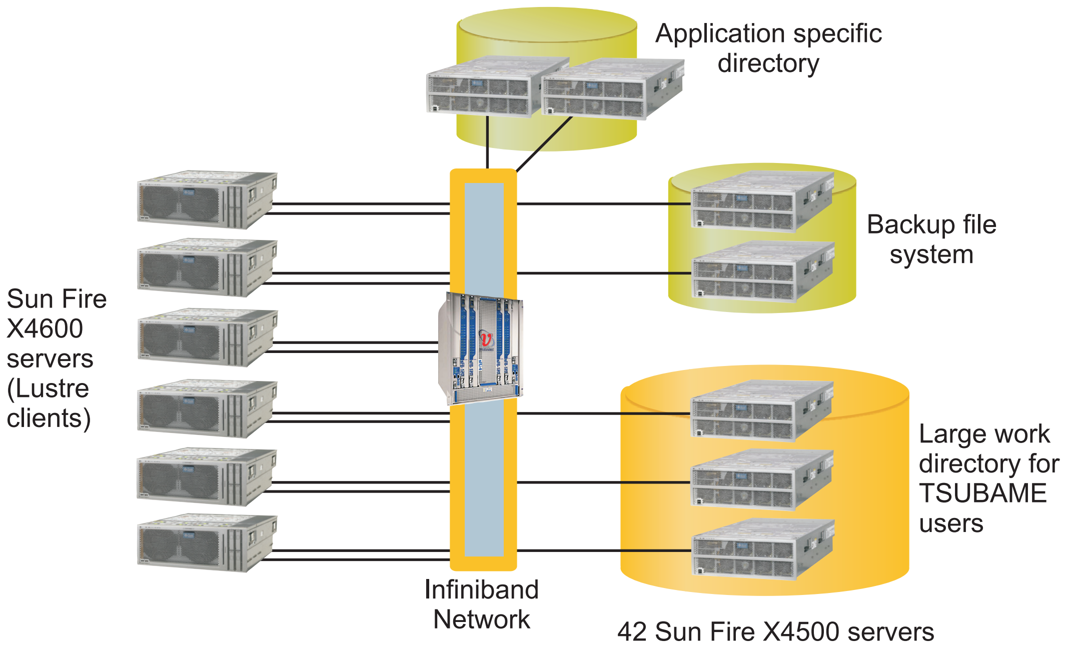
Application specific directory
Backup file system
Sun Fire X4600 servers (Lustre clients)
Large work directory for TSUBAME users
Infiniband Network
42 Sun Fire X4500 servers

Figure 2. TSUBAME storage architecture

Operating Systems

Cluster File Systems supports Lustre on Red Hat Enterprise Linux 3 and 4, SuSE Linux Enterprise Server 9 and 10, and Linux 2.4 and 2.6. Lustre currently supports TCP/IP, Quadrics Elan 3 and 4, Myrinet, and Infiniband (Voltaire, OpenIB, Cisco/Topspin, OpenFabrics, and Silverstorm).

Linux is supported on the Sun Fire X4500 server for 48-disk systems only. See the Sun Fire X4500 Server Windows and Linux Operating Systems Supplement at: *http://www.sun.com/products-n-solutions/hardware/docs/pdf/820-0642-10.pdf* for more information.

Chapter 3
Installing Required RPMs

This chapter details the necessary RPMs required to install Lustre on the Sun Fire X4500 servers and the Sun Fire X4600 servers. The software stacks for both systems are illustrated in Figure 3 and Figure 4. The components addressed in this article are outlined in black boxes.

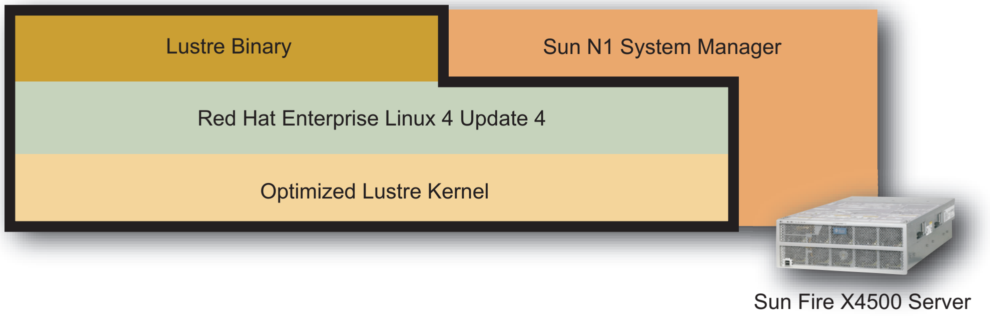

Figure 3. Sun Fire X4500 server software stack

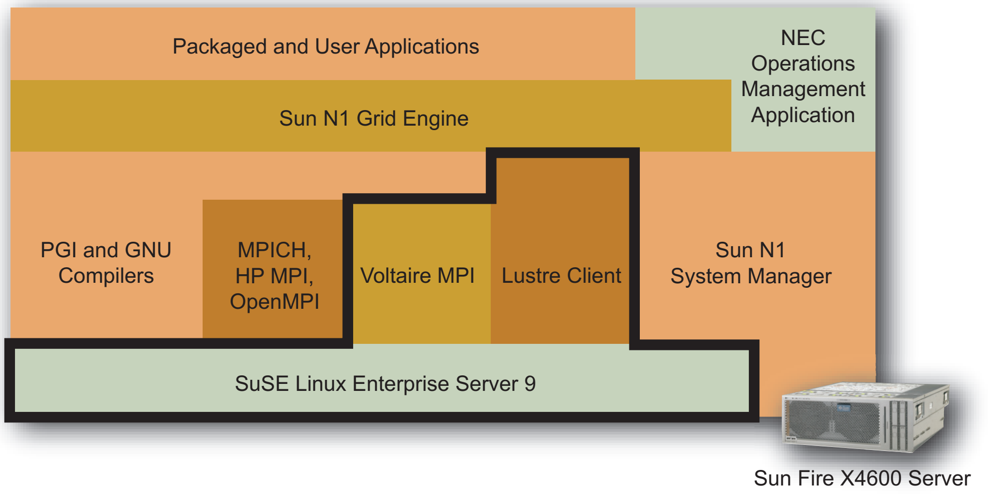

Figure 4. Sun Fire X4600 server software stack

Note – The steps in this chapter assume that Linux is already installed on both the Sun Fire X4500 and X4600 servers, i.e., Red Hat and SUSE respectively.

Required RPMs

Because Lustre is incorporated into the kernel, building and installing Lustre can be a complex task. Cluster File Systems helps reduce the effort by supplying several different pre-packaged kernel releases for common configurations. These RPMs are available for download at:
http://www.downloads.clusterfs.com/customer/public-releases/production/.
The pre-packaged releases are described below and should be installed in the following order:

- **kernel-smp-<release-version>.rpm**
  The Lustre patched Linux kernel. It should be used with matching Lustre Utilities and Modules packages.
- **kernel-source-<release-version>.rpm**
  The Lustre patched Linux kernel source RPM. It comes with the kernel package, but is not required to build or use Lustre.
- **lustre-modules-<release-version>.rpm**
  The Lustre kernel modules for the above kernel.
- **lustre-<release-version>.rpm**
  The Lustre Utilities or user space utilities for configuring and running Lustre.
- **lustre-source-<release-version>.rpm**
  The Lustre source code (including kernel patches). It is not required to build or use Lustre.

