Lustre Tutorial

Presented by:
Rick Slick, Cray Inc.

- CUG 2006

Outline for this Tutorial

- Lustre overview
- Lustre on the Cray XT3
  - Including llrd and liblustre
- Lustre on the Cray XD1
- General Lustre discussion
- Reconfiguring Lustre
- Trouble-shooting

Other Presentations

- Storage Architectures for Cray-based Supercomputing Environments
  - Matthew O'Keefe, Cray Inc.
  - Tuesday at 10:30
- Lustre File System Plans and Performance on Cray Systems
  - Charlie Carroll and Branislav Radovanovic, Cray Inc.
  - Thursday at 8:20
- XT3 Status and Plans
  - Charlie Carroll and David Wallace, Cray Inc
  - Thursday at 11:45

What is Lustre

- A fairly elaborate storage architecture
- A scalable cluster file system for Linux
- Developed by Cluster File Systems, Inc.
- Name derives from “Linux Cluster”
- The Lustre file system consists of software subsystems, storage, and an associated network

Lustre Attributes

- POSIX compliant
- Layering of object-based disks (OBDs)
- Distributed-lock management
- Separate metadata and file data servers
- No single points of failure
- Open source

Lustre Fundamentals

- The concept of object storage is basic to Lustre
  - Objects can be thought of as inodes and are used to store file data. Lustre inodes simply contain references to the object storage target (OST) that stores the file data
  - Access to these objects occurs through object storage servers (OSTs), which provide the file I/O service
  - The OSTs perform the block allocation for data objects, which results in distributed and scalable allocation
  - The client-OST protocol combines request processing with remote DMA

Lustre Fundamentals

- The namespace is managed by metadata services that manage the Lustre inodes
  - The services perform file lookups, file creation, file and directory attribute manipulation
  - Such inodes can be directories, symbolic links, or special devices
  - The associated data and metadata is stored on the metadata servers

Lustre Fundamentals

- Lustre networking
  - Lustre enhances Portals with the concept of a network
  - Lustre request processing is built on a thin API, called the Portals API, which was developed at Sandia
    - This API provides for delivery and event generation in connection with network messages and provides advanced capabilities such as using Remote DMA (RDMA)

Lustre Terminology

- MDS – metadata server
  - The Server node
- MDT – metadata target
  - This is the software interface to the backend volume
  - Controls filesystem metadata (inodes) and locking mechanism
    - The backend volume is an ext3 file system
    - On Cray systems this is a RAID volume
    - LUNs are formatted with 4096 byte blocks

Lustre Terminology

- OSS – object storage server
  - The server node
  - Support multiple OSTs
- OST – object storage target
  - This is the software interface to the backend volume
    - The backend volume is an ext3 file system
    - LUNs are formatted with 4096 byte blocks
    - The multi-block allocator (MBA) (Linux 2.6) is used for performance
    - LUN size is limits to 2 TB
- OBD – object-based disk

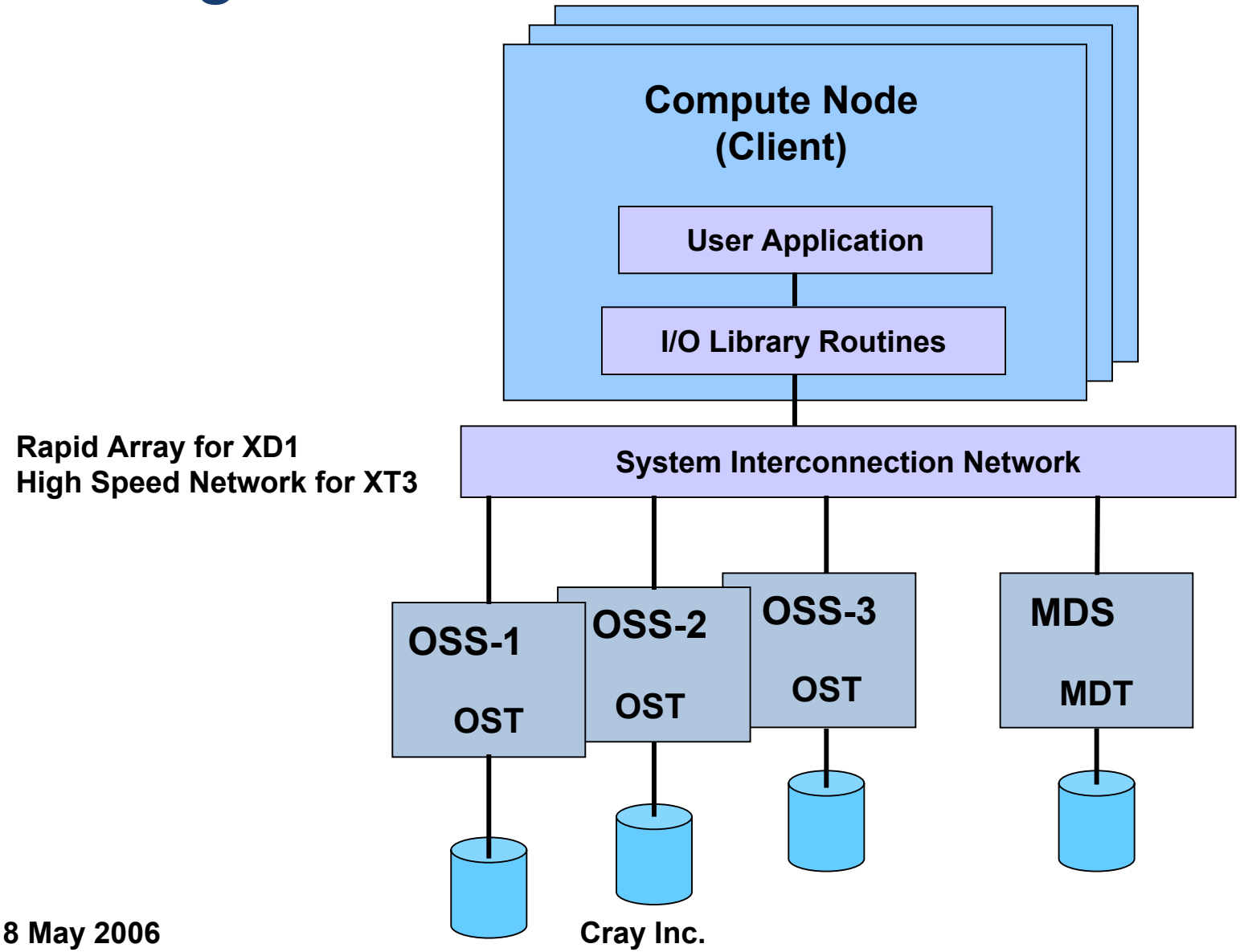
Lustre Terminology

- LOV – logical object volume
  - Combined structure of OSTs in a filesystem
- NAL – Network Abstraction Layer
  - Cray XD1 Lustre hides behind this
  - Portals functionality

Lustre Terminology

- Zero Configuration
  - Clients just mount Lustre like an NFS exported filesystem
- Stripes
  - Defines the number of OSTs to write the file across
    - Recommend not striping across all OSTs
    - Recommend a default stripe count of one to four
  - Can be set on a per file or directory basis
    - When set on directories, inheritance is the new file will be created with directory settings
- liblustre – Applications interface to Lustre I/O
  - Cray XT3 only, not on Cray XD1

Lustre High-level Overview



Lustre Commands

- **`lconf`** (Lustre configuration utility) reads the .xml file, configures/starts lustre on MDS, OSS nodes
  - Options to start, stop, and reformat Lustre
- **`lmc`** (Lustre configuration maker) creates a .xml file that **`lconf`** uses to create a Lustre file system
- **`lctl`** (Lustre control utility) low-level configuration utility
  - Low level control and visibility
  - Be very careful