The following RPMs are required to install Lustre with a Red Hat Enterprise Linux 4 kernel on the Sun Fire X4500 servers:

- Red Hat Enterprise Linux 4 patched kernel.
  - kernel-smp-2.6.9-42.0.2.EL_lustre.1.4.7.1.x86_64.rpm
- Lustre-1.4.7.1 modules and binaries with Voltaire ibhost for Red Hat Enterprise Linux 4. These are not provided by CFS and cannot be obtained from the downloads URL listed above. Customers who need Lustre packages with Voltaire support will need to build them.
  - lustre-modules-1.4.7.1-2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
  - lustre-1.4.7.1-2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
- ibhost-3.5.5-21 (Voltaire InifinBand driver) for Red Hat Enterprise Linux 4. This driver is available from Voltaire at: *http://www.voltaire.com/SupportAndServices/Drivers/*.
  - ibhost-biz-3.5.5_21-1.k2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
- Ext2 file system utilities: e2fsprogs (*http://downloads.clusterfs.com/customer/public-releases/lfsck*).
  - e2fsprogs-1.39.cfs1-1.x86_64.rpm

Note – These binaries were the latest versions when the TSUBAME grid was installed. Sun recommends installing the most up-to-date binaries.

The patched kernel is also required to enable all of the Lustre client features, such as quota. However, full functionality was not required for the initial TSUBAME grid installation. Thus, the following RPMs are required to install the Lustre client with a SUSE Linux Enterprise Server 9 kernel on the Sun Fire X4600 servers.

- SUSE Linux Enterprise Server 9 kernel RPMs.
  - kernel-smp-2.6.5-7.282.x86_64.rpm
  - kernel-source-2.6.5-7.282.x86_64.rpm
- Lustre-1.4.7.1 modules and binaries with Voltaire ibhost for SUSE Linux Enterprise Server 9. Again, customers who need Lustre packages with Voltaire support will need to build them.
  - lustre-modules-1.4.7.1-2.6.5_7.282_smp.x86_64.rpm
  - lustre-1.4.7.1-2.6.5_7.282_smp.x86_64.rpm
- ibhost-3.5.5-21 (Voltaire InfiniBand driver) for SUSE Linux Enterprise Server 9, compiled by Voltaire. This driver is available from Voltaire at: *http://www.voltaire.com/SupportAndServices/Drivers/*.
  - ibhost-biz-3.5.5_21-1.k2.6.5_7.282_smp.x86_64.rpm

Creating a Patched Kernel on the Sun Fire X4500 Servers

It is necessary to build and install a new kernel in order to incorporate the Marvel driver for the Sun Fire X4500 server disk controllers. CFS recommends using the Quilt package to simplify the process of managing patches for kernels. Quilt is available from: *http://download.savannah.gnu.org/releases/quilt/*.

The source for the kernel, Marvell driver, Voltaire ibhost, and Lustre source are required to create the `initrd`. The necessary files are:

- kernel-2.6.9-42.0.2.EL.src.rpm
- mvSata_Linux_3_6_3.zip (available from the Sun Download Center)
- ibhost-3.5.5_21-1.src.rpm (not open source)
- quilt (*http://download.savannah.gnu.org/releases/quilt/*)
- lustre-1.4.7.1.tar.gz (*https://downloads.clusterfs.com/*)

1. Make a temporary directory, build and install a quilt RPM, then install kernel sources and extract Luster sources.

```
# mkdir /var/tmp/lustrebuild
# cd /var/tmp/lustrebuild
# tar xvfz quilt-0.29_CFS8.tar.gz
# cp quilt-0.29_CFS8.tar.gz /usr/src/redhat/SOURCES
# cd quilt-0.29_CFS8
# rpmbuild -bb quilt.spec
# rpm -ivh /usr/src/redhat/RPMS/x86_64/quilt-0.29_CFS8-1.x86_64.rpm
# cd ..
# rpm -ivh kernel-2.6.9-42.0.2.EL.src.rpm
# tar xvfz lustre-1.4.7.1.tar.gz
# cd lustre/kernel_patches/series/
```

Note – If using Lustre 1.4.7.1 as shown in these instructions, a patch is required for software RAID to fix a corruption issue. This is fixed in the latest version of Lustre. Download the `raid5-optimize-memcpy-fix.patch`, then apply it to Lustre to 1.4.7.1. See: *https://bugzilla.lustre.org/show bug.cgi?id=11313/*.

2. Lustre provides a patch series for several kernels. The 2.6-rhel4-titech.series is suitable for Sun Fire X4500 servers with Linux software RAID. It includes the following software RAID 5 patches.

```
# cat 2.6-rhel4-titech.series | grep raid5
raid5-stats.patch
raid5-configurable-cachesize.patch
raid5-large-io.patch
raid5-stripe-by-stripe-handling.patch
raid5-optimize-memcpy.patch
raid5-merge-ios.patch
raid5-serialize-ovelapping-reqs.patch
```

3. Remove *compile-fixes-2.6.9-rhel4-22.patch* from `2.6-rhel4-titech.series`, because it is not necessary on kernel-2.6.9-42.0.2. This patch is removed in recent Lustre versions.

```
# cp 2.6-rhel4-titech.series 2.6-rhel4-titech.series.orig
# cat 2.6-rhel4-titech.series.orig | sed -e '/^compile-fixes-2.6.9/d' >
2.6-rhel4-titech.series
# cd /var/tmp/lustrebuild
# cp /usr/src/redhat/SPECS/kernel-2.6.spec .
# patch -p0 < patched-kernel-2.6_spec.patch
```

4. Modify the following points before building the patched kernel.

```
# vi patched-kernel-2.6_spec.patch
# diff -Nu kernel-2.6.spec patched-kernel-2.6_spec.patch
--- kernel-2.6.spec.orig2007-02-08 20:17:19.000000000 +0900
+++ kernel-2.6.spec2007-02-08 21:47:32.000000000 +0900
@@ -3,13 +3,13 @@
 # What parts do we want to build? We must build at least one kernel.
 # These are the kernels that are built IF the architecture allows it.

-%define buildup 1
+%define buildup 0
 %define buildsmp 1
 %define buildsource 0
-%define buildhugemem 1
-%define buildlargesmp 1
+%define buildhugemem 0
+%define buildlargesmp 0
 %define builddoc 0
-%define kabi 1
+%define kabi 0

 %define FC2 0
 %define FC3 0
@@ -22,7 +22,7 @@
 # that the kernel isn't the stock distribution kernel, for example by
 # adding some text to the end of the version number.
 #
-%define release 42.0.2.EL
+%define release 42.0.2.EL_lustre.1.4.7.1
 %define sublevel 9
 %define kversion 2.6.%{sublevel}
 %define rpmversion 2.6.%{sublevel}
@@ -3276,7 +3276,9 @@
 cd linux-%{kversion}


-
+ln -s /var/tmp/lustrebuild/lustre-1.4.7.1/lustre/kernel_patches/patches/ .
+ln -s /var/tmp/lustrebuild/lustre-1.4.7.1/lustre/kernel_patches/series/
2.6-rhel4-titech.series ./series
+quilt push -a -v
 BuildKernel() {

     # Pick the right config file for the kernel we're building
@@ -3299,7 +3301,7 @@
     make -s mrproper
     cp configs/$Config .config

-    make -s nonint_oldconfig > /dev/null
+    make menuconfig
     make -s include/linux/version.h

     make -s %{?_smp_mflags} %{make_target}
```

5. The preparation for the build is completed. Now run `rpmbuild`.

```
# rpmbuild -bb kernel-2.6.spec
```

After running the rpmbuild, the Linux kernel configuration CLI tool can be used. The kernel configuration needs to be modified for the Voltaire IB driver. Red Hat Enterprise Linux 4 includes Infiniband drivers, but they are based on OpenIB. TSUBAME grid uses Voltaire IB, so the OpenIB driver is not necessary.

6. Disable OpenIB modules.

```
Device Drivers
 ---> InfiniBand support
       ---> InfiniBand support
              InfiniBand support
              PathScale InfiniPath Driver
```

The file `kernel-smp-2.6.9 42.0.2.EL_lustre.1.4.7.1.x86_64.rpm` should be visible in the `/usr/src/redhat/RPMS/x86_64` file after completing the `rpmbuild` command.

7. Build Lustre binaries with Voltaire for Red Hat.

```
# cd /usr/src
# rm -f linux linux-2.6
# ln -s /usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9 linux
# ln -s linux linux-2.6
# cd /var/tmp/lustrebuild
# rpm -ivh ibhost-3.5.5_21-1.src.rpm
# tar xvfj /usr/src/redhat/SOURCES/ibhost-3.5.5_21-1.src.tar.bz2
# cd lustre-1.4.7.1
# ./configure --with-vib=/var/tmp/lustrebuild/ibhost-3.5.5_21
# cd build
# cp lustre.spec lustre.spec.orig
# cat lustre.spec.orig  | sed -e 's/2.6.9_42.0.2.EL_lustre.1.4.7.1smp_[0-
9]*/2.6.9_42.0.2.EL_lustre.1.4.7.1smp/g' > lustre.spec
# cd ..
# make rpms
```

The binaries are built in `/usr/src/redhat/RPMS/x86_64.`

```
/usr/src/redhat/RPMS/x86_64/lustre-1.4.7.1-
2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
/usr/src/redhat/RPMS/x86_64/lustre-modules-1.4.7.1-
2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
/usr/src/redhat/RPMS/x86_64/lustre-debuginfo-1.4.7.1-
2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
/usr/src/redhat/RPMS/x86_64/lustre-source-1.4.7.1-
2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
```

Installing Lustre Related RPMs on Red Hat Enterprise Linux 4

1. For Red Hat Enterprise Linux 4 use the `rpm` command to install Lustre RPMs after Red Hat is installed.

```
# rpm -ivh kernel-smp-2.6.9-42.0.2.EL_lustre.1.4.7.1.x86_64.rpm
# rpm -ivh lustre-modules-1.4.7.1-
2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
# rpm -ivh lustre-1.4.7.1-2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
# rpm -e --nodeps e2fsprogs-1.35-12.4.EL4
# rpm -ivh e2fsprogs-1.39.cfs1-1.x86_64.rpm
```

`/boot/grub/grub.conf` is updated for the patched kernel, but the default kernel is still the previous kernel. It is possible to change the default parameter in `grub.conf` using the `grub`(8) utility.

2. If using a Voltaire Infiniband network, add kernel module options into `/etc/modprobe.conf`.

```
options lnet networks=vib
```

Modifying and Installing the Marvell Driver for the Patched Kernel

The mv_sata driver enables the *Write Cache* (write back mode) feature by default. Write Cache helps write performance. Because the Sun Fire X4500 server can potentially power-off as a result of hardware or software errors, Sun recommends disabling (write through mode) this feature on the Sun Fire X4500 server for availability.

With write cache off, the file system can be recovered automatically from the journal the next time it is mounted. In the write back mode, a write operation is finished when the CPU writes to memory. The data is not stored in disk completely, which can result in some risk of data corruption if the system is suddenly powered off.

1. Disable the write cache on the Marvell driver.

```
# cd /var/tmp/lustrebuild
# tar xvfz mvSatalinux-3.6.3_2-src.tar.gz
# patch -p0 < mv_sata-always_write_through.patch
# cd mvSata_Linux_3_6_3/LinuxIAL
# KERNEL_SRC=/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9 make
EXTERNAL_CFLAGS=-DNO_WCACHE
```

2. Update `initrd` with the patched `mv_sata`.

```
# cp mv_sata.ko /lib/modules/2.6.9-42.0.2.EL_lustre.1.4.7.1smp/kernel/
drivers/scsi/
# mkinitrd -f --with=mv_sata --with=raid1 /boot/initrd-2.6.9-
42.0.2.EL_lustre.1.4.7.1smp.img 2.6.9-42.0.2.EL_lustre.1.4.7.1smp
```

3. After the reboot, check to see that the drive cache mode has changed to write through.

```
# dmesg | grep SCSI
SCSI subsystem initialized
scsi0 : Marvell SCSI to SATA adapter
scsi1 : Marvell SCSI to SATA adapter
scsi2 : Marvell SCSI to SATA adapter
scsi3 : Marvell SCSI to SATA adapter
scsi4 : Marvell SCSI to SATA adapter
scsi5 : Marvell SCSI to SATA adapter
scsi6 : Marvell SCSI to SATA adapter
scsi7 : Marvell SCSI to SATA adapter
  Type:   Direct-Access                      ANSI SCSI revision: 03
SCSI device sda: 976773168 512-byte hdwr sectors (500108 MB)
SCSI device sda: drive cache: write through
SCSI device sda: 976773168 512-byte hdwr sectors (500108 MB)
SCSI device sda: drive cache: write through
  Type:   Direct-Access                      ANSI SCSI revision: 03
SCSI device sdb: 976773168 512-byte hdwr sectors (500108 MB)
SCSI device sdb: drive cache: write through
SCSI device sdb: 976773168 512-byte hdwr sectors (500108 MB)
SCSI device sdb: drive cache: write through
Type:   Direct-Access                      ANSI SCSI revision: 03
SCSI device sdc: 976773168 512-byte hdwr sectors (500108 MB)
SCSI device sdc: drive cache: write through
```

A reboot is required to build the IB driver.

4. Create and install the Voltaire IB driver for the patched kernel.

```
# uname -r
2.6.9-42.0.2.EL_lustre.1.4.7.1smp
# cd /usr/src
# rm -f linux linux-2.6
# ln -s /usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9 linux
# ln -s /usr/src/linux linux-2.6
# ln -s /usr/src/linux /lib/modules/2.6.9-42.0.2.EL_lustre.1.4.7.1smp/build
# cd /var/tmp/lustrebuild
# rpmbuild --rebuild ibhost-3.5.5_21-1.src.rpm
```

The binaries are built in:

```
/usr/src/redhat/RPMS/x86_64/ibhost-biz-3.5.5_21-
1.k2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
/usr/src/redhat/RPMS/x86_64/ibhost-hpc-3.5.5_21-
1.k2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm

# rpm -ivh ibhost-biz-3.5.5_21-
1.k2.6.9_42.0.2.EL_lustre.1.4.7.1smp.x86_64.rpm
# /etc/init.d/voltaireibhost start
# vstat
1 HCA found:
        hca_id=InfiniHost0
        pci_location={BUS=0x0E,DEV/FUNC=0x00}
        vendor_id=0x02C9
        vendor_part_id=0x5A44
        hw_ver=0xA1
        fw_ver=3.2.0
        PSID=
        num_phys_ports=2
                port=1
                port_state=PORT_ACTIVE
                sm_lid=0x0001
                port_lid=0x0053
                port_lmc=0x00
                max_mtu=2048
                port=2
                port_state=PORT_DOWN
                sm_lid=0x0000
                port_lid=0x0000
                port_lmc=0x00
                max_mtu=2048
```

Installing Lustre Client-Related RPMs on SUSE Linux Enterprise Server 9

CFS provides a SUSE Linux Enterprise Server 9 patched kernel. SUSE Linux Enterprise Server 9 SP3's kernel is also patched for Lustre, but has a critical issue for Lustre that is fixed on 2.6.5-7.267 and later.