Lustre Commands

- **`lfs`** Lustre utility that can be used to create a file with a specific striping pattern, displays file striping patterns, and find file locations
  - Suboptions:
    - **`setstripe`**
    - **`getstripe`**
    - **`find`**

Lustre Limitations

- Maximum number of OSTs is 512
- Maximum number of inodes per file system
  - Device size (2 TB MDT) / 4096 = 512M
    - ext3 default is based on file system block size
- Maximum files per directory is ~25 million
  - The directory size is ~2 GB

Lustre Limitations

- The maximum file size is 360 TB
  - Maximum number of stripes per file is 180
    - 2 TB x 180 = 360 TB
- Sparse files are supported
  - Seek beyond EOF, unallocated hole
- Filename/pathname limits
  - Filename 255 bytes
  - Pathname 4096 bytes

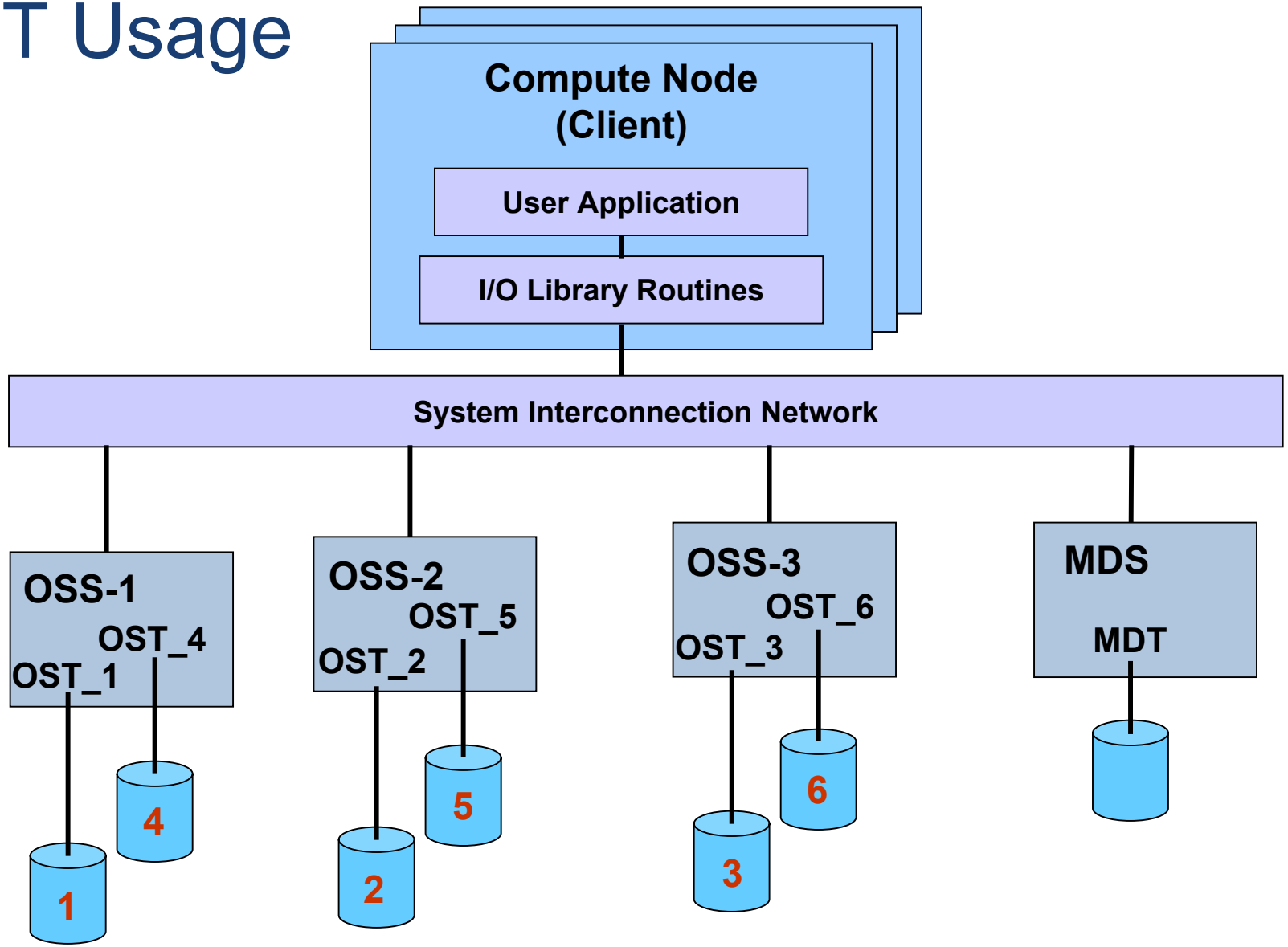
Server RAM

- Most MDS nodes do not need huge amounts of RAM; however, high performance on big directories does
  - Consider a directory with 10,000,000 files:
    - To cache 2/3 of this directory + inodes requires ~3 GB ram
    - If you don't have the ram, a random lookup requires a disk seek
    - If you do, Lustre can sustain random lookups 5000/second
  - The same is true of OSTs with many files

General Configuration Guidelines

- Single MDS per filesystem
- Up to 512 OSTs
- 10000+ clients
- Order use of OSTs across OSSs