1. For SUSE Linux Enterprise Server 9 use the `rpm` command to install Lustre RPMs after SUSE Linux Enterprise Server 9 is installed.

```
# rpm -ivh kernel-smp-2.6.5-7.276.x86_64.rpm
# rpm -ivh kernel-source-2.6.5-7.276.x86_64.rpm
```

2. Lustre is compiled with Voltaire for the client. The steps are the same as for Red Hat. It is not necessary to compile all of the modules. The process can be interrupted after a few seconds with `ctrl-c`.

```
# mkdir /var/tmp/lustrebuild
# cd /usr/src
# rm linux linux-2.6
# ln -s linux-2.6.5-7.276 linux
# ln -s linux-2.6.5-7.276 linux-2.6
# cd linux
# make
```

3. Continue with these steps.

```
# cd /var/tmp/lustrebuild
# rpm -ivh ibhost-3.5.5_21-1.src.rpm
# tar xvfj /usr/src/packages/SOURCES/ibhost-3.5.5_21-1.src.tar.bz2
# tar xvfz lustre-1.4.7.1.tar.gz
# cd lustre-1.4.7.1
# ./configure --with-vib=/var/tmp/lustrebuild/ibhost-3.5.5_21
# cd build
# cp lustre.spec lustre.spec.orig
# cat lustre.spec.orig | sed -e 's/2.6.5_7.276_smp_[0-9]*/2.6.5_7.276_smp/
g' > lustre.spec
# cd ..
# make rpms

The binaries are built in:
/usr/src/packages/RPMS/x86_64/lustre-1.4.7.1-2.6.5_7.276_smp.x86_64.rpm
/usr/src/packages/RPMS/x86_64/lustre-modules-1.4.7.1-
2.6.5_7.276_smp.x86_64.rpm
/usr/src/packages/RPMS/x86_64/lustre-debuginfo-1.4.7.1-
2.6.5_7.276_smp.x86_64.rpm
/usr/src/packages/RPMS/x86_64/lustre-source-1.4.7.1-
2.6.5_7.276_smp.x86_64.rpm

# rpm -ivh --force lustre-modules-1.4.7.1-2.6.5_7.276_smp.x86_64.rpm
# rpm -ivh lustre-1.4.7.1-2.6.5_7.276_smp.x86_64.rpm
# rpm -ivh ibhost-biz-3.5.5_21-1sles9.k2.6.5_7.276_smp.x86_64.rpm
```

4. If using a Voltaire Infiniband network and PCI-Express Mem free HCA, add the following kernel module options into `/etc/modprobe.conf`.

```
options lnet networks=vib
options kviblnd hca_basename=InfiniHost_III_Lx
options kviblnd fmr_remaps=250
```

Chapter 4
Configuring Storage and Lustre

This chapter provides details on configuring the Sun Fire X4500 server disks and RAID arrays, as well as the Lustre OSS, MDT, and clients.

Sun Fire x4500 Disk Management

The Sun Fire X4500 system includes six Marvell controllers. Balancing the I/O across controllers is key in achieving performance for the Lustre file system. In a traditional Linux system the device nodes in the `/dev` directory are a static set of files, which can be confusing when managing 48 drives. Tokyo Tech wanted to use physical and logical device maps similar to the method used in the Solaris OS. The Linux `udev`(2) utility is helpful in this configuration because it enables configurable dynamic device naming in the `/dev` directory and provides the ability to have persistent device names. `udev` dynamically provides only the nodes for the devices actually existing on the system. The Linux device mapping for the Sun Fire X4500 servers is depicted in Figure 5.

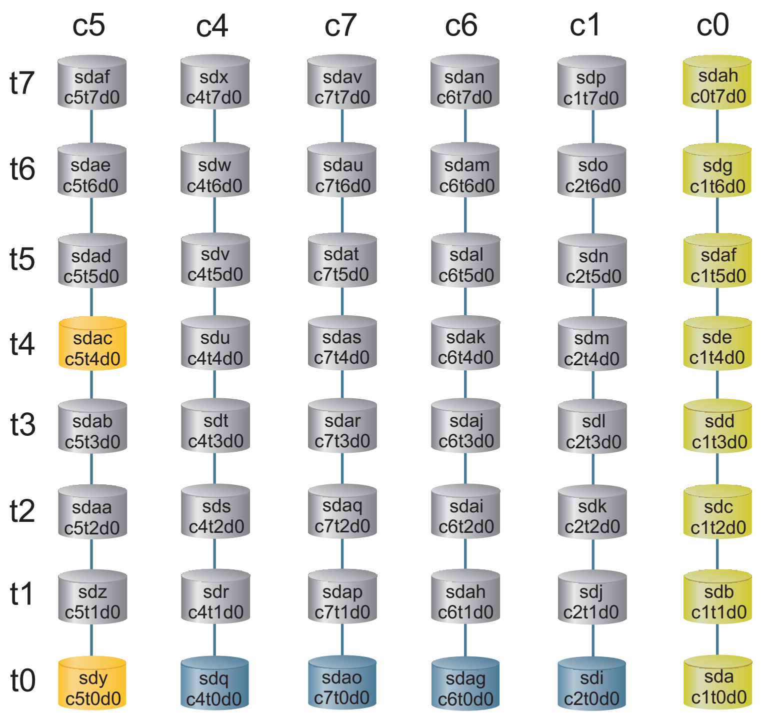

Figure 5. Linux device mapping

`udev` provides persistent naming for some device types by default. For other devices, such as the disks in the Sun Fire X4500 server, it is necessary to write custom rules in order to create and name `/dev` device nodes corresponding to the hardware.

1. Create a `/etc/udev/rules.d/99-dsk.rules` file to define the disks in the Sun Fire X4500 server.

```
#
# /etc/udev/rules.d/99-dsk.rules
#
BUS="scsi", ID="0:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t0d0"
BUS="scsi", ID="1:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t1d0"
BUS="scsi", ID="2:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t2d0"
BUS="scsi", ID="3:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t3d0"
BUS="scsi", ID="4:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t4d0"
BUS="scsi", ID="5:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t5d0"
BUS="scsi", ID="6:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t6d0"
BUS="scsi", ID="7:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c0t7d0"

BUS="scsi", ID="8:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t0d0"
BUS="scsi", ID="9:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t1d0"
BUS="scsi", ID="10:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t2d0"
BUS="scsi", ID="11:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t3d0"
BUS="scsi", ID="12:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t4d0"
BUS="scsi", ID="13:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t5d0"
BUS="scsi", ID="14:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t6d0"
BUS="scsi", ID="15:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c1t7d0"

BUS="scsi", ID="16:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t0d0"
BUS="scsi", ID="17:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t1d0"
BUS="scsi", ID="18:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t2d0"
BUS="scsi", ID="19:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t3d0"
BUS="scsi", ID="20:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t4d0"
BUS="scsi", ID="21:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t5d0"
BUS="scsi", ID="22:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t6d0"
BUS="scsi", ID="23:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c4t7d0"

BUS="scsi", ID="24:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t0d0"
BUS="scsi", ID="25:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t1d0"
BUS="scsi", ID="26:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t2d0"
BUS="scsi", ID="27:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t3d0"
BUS="scsi", ID="28:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t4d0"
BUS="scsi", ID="29:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t5d0"
BUS="scsi", ID="30:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t6d0"
BUS="scsi", ID="31:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c5t7d0"

BUS="scsi", ID="32:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t0d0"
BUS="scsi", ID="33:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t1d0"
BUS="scsi", ID="34:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t2d0"
BUS="scsi", ID="35:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t3d0"
BUS="scsi", ID="36:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t4d0"
BUS="scsi", ID="37:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t5d0"
BUS="scsi", ID="38:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t6d0"
BUS="scsi", ID="39:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c6t7d0"

BUS="scsi", ID="40:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t0d0"
BUS="scsi", ID="41:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t1d0"
BUS="scsi", ID="42:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t2d0"
BUS="scsi", ID="43:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t3d0"
BUS="scsi", ID="44:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t4d0"
BUS="scsi", ID="45:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t5d0"
BUS="scsi", ID="46:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t6d0"
BUS="scsi", ID="47:0:0:0", KERNEL="sd*[a-z]",  SYMLINK="dsk/c7t7d0"
```

```
BUS="scsi", ID="0:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t0d0s%c"
BUS="scsi", ID="1:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t1d0s%c"
BUS="scsi", ID="2:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t2d0s%c"
BUS="scsi", ID="3:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t3d0s%c"
BUS="scsi", ID="4:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t4d0s%c"
BUS="scsi", ID="5:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t5d0s%c"
BUS="scsi", ID="6:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t6d0s%c"
BUS="scsi", ID="7:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c0t7d0s%c"

BUS="scsi", ID="8:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t0d0s%c"
BUS="scsi", ID="9:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t1d0s%c"
BUS="scsi", ID="10:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t2d0s%c"
BUS="scsi", ID="11:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t3d0s%c"
BUS="scsi", ID="12:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t4d0s%c"
BUS="scsi", ID="13:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t5d0s%c"
BUS="scsi", ID="14:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t6d0s%c"
BUS="scsi", ID="15:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c1t7d0s%c"

BUS="scsi", ID="16:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t0d0s%c"
BUS="scsi", ID="17:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t1d0s%c"
BUS="scsi", ID="18:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t2d0s%c"
BUS="scsi", ID="19:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t3d0s%c"
BUS="scsi", ID="20:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t4d0s%c"
BUS="scsi", ID="21:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t5d0s%c"
BUS="scsi", ID="22:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t6d0s%c"
BUS="scsi", ID="23:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c4t7d0s%c"

BUS="scsi", ID="24:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t0d0s%c"
BUS="scsi", ID="25:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t1d0s%c"
BUS="scsi", ID="26:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t2d0s%c"
```

```
BUS="scsi", ID="27:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t3d0s%c"
BUS="scsi", ID="28:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t4d0s%c"
BUS="scsi", ID="29:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t5d0s%c"
BUS="scsi", ID="30:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t6d0s%c"
BUS="scsi", ID="31:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c5t7d0s%c"

BUS="scsi", ID="32:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t0d0s%c"
BUS="scsi", ID="33:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t1d0s%c"
BUS="scsi", ID="34:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t2d0s%c"
BUS="scsi", ID="35:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t3d0s%c"
BUS="scsi", ID="36:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t4d0s%c"
BUS="scsi", ID="37:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t5d0s%c"
BUS="scsi", ID="38:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t6d0s%c"
BUS="scsi", ID="39:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c6t7d0s%c"
BUS="scsi", ID="40:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t0d0s%c"
BUS="scsi", ID="41:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t1d0s%c"
BUS="scsi", ID="42:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t2d0s%c"

BUS="scsi", ID="43:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t3d0s%c"
BUS="scsi", ID="44:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t4d0s%c"
BUS="scsi", ID="45:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t5d0s%c"
BUS="scsi", ID="46:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t6d0s%c"
BUS="scsi", ID="47:0:0:0", KERNEL="sd*[a-z][0-9]*",  PROGRAM="/etc/udev/
scripts/minus-1.sh %n", SYMLINK="dsk/c7t7d0s%c"
```

2. Linux partition numbers start from the number one, while Solaris partition numbering starts from zero. Create the `/etc/udev/scripts/minus-1.sh` script to make naming consistent. The script is employed by `udev`.

```
#!/bin/sh
a=`expr $1 - 1`
echo $a
```

Having persistently named device nodes has several advantages. In a traditional Linux system using `devfs`, `/dev` is typically populated in the order in which devices are originally connected to the system. If a device fails or is removed, this causes the names of the remaining devices to change when the system is rebooted, which can be confusing as well as disastrous. With persistent naming, devices retain their names across reboots.

Configuring the Object Storage Server (OSS)

Sun Fire X4500 servers do not include a hardware RAID controller. The TSUBAME grid uses RAID capabilities provided by Linux software (RAID 0, 1, and 5). The Sun Fire X4500 server is utilized for the OSSs and the OSTs. RAID 5 is used for OSTs. RAID 6 is desirable, but not ready at the time of the installation. RAID 1 is used for the external journal. Figure 6 illustrates an example RAID configuration for the TSUBAME grid. If more disk space is required, use the `lvm`(8) command to configured the mirrored journal devices with boot disks

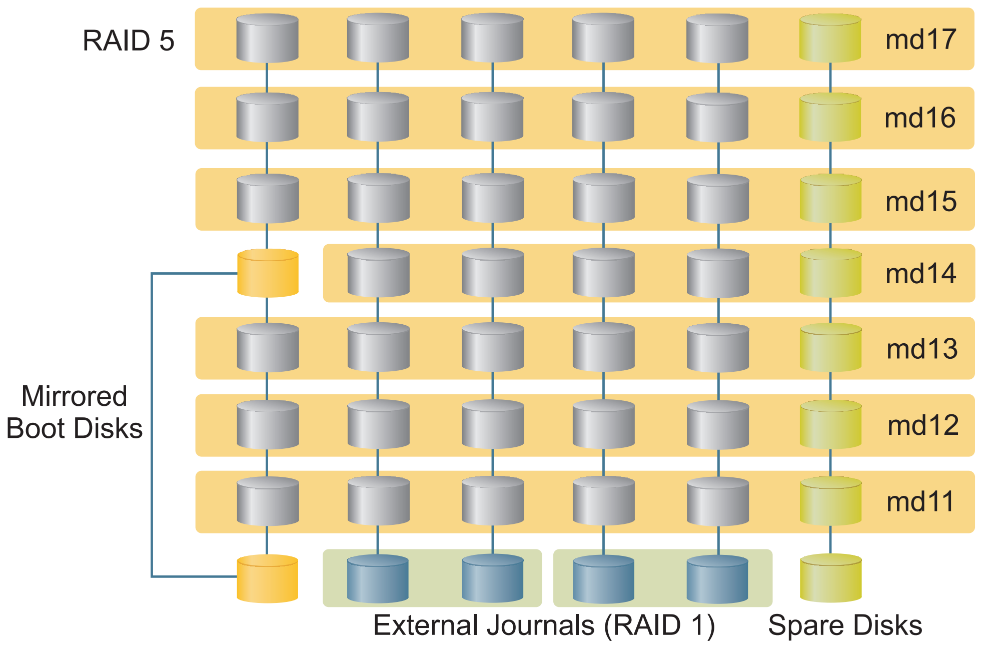

Figure 6. RAID configuration

Changing the I/O Scheduler

Linux supports three I/O schedulers — CFS, noop, and deadline. The deadline scheduler improves performance on the Sun Fire X4500 server when using Linux software RAID. The I/O scheduler can be configured for deadline by adding *elevator=dealine* as a boot parameter in the `/boot/grub/grub.conf` file.