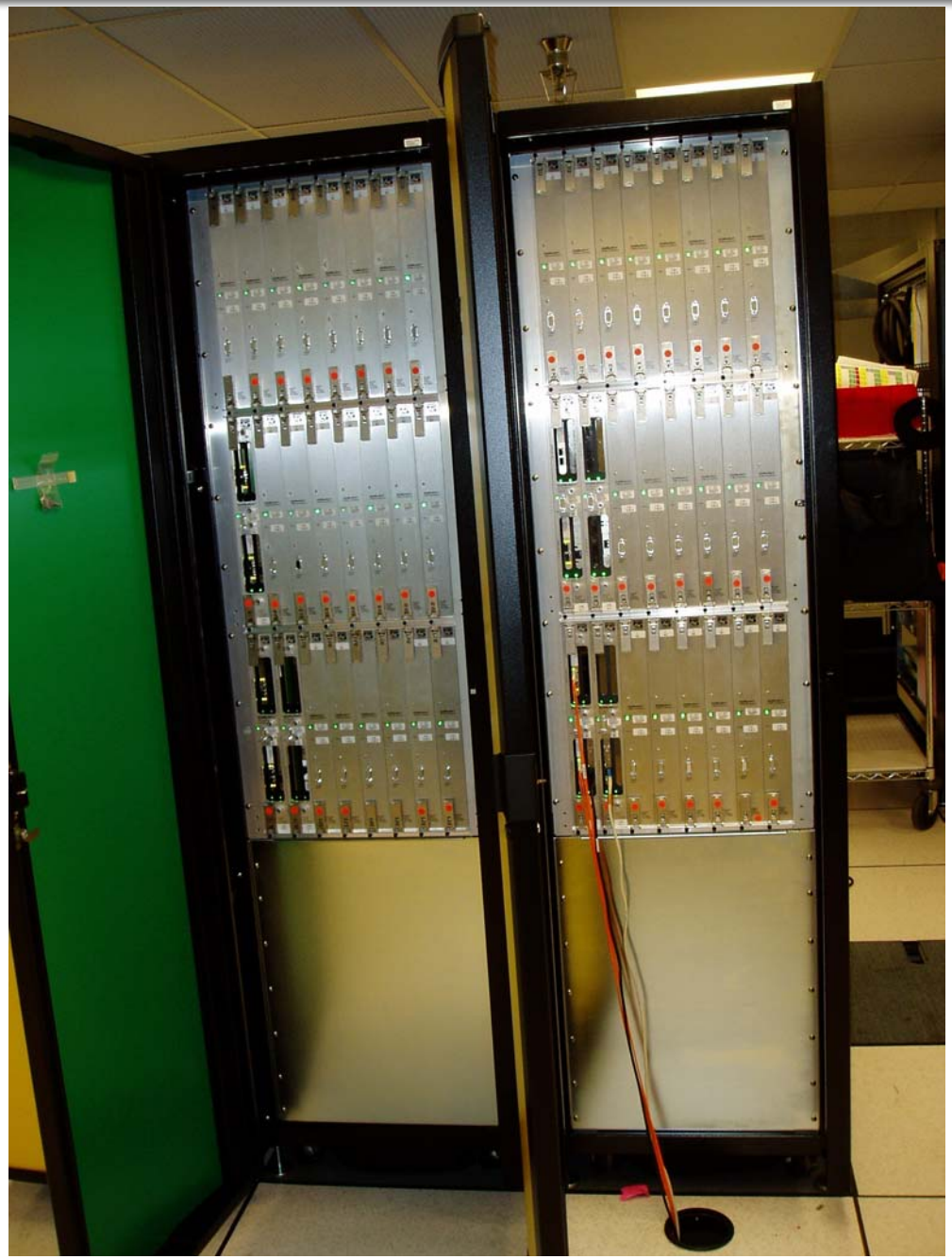
OST Usage


Compute Node
(Client)
User Application
I/O Library Routines
System Interconnection Network
OSS-1
OST_4
OST_1
OSS-2
OST_5
OST_2
OSS-3
OST_6
OST_3
MDS
MDT
1
4
2
5
3
6

Lustre on the Cray XT3

- CUG 2006

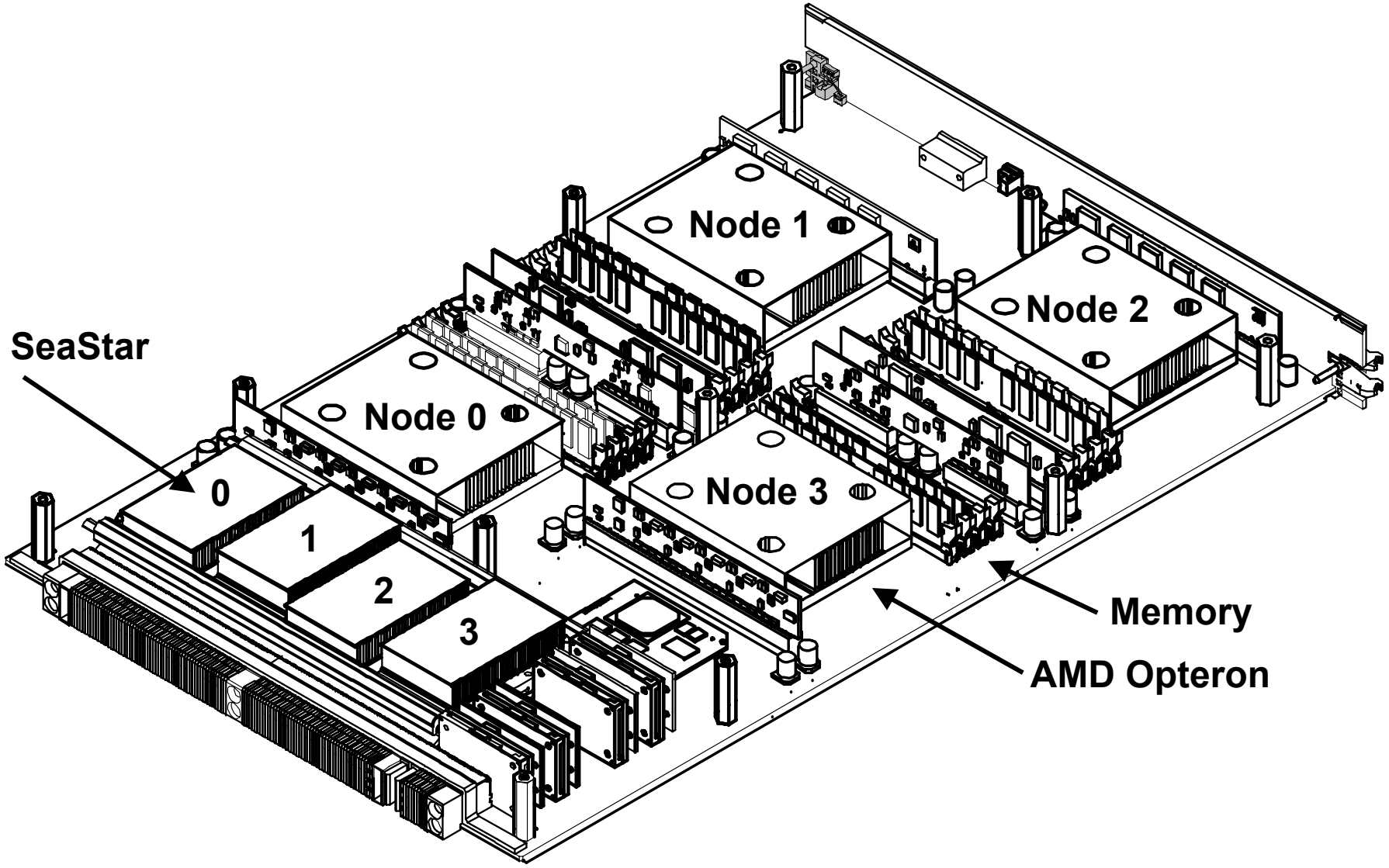
Cray XT3 System



Blade Types

- The Cray XT3 uses two blade types
  - Compute blades
    - Four compute nodes
  - Service blades
    - Two service nodes
    - Four PCI-X slots

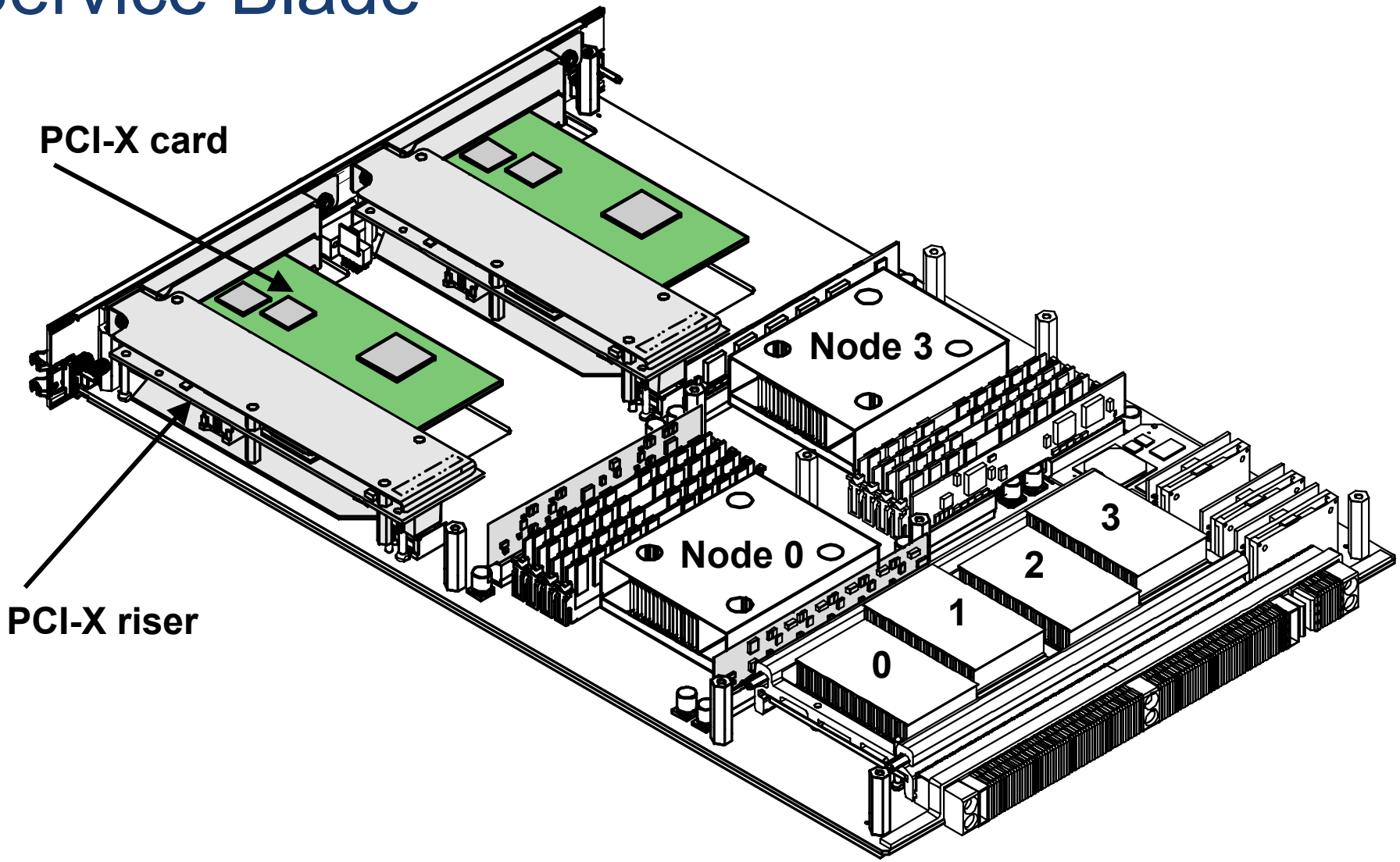
Compute Blade



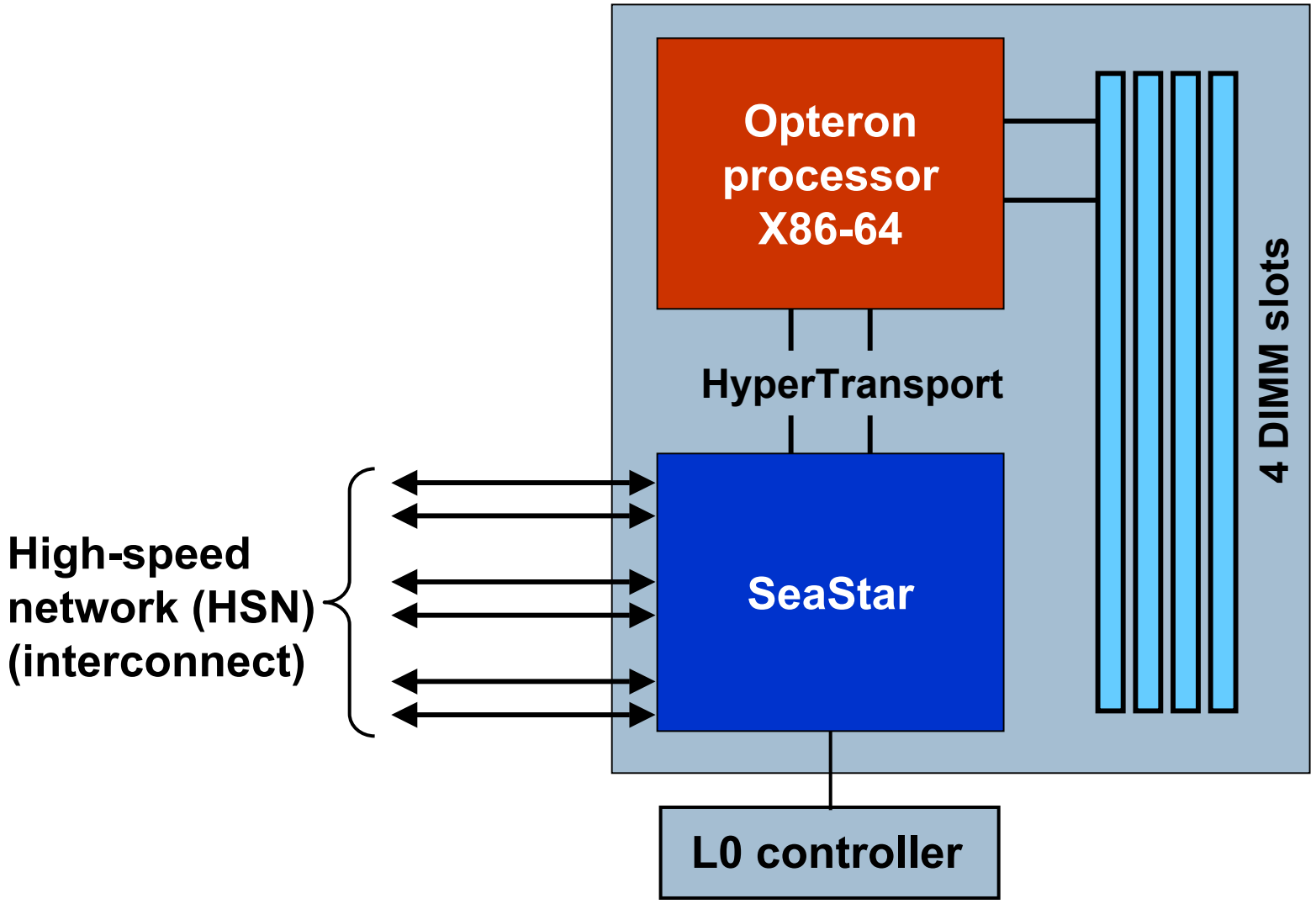
Service Blades

- Two nodes; each node contains:
  - One AMD Opteron socket
    - Single or dual core
  - Maximum of 8 GB of memory
  - One SeaStar chip
- Processor and memory VRMs
- Mezzanine card
  - Four SeaStar chips
    - Two are connected to the Opterons on the Service blade
- 2 PCI-X risers
  - One 2-slot riser per node
  - AMD 8131 bridge chip