Creating a RAID-1 Array for the External Journal

Each journal device requires about 500 MB of space. The following steps create this space.

1. This script creates four partitions with 500 MB on four drives. It then creates the mirrored devices with paired partitions.

```
# cat mk-jdev_oss.sh
#!/bin/sh
for i in 1 4 6 7; do
sfdisk -uM /dev/dsk/c${i}t0d0 << EOF
,500,L
,500,L
,500,L
,500,L
EOF
done
echo "yes" | mdadm -C /dev/md21 -l 1 -n 2 /dev/dsk/c{4,7}t0d0s0
echo "yes" | mdadm -C /dev/md22 -l 1 -n 2 /dev/dsk/c{4,7}t0d0s1
echo "yes" | mdadm -C /dev/md23 -l 1 -n 2 /dev/dsk/c{4,7}t0d0s2
echo "yes" | mdadm -C /dev/md24 -l 1 -n 2 /dev/dsk/c{4,7}t0d0s3
echo "yes" | mdadm -C /dev/md25 -l 1 -n 2 /dev/dsk/c{1,6}t0d0s0
echo "yes" | mdadm -C /dev/md26 -l 1 -n 2 /dev/dsk/c{1,6}t0d0s1
echo "yes" | mdadm -C /dev/md27 -l 1 -n 2 /dev/dsk/c{1,6}t0d0s2
#
```

2. Format all of the external journal devices using `mke2fs -O journal_dev`.

```
#!/bin/sh


md21 md22 md23 md24 md25 md26 md27"

mke2fs -V 2>&1 | grep cfs > /dev/null
if [ $? -ne 0 ]; then
echo "Please migrate to e2fsprogs-1.39.cfs1-1 before $0"
exit 1
fi

for dev in ${JOURNAL_DEVS}; do
mke2fs -O journal_dev -b 4096 /dev/${dev}
done
```

3. Run `/proc/mdstat` to check the configuration.

```
# /proc/mdstat
Personalities : [raid1] [raid5]
md27 : active raid1 sdag3[1] sdi3[0]
      530048 blocks [2/2] [UU]
        resync=DELAYED
md26 : active raid1 sdag2[1] sdi2[0]
      530048 blocks [2/2] [UU]
        resync=DELAYED
md25 : active raid1 sdag1[1] sdi1[0]
      530048 blocks [2/2] [UU]
      [==========>.........]  resync = 54.4% (289152/530048) finish=0.1min
speed=36144K/sec
md24 : active raid1 sdao4[1] sdq4[0]
      530048 blocks [2/2] [UU]
        resync=DELAYED
md23 : active raid1 sdao3[1] sdq3[0]
      530048 blocks [2/2] [UU]
        resync=DELAYED
md22 : active raid1 sdao2[1] sdq2[0]
      530048 blocks [2/2] [UU]
        resync=DELAYED
md21 : active raid1 sdao1[1] sdq1[0]
```

Making Journal Devices

Luster provides device drivers that support Lustre within journaling Linux file systems. This capability increases the reliability and recoverability of Lustre by leveraging the journaling mechanism existing in these file systems. By default, ext3 journaling occurs within the same drive. However, it also supports an external journal device, which provides better performance and availability.

1. Create the external journal device.

```
# cat format-jdev.sh
#!/bin/sh
JOURNAL_DEVS="md21 md22 md23 md24 md25 md26 md27"
mke2fs -V 2>&1 | grep cfs > /dev/null
if [ $? -ne 0 ]; then
echo "Please migrate to e2fsprogs-1.39.cfs1-1 before $0"
exit 1
fi
for dev in ${JOURNAL_DEVS}; do
mke2fs -O journal_dev -b 4096 /dev/${dev}
done
```

2. Run the `format-jdev.sh` script.

```
# ./format-jdev.sh
```

Creating RAID-5 Devices for the File Systems

This RAID configuration consists of four disks for data, one disk for parity, and one disk for spare for a total of 2 TB. The `mkraid5_oss.sh` script creates 256K chunks for a full stripe request of 1 MB.

1. Create the `mkraid5_oss.sh` script, then run the script.

```
#!/bin/sh
CHUNK=256

echo "yes" | mdadm -C /dev/md11 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t1d0
echo "yes" | mdadm -C /dev/md12 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t2d0
echo "yes" | mdadm -C /dev/md13 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t3d0
echo "yes" | mdadm -C /dev/md14 -c ${CHUNK} -l 5 -n 4 -x 1 /dev/dsk/
c{0,1,4,6,7}t4d0
echo "yes" | mdadm -C /dev/md15 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t5d0
echo "yes" | mdadm -C /dev/md16 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t6d0
echo "yes" | mdadm -C /dev/md17 -c ${CHUNK} -l 5 -n 5 -x 1 /dev/dsk/
c{0,1,4,5,6,7}t7d0
```

2. Check the status of rebuild from `/proc/mdstat`.

```
# cat mk-raid5_oss.sh
# cat /proc/mdstat
# watch -n 1 cat /proc/mdstat
Personalities : [raid1] [raid5]
md17 : active raid5 sdan[5] sdav[6] sdaf[3] sdx[2] sdp[1] sdh[0]
1953545984 blocks level 5, 256k chunk, algorithm 2 [5/4] [UUUU_]
in: 0 reads, 0 writes; out: 364160 reads, 90688 writes
179680 in raid5d, 79 out of stripes, 270720 handle called
reads: 0 for rmw, 0 for rcw
0 delayed, 2048 active, queues: 0 in, 2848 out
[>....................] recovery = 0.0% (355968/488386496) finish=137.0min
speed=59328K/sec
```

Modifying mdadm.conf

Linux RAID does not automatically start at boot time, so `/etc/mdadm.conf` needs to be configured to force the system to start RAID with the correct arrays. `mdadm.conf` is the Linux configuration file for managing software RAID with the `mdadm` command.