Service Blade

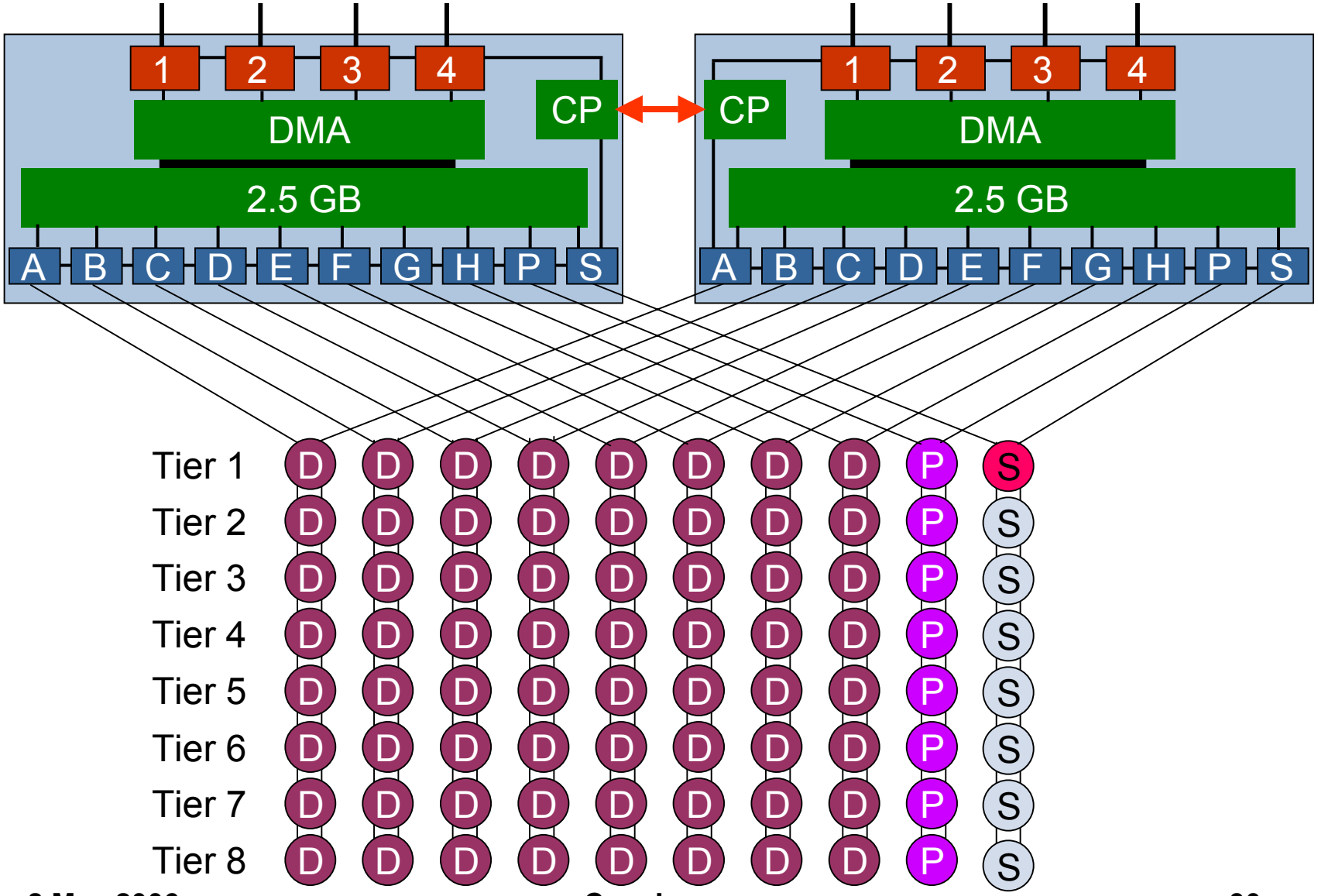


Node Block Diagram

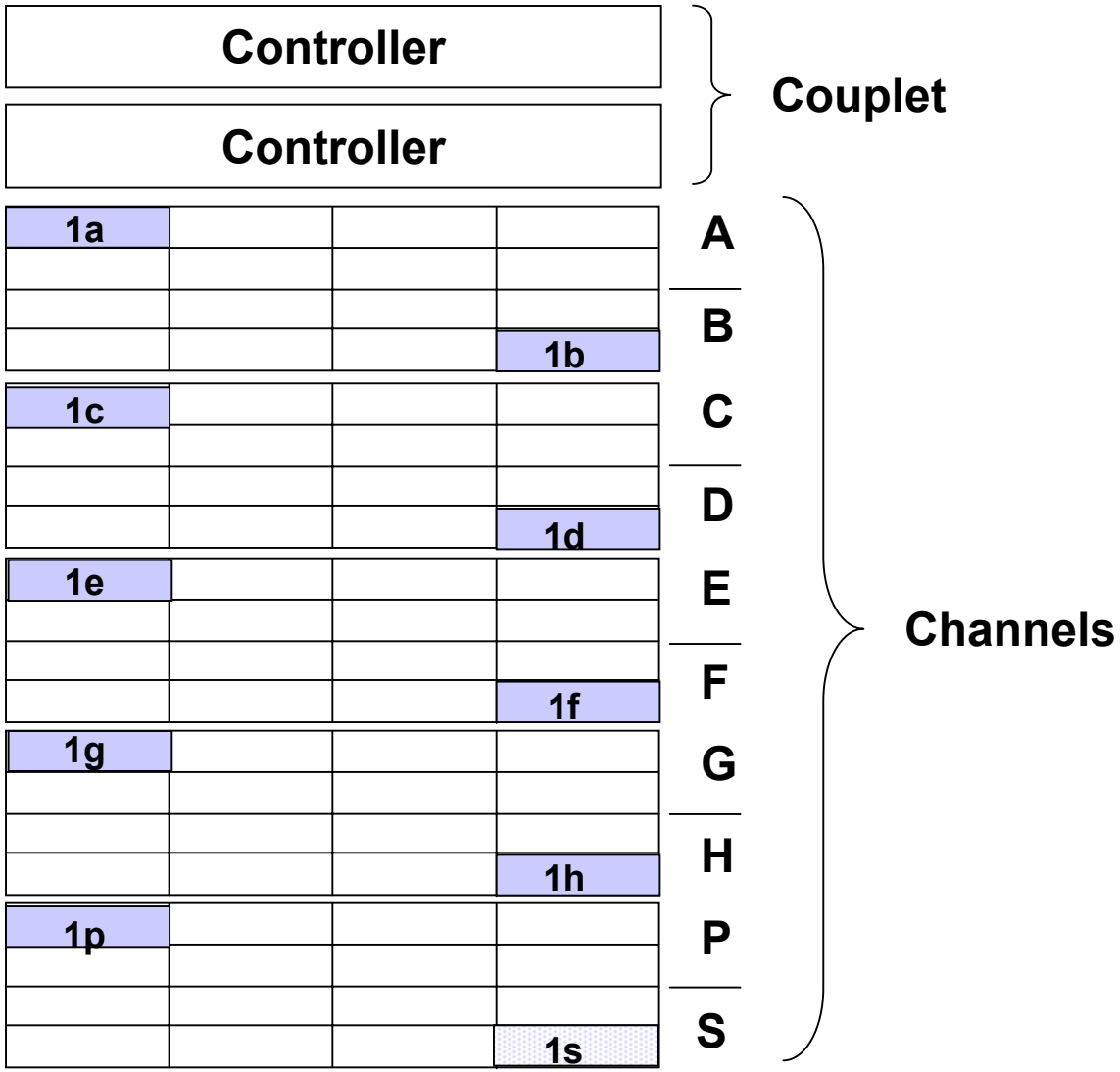


DDN Disk Enclosure

DDN RAID Illustration S2A8500

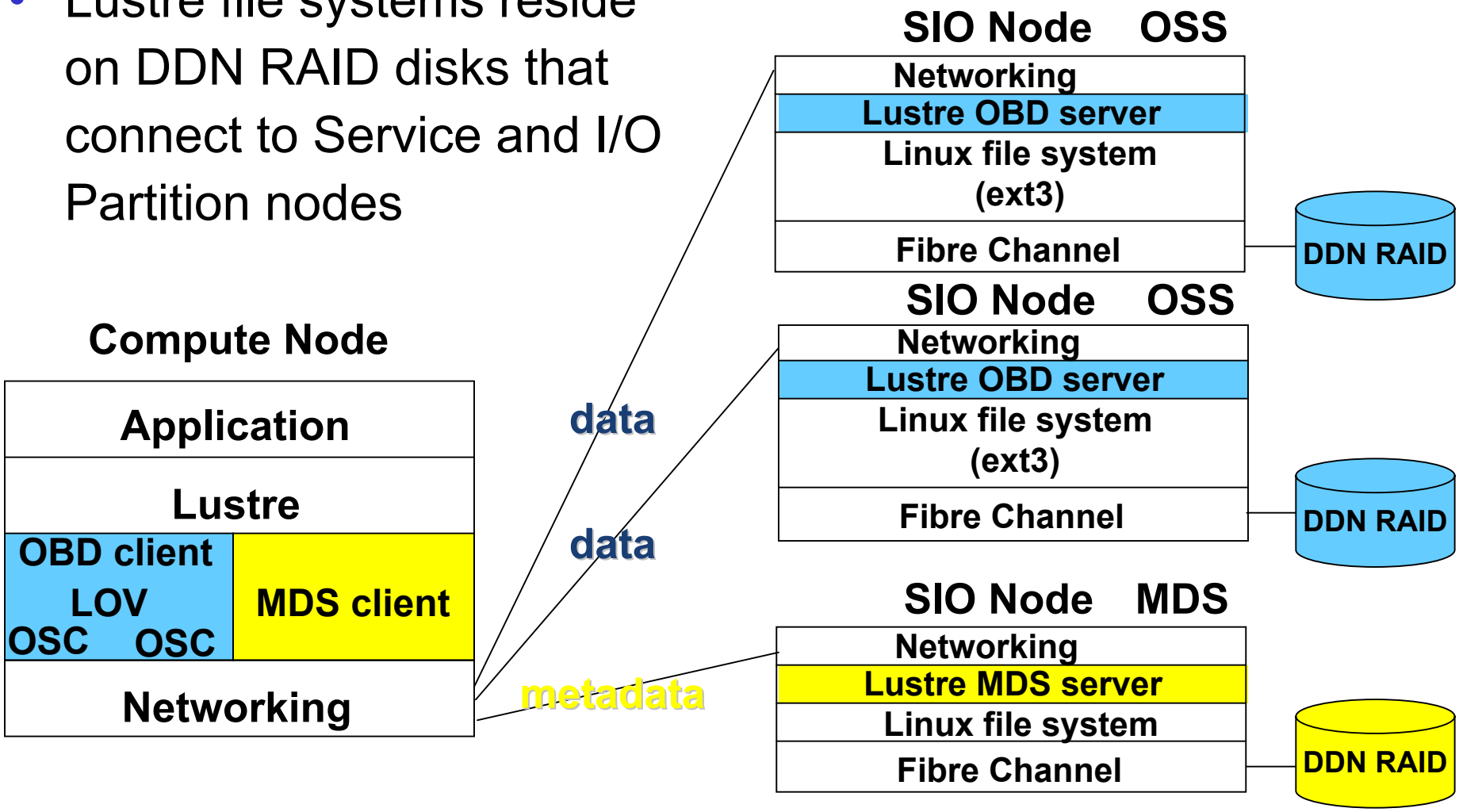


DDN Tier Configuration S2A8500



Lustre on Cray XT3

- Lustre file systems reside on DDN RAID disks that connect to Service and I/O Partition nodes



Cray XT3 Lustre Configuration

- 1 MDS per file system
- Keep the MDS and OSSs on separate nodes
  - Do this to avoid double failure
- The Lustre configuration file is an XML file
  - According to the Cray standard the configuration file is placed in **`/etc/lustre`** in the sharedroot file system (**`xtopview`**)
  - The easiest way to create the .xml file for **`lconf`** to use is to create a configuration script
    - The shell script contains **`lmc`** commands
    - The shell script creates the configuration file when it executes

Lustre Configuration Script

- Find a sample configuration script
- Copy the script to a file name that reflects your configuration
- Edit the top portion of the script to match your configuration
  - Setup variables for MDS, OSS hosts, MDT, OST devices, xml file name, default stripe info., etc.

Lustre Configuration File

```
#!/bin/bash
#########################################################
### Modify the following parameters as appropriate
#########################################################

# pathname to Lustre XML configuration file
LUSTRE_XML=/etc/lustre/4oss_8ost.xml

# MDS information
MDSHOST=nid00135
MDSNAME=${MDSHOST}_mds
MDSNID=135
MDSDEV=/dev/sda
```

Lustre Configuration File

```
# LOV information
LOVNAME=${MDSHOST}_lov
# Three arrays create a 2-D matrix of OST information.
# Ensure that information for individual OSTs is
# positioned at the same index in each array.
# Space separated array of OST hostnames
declare -a OSTHOST=( nid00128 nid00131 nid00136
nid00139 nid00128 nid00131 nid00136 nid00139 )
# Space separated array of OST host NIDs
declare -a OSTNID=( 128 131 136 139 128 131 136 139 )
# Space separated array of OST block devices
declare -a OSTDEV=( /dev/sda /dev/sda /dev/sda
/dev/sda /dev/sdb /dev/sdb /dev/sdb /dev/sdb )
```

Lustre Configuration File

```