1. Modify the `/etc/mdadm.conf` file to define the arrays.

```
# cat /etc/mdadm.conf
DEVICE /dev/dsk/c*t*d*
ARRAY /dev/md11 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t1d0
ARRAY /dev/md12 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t2d0
ARRAY /dev/md13 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t3d0
ARRAY /dev/md14 level=raid5 num-devices=4 devices=/dev/dsk/c[01467]t4d0
ARRAY /dev/md15 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t5d0
ARRAY /dev/md16 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t6d0
ARRAY /dev/md17 level=raid5 num-devices=5 devices=/dev/dsk/c[014567]t7d0
ARRAY /dev/md21 level=raid1 num-devices=2 devices=/dev/dsk/c[47]t0d0s0
ARRAY /dev/md22 level=raid1 num-devices=2 devices=/dev/dsk/c[47]t0d0s1
ARRAY /dev/md23 level=raid1 num-devices=2 devices=/dev/dsk/c[47]t0d0s2
ARRAY /dev/md24 level=raid1 num-devices=2 devices=/dev/dsk/c[47]t0d0s3
ARRAY /dev/md25 level=raid1 num-devices=2 devices=/dev/dsk/c[16]t0d0s0
ARRAY /dev/md26 level=raid1 num-devices=2 devices=/dev/dsk/c[16]t0d0s1
ARRAY /dev/md27 level=raid1 num-devices=2 devices=/dev/dsk/c[16]t0d0s2
```

2. Use the `-S` option for `mdadm` to deactivate the arrays, and release all resources.

```
# mdadm -S /dev/md*
mdadm: fail to stop array /dev/md1: Device or resource busy
mdadm: fail to stop array /dev/md2: Device or resource busy
mdadm: fail to stop array /dev/md3: Device or resource busy
```

3. Use the `-A` option with `--scan` to scan the config file (`mdadm.conf`) and assemble the arrays.

```
# mdadm -A --scan
mdadm: /dev/md11 has been started with 5 drives and 1 spare.
mdadm: /dev/md12 has been started with 5 drives and 1 spare.
mdadm: /dev/md13 has been started with 5 drives and 1 spare.
mdadm: /dev/md14 has been started with 4 drives and 1 spare.
mdadm: /dev/md15 has been started with 5 drives and 1 spare.
mdadm: /dev/md16 has been started with 5 drives and 1 spare.
mdadm: /dev/md17 has been started with 5 drives and 1 spare.
mdadm: /dev/md21 has been started with 2 drives.
mdadm: /dev/md22 has been started with 2 drives.
mdadm: /dev/md23 has been started with 2 drives.
mdadm: /dev/md24 has been started with 2 drives.
mdadm: /dev/md25 has been started with 2 drives.
mdadm: /dev/md26 has been started with 2 drives.
mdadm: /dev/md27 has been started with 2 drives.
```

Formatting the OSTs

Lustre configuration information is stored in eXtensible Markup Language (XML) files that can be created and updated using the Lustre make configuration (`lmc`) utility. The Lustre configuration (`lconf`) utility enables administrators to configure Lustre on specific nodes using user specified configuration files. For more information on using these commands to configure, start, reconfigure, stop, or restart Lustre services, see the Lustre documentation provided at: *https://lustre.org/manual.html.*

1. lconf hits a bug with the external journal. This bus is fixed in more recent versions of Lustre. Modify `lconf` as follows as a workaround for this bug.

```
 cp /usr/sbin/lconf /usr/sbin/lconf.orig
# cat /usr/sbin/lconf.orig | sed -e 's/blkid -o device/blkid -o device -l/
g' > /usr/sbin/lconf
# diff /usr/sbin/lconf.orig /usr/sbin/lconf
839c839
<             blkid = "blkid -o device -t UUID='%s'" % (journal_UUID)
---
>             blkid = "blkid -o device -l -t UUID='%s'" % (journal_UUID)
```

2. Create the Lustre xml configuration file using the `lmc` command.

```
#!/bin/sh

MDS="tgt075140"
MDS_OSS="tgt075140"
MDT="/dev/md14"
MDSNAME="work-mds"
LOVNAME="work-lov"
OSS="tgt075139 tgt075138 tgt075137 tgt075136"
CONFIG=config.xml
LMC="lmc -m"

# Create Nodes
if [ -f ${CONFIG} ]; then
rm -f ${CONFIG}
fi

# Configure Nets
${LMC} ${CONFIG} --add net --node ${MDS} --nettype lnet --nid ${MDS}@vib
for host in ${OSS}; do
${LMC} $CONFIG --add net --node ${host} --nettype lnet --nid ${host}@vib
done
${LMC} ${CONFIG} --add net --node client --nettype lnet --nid '*'

# Configure MDS
${LMC} ${CONFIG} --add mds --node ${MDS} --mds ${MDSNAME} --dev ${MDT}

# Configure OSTs
${LMC} ${CONFIG} --add lov --lov ${LOVNAME} --mds ${MDSNAME}

# for MDS and OSS
if [ ! -z "${MDS_OSS}" ]; then
for i in 1 2 3 5 6 7; do
${LMC} ${CONFIG} --add ost --node ${MDS} --lov ${LOVNAME} --ost ${MDS}-
md1${i}  --mountfsoptions "mballoc,extents" --dev /dev/md1${i} --
mkfsoptions "-j -J device=/dev/md2${i}"
done
fi

# for OSS
for host in ${OSS}; do
for i in 1 2 3 4 5 6 7; do
${LMC} ${CONFIG} --add ost --node $host --lov ${LOVNAME} --ost ${host}-
md1${i} --mountfsoptions "mballoc,extents" --dev /dev/md1${i} --mkfsoptions
"-j -J device=/dev/md2${i}"
done
done

# Configure client
${LMC} ${CONFIG} --add mtpt --node client --path /work --mds ${MDSNAME} --
lov ${LOVNAME}
```

3. After saving the script, run the script to generate the config.xml file.

```
# ./config.sh
```

4. Verify that the script created the correct values.

```
# ls config.xml
```

Put the config.xml file at a location where all of the nodes can access it, for example in an NFS share. When Lustre is started for the first time it uses the config.xml file to reformat and start the OSTs.

5. Reformat and start the OSTs.

```
# lconf --reformat config.xml
loading module: libcfs srcdir None devdir libcfs
loading module: lnet srcdir None devdir lnet
loading module: ksocklnd srcdir None devdir klnds/socklnd
loading module: lvfs srcdir None devdir lvfs
loading module: obdclass srcdir None devdir obdclass
loading module: ptlrpc srcdir None devdir ptlrpc
loading module: ost srcdir None devdir ost
loading module: ldiskfs srcdir None devdir ldiskfs
loading module: fsfilt_ldiskfs srcdir None devdir lvfs
loading module: obdfilter srcdir None devdir obdfilter
NETWORK: NET_tgt075139_lnet NET_tgt075139_lnet_UUID lnet tgt075139@vib
OSD: tgt075139-md11 tgt075139-md11_UUID obdfilter /dev/md11 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md12 tgt075139-md12_UUID obdfilter /dev/md12 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md13 tgt075139-md13_UUID obdfilter /dev/md13 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md14 tgt075139-md14_UUID obdfilter /dev/md14 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md15 tgt075139-md15_UUID obdfilter /dev/md15 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md16 tgt075139-md16_UUID obdfilter /dev/md16 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
OSD: tgt075139-md17 tgt075139-md17_UUID obdfilter /dev/md17 0 ldiskfs no 0
256
OST mount options: errors=remount-ro,mballoc,extents
```

The `lconf` command loads all of the required Lustre and LNET modules and also performs low-level configuration of every device using `lctl`. It is essential to use the `--reformat` option the first time to initialize the file systems. Caution: If the `--reformat` option is used on subsequent attempts to bring up the Lustre system, it will re-initialize the file systems.