# Set default stripe parameters
STRIPE_SIZE=1048576
STRIPE_COUNT=2
# Generally best to leave set at 400
EXT3_JRNL_SIZE=400
# Lustre interconnect network type
NETTYPE=cray_kern_nal
# Client mount on MDS server
MDS_MOUNT_POINT=/mnt/lustre_server
# Set to nonzero-length value for verbose output
VERBOSE=yes
##########################################################
#####     END OF modifiable parameters                ####
```

Lustre Configuration

- For UNICOS/lc 1.3
  - Update /etc/modules.conf.local

```
alias below ptlrpc libcfs
alias below llite lov osc
alias lustre llite
```

- For UNICOS/lc 1.4.x
  - Update /etc/modprobe.conf.local

```
options lnet networks=ptl
alias lustre llite
```

  - LNET – Lustre Network
  - LND – Lustre network device

Additional Cray XT3 Lustre Limitations

- In UNICOS/lc 1.3.x (2.4 kernel)
  - The maximum useable LUN size is 1 TB
- In UNICOS/lc 1.4.x (2.6 kernel)
  - The maximum useable LUN size is 2 TB
  - The maximum file system size is 1 PB

Using Lustre on Cray XT3

- Using the standard programming environment (PrgEnv) **`liblustre`** is automatically linked to program
  - Is required to send I/O to a lustre filesystem
  - Upgrades may requires applications to relink before running
- Catamount limitations limit knowledge of the application state at abnormal termination
  - Solution: lustre lock recovery daemon (llrd)
    - At application termination pass the application nid list to **`llrd`**
    - **`llrd`** then communicates with the MDS to evict the clients

Eviction

- Eliminate server state for a client after a timeout or server failure has occurred
  - Evict all compute nodes post-job completion
  - As a result syslog contains many alarming messages, like:

```
LustreError: …obd_export_evict_by_nid() evicting nid
```

  - This is being investigated and will be resolved in a future release

Cray XT3 Lustre Documentation

- Cray XT3 Installation Guide (S-2444)
- Cray XT3 System Management (S-2393)
- Man pages

Lustre Releases

- Lustre 1.4.5 (UNICOS/lc 1.3)
  - ext3 filesystem with enhancements (patches)
- Lustre 1.4.6 (UNICOS/lc 1.4)
  - **`ldiskfs`** kernel module (to reduce size of lustre kernel patch)
  - Added performance enhancements
    - extents
    - mballoc