Setting Up the Meta Data Server

In the TSUBAME grid, 10 Sun Fire X4500 servers are used for the largest file system. In addition, 3 servers are used as meta data servers for smaller file systems. A MDS needs to be set up for each file system.

It is also necessary to run `lconf --reformat` on the Meta Data Target (MDT), which should occur after all of the OSTs are configured. Because the MDS contains all of the meta data for the file systems, it requires RAID 1 or RAID 1+0 (using the latest `mdadm` utility), rather than RAID 5. If a large MTD is required, it can be configured with `lvm`(8).

The size of the meta data depends on the number and type of files. For example, a heavily striped file requires more than 4K for meta data, whereas a non-striped requires less space to contain its meta data information. Cluster File Systems recommends planning for 4K of metadata per file. The following configuration is a multi-service with OSS and MDS on the same Sun Fire X4500 server. In this case, md14 is configured with RAID1 (RAID1+0) for the MDT.

1. Use the `-S` option for `mdadm` to deactivate the array, and release all resources.

```
# mdadm -S /dev/md14
```

2. Create a new array.

```
# mdadm -C /dev/md14 -l 1 -n 2 /dev/dsk/c{0,1}t4d0
```

3. Update `/etc/mdadm.conf`.

```
ARRAY /dev/md14 level=raid1
num-devices=2
devices=/dev/dsk/c[01]t4d0
```

4. Modify the following configurations in `config.sh` for MDS and MDT.

```
MDS="tgt075140" # MDS hostname
MDS_OSS="tgt075140" # if MDS and OSS are same machine
MDT="/dev/md14" # MDT device
```

5. Run the `config.sh` script to re-make the Lustre configuration, then reformat.

```
# ./config.sh
# lconf --reformat config.xml
```

6. Start the MDS.

```
# mkdir /etc/lustre
# cp config.xml /etc/lustre
# /etc/init.d/lustre {start}
```

Configuring Clients on the Sun Fire X4600 Servers

The Lustre client requires a configuration file, but Luster also supports 0-config, which can use the mount command. Therefore, the only step is to mount the file system.

1. Mount the file system on the clients.

```
# mount -t lustre tgt075140@vib:/work-mds/lustre /work
```

2. It is also advisable to configure the `/etc/fstab` file to mount the Lustre file system at the boot time. Modify the `/etc/fstab` with the following line.

```
tgt075140@vib:/work-mds/lutre /work lustre defaults,rw,_netdev 0 0
```

Lustre supports an HA-MDS configuration with shared disk. This feature is not used in the initial TSUBAME configuration but is planned for the future to provide higher availability.

Chapter 5
Software RAID and Disk Monitoring

The `mdadm` command includes an array monitoring feature (`--monitor` option). If the array is changed, `mdadm` can be configured to send an email to the system administrator. Sendmail must first be configured.

1. Add the following line into `/etc/mdadm.conf` to configure administrator notification.

```
MAILADDR yourname@email_address
```

The code below is a sample of `mdadm --monitor` when one drive has failed on the array. In this case, mdadm sent a Fail event to the system administrator.

```
Subject: Fail event on /dev/md13:tgt075124
From: root
To: System Administrator

This is an automatically generated mail message from mdadm
running on tgt075124

A Fail event had been detected on md device /dev/md13.

Faithfully yours, etc.
```

The Sun Fire X4500 server's drives support Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) and Red Hat Enterprise Linux 4 includes the S.M.A.R.T. utilities `smartctl`(8) and `smartd`(8). `smartctl` can access the S.M.A.R.T. error and log information and perform self tests. For example, to see a `smart-a.log` file, use the following command:

```
#smartctl -d marvell -a /dev/dsk/c4t0d0
```

S.M.A.R.T. errors and changes to S.M.A.R.T. attributes are logged via the SYSLOG interface. And `smartd` can send email alerts to the system administrator.
The `smartd` daemon can be used to monitor changes in drive status for all of the drives. The daemon's configuration is below.

1. To configure `/etc/smartd.conf` add the following lines for all 48 drives.

```
/dev/dsk/c0t0d0 -H -d marvell -m root
/dev/dsk/c0t1d0 -H -d marvell -m root
/dev/dsk/c0t2d0 -H -d marvell -m root
/dev/dsk/c0t3d0 -H -d marvell -m root
...
```

2. Then, start the S.M.A.R.T. daemon.

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

If `smartd` detects an error event, smartd sends an email similar to the one below to the system administrator.

```
Subject: SMART error (FailedOpenDevice) detected on host: tgt075107
From: root
To: System Administrator

This email was generated by the smartd daemon running on:

host name: tgt075107
DNS domain: [Unknown]
NIS domain: (none)

The following warning/error was logged by the smartd daemon:

Device: /dev/dsk/c1t0d0, unable to open device

For details see host's SYSLOG (default: /var/log/messages).

You can also use the smartctl utility for further investigation.
No additional email messages about this problem will be sent.
```

Chapter 6
Summary

TSUBAME grid is the first production system combining Linux software RAID and the Lustre file system on an enterprise grid cluster. All data in the TSUBAME grid is stored on 42 Sun Fire X4500 data servers providing 1 PB of storage. Access to applications and data stored on these systems is provided by the Lustre file system. The TSUBAME grid achieved over 800 MB/sec. aggregate write performance per Sun Fire X 4500 server. The initial configuration of 10 Sun Fire X4500 servers configured as OSSs yielded nearly 8 GB/sec. aggregate performance, providing high performance file services for the TSUBAME grid. Since Lustre is a scalable file system, more Sun Fire X4500 servers can be added to accommodate data growth and requirements for increased I/O performance. In addition, mirrored copies of a transaction journal help ensure data availability and integrity in the event of a hardware or software failure.

About the Author

Syuuichi Ihara joined Sun in 2001 as a systems consultant in Sun's professional services organization, designing and implementing solutions combining Sun and third-party technologies to meet customer requirements. He is now a solution architect in Sun's system practice organization. As the TSUBAME grid consists of new technologies from both Sun and Lustre, his extensive experience was very beneficial in building the complex software stacks in the TSUBAME grid cluster.

Acknowledgements

The writer would like to thank the following people for their help and contributions.

- Professor Satoshi Matsuoka of the Tokyo Institute of Technology, for his advice on TSUBAME grid architecture and performance
- Peter Bojanic and Jody McIntyre of Cluster File System, Inc., for support on Lustre file system architecture
- Yiftah Shahar of Voltaire, Inc., for his advice on InfiniBand Network
- Larry McIntosh, for leading the TSUBAME enterprise HPC storage architecture
- Hanxi Chen, for leading the Sun Fire X4500 servers on the TSUBAME HPC grid project
- Marshall Choy, for providing insight into the Sun x64 server architecture
- Junichi Koike, for leading the TSUBAME grid architecture design effort and advice on the implementation
- Toshihiro Kujiraoka, for leading the TSUBAME grid project and providing tremendous support throughout our system delivery
- All other individuals involved in the TSUBAME grid project

References

TOP500 Supercomputer Sites
`http://www.top500.org`

Lustre Documentation
`http://mail.clusterfs.com/wikis/lustre/LustreDocumentation`

Sun Fire X4500 Server Documentation
`http://www.sun.com/products-n-solutions/hardware/docs/Servers/x64_servers/x4500/index.html`

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` Web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is
`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:
`http://www.sun.com/blueprints/online.html`