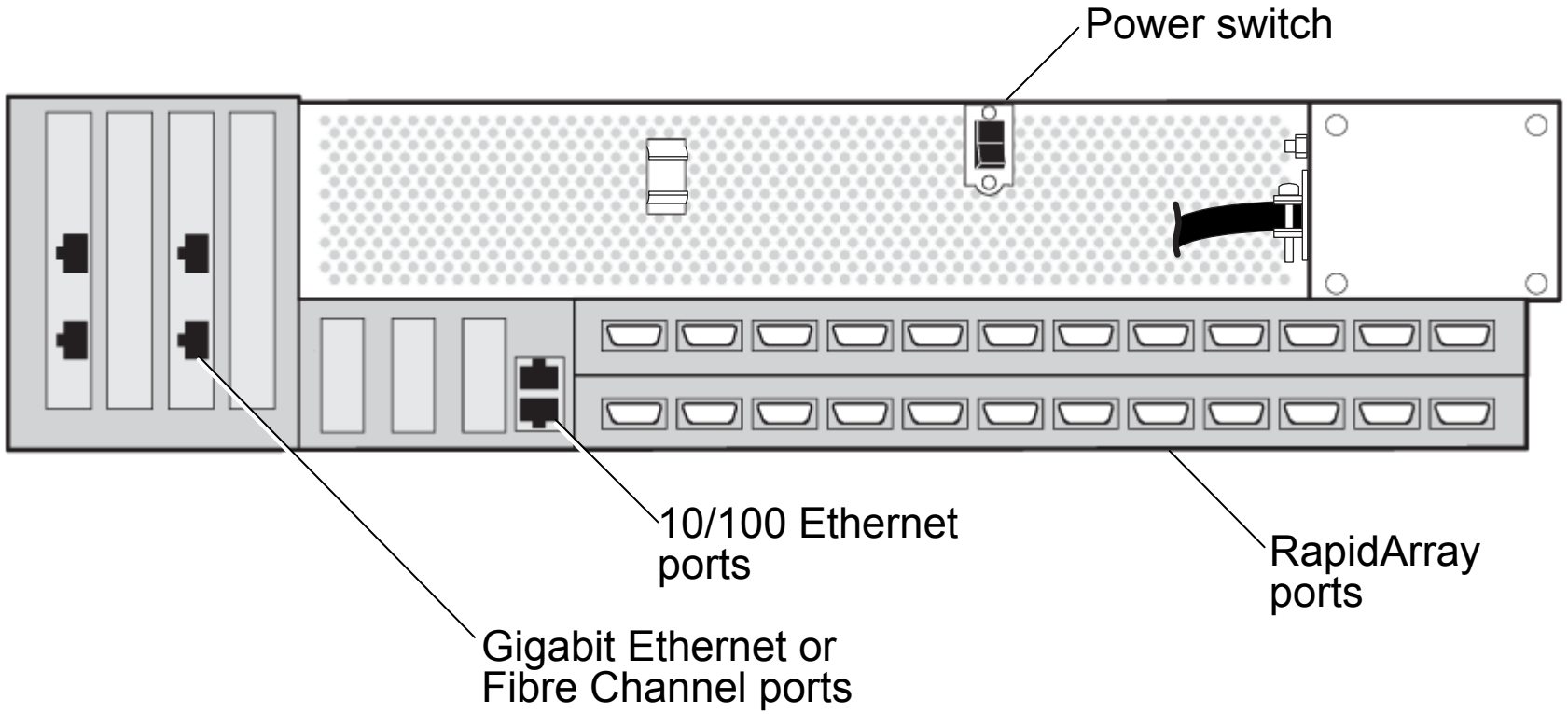
Lustre on the Cray XD1

- CUG 2006

Cray XD1 System

Cray XD1 Chassis

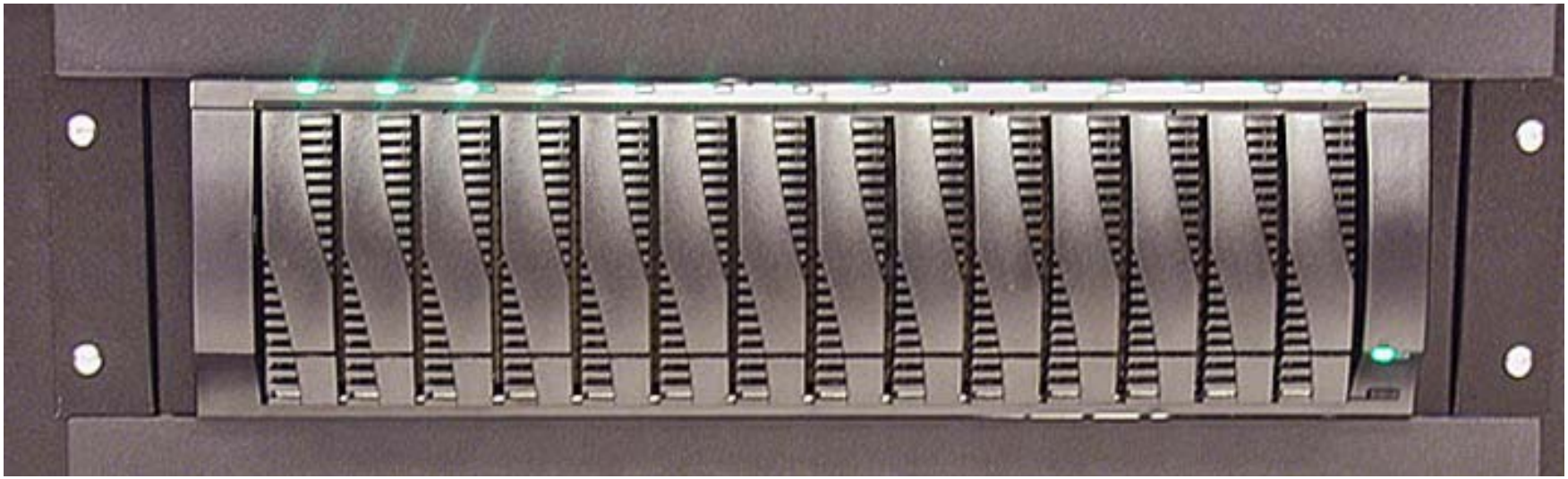
Chassis Rear View



Cray XD1 Disks

- The Cray XD1 system can be configured with either Engenio or DDN disk subsystems
  - The DDN products is
    - The S2A 8500
      - RAID 5 8+1+1
    - These were discussed with the Cray XT3 system
  - The Engenio Products are
    - The Engenio 5884
      - Separate controller and disk enclosure
    - The Engenio 2884 and 2882
      - Disks and controller in a single enclosure

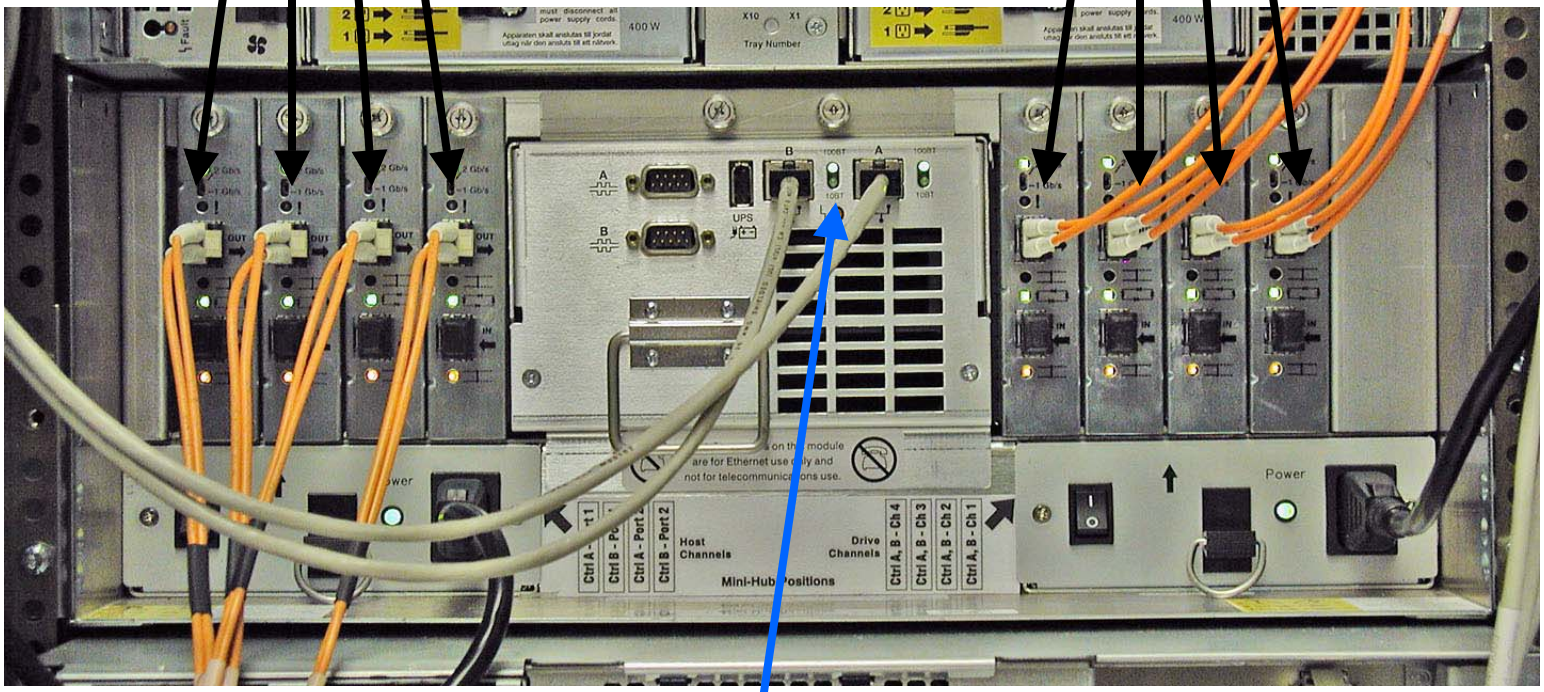
Engenio Disks



Engenio 5884 Controller

Front end connections (From XD1)

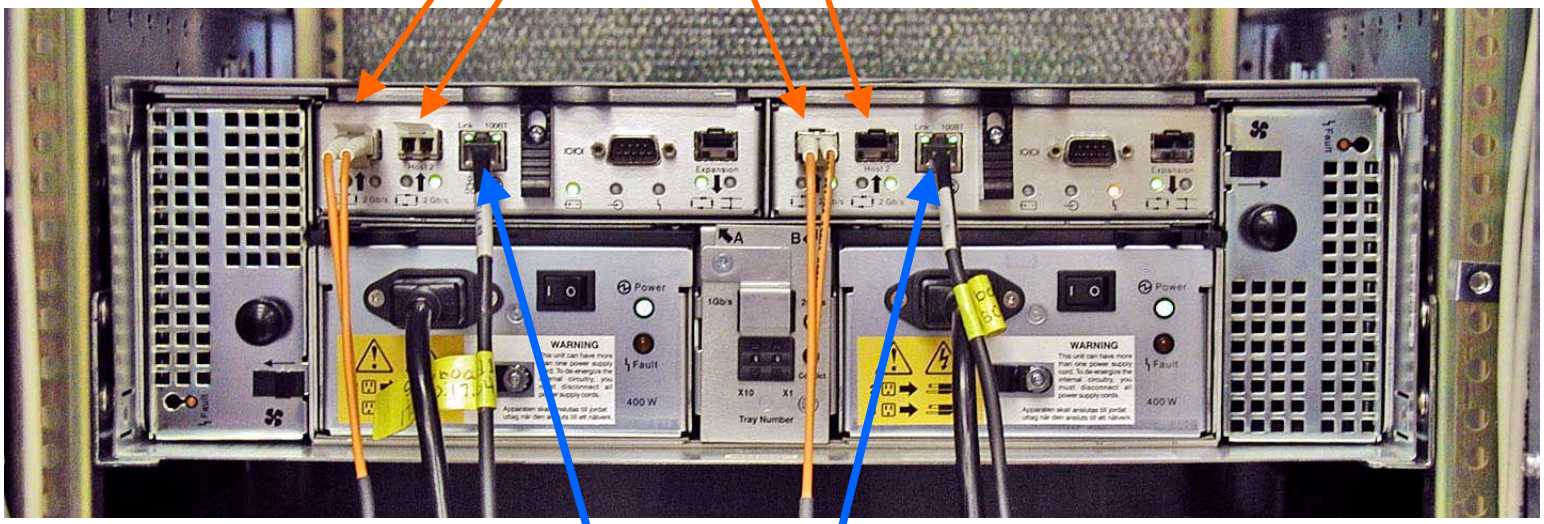
Back end connections (To disks)



Telnet connections (for out-of-band configuration)

Engenio 2882 Enclosure

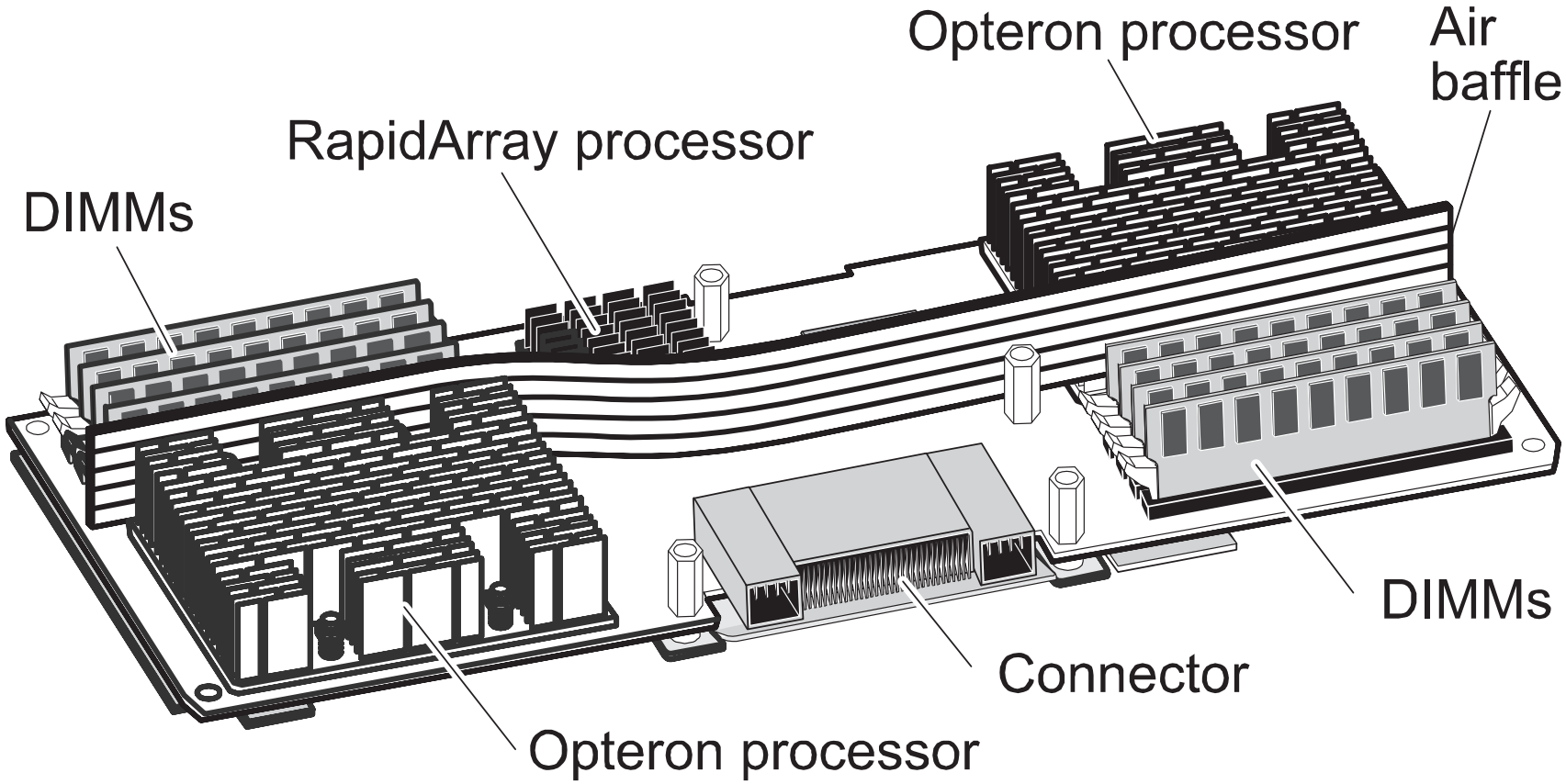
Fibre Channel host connections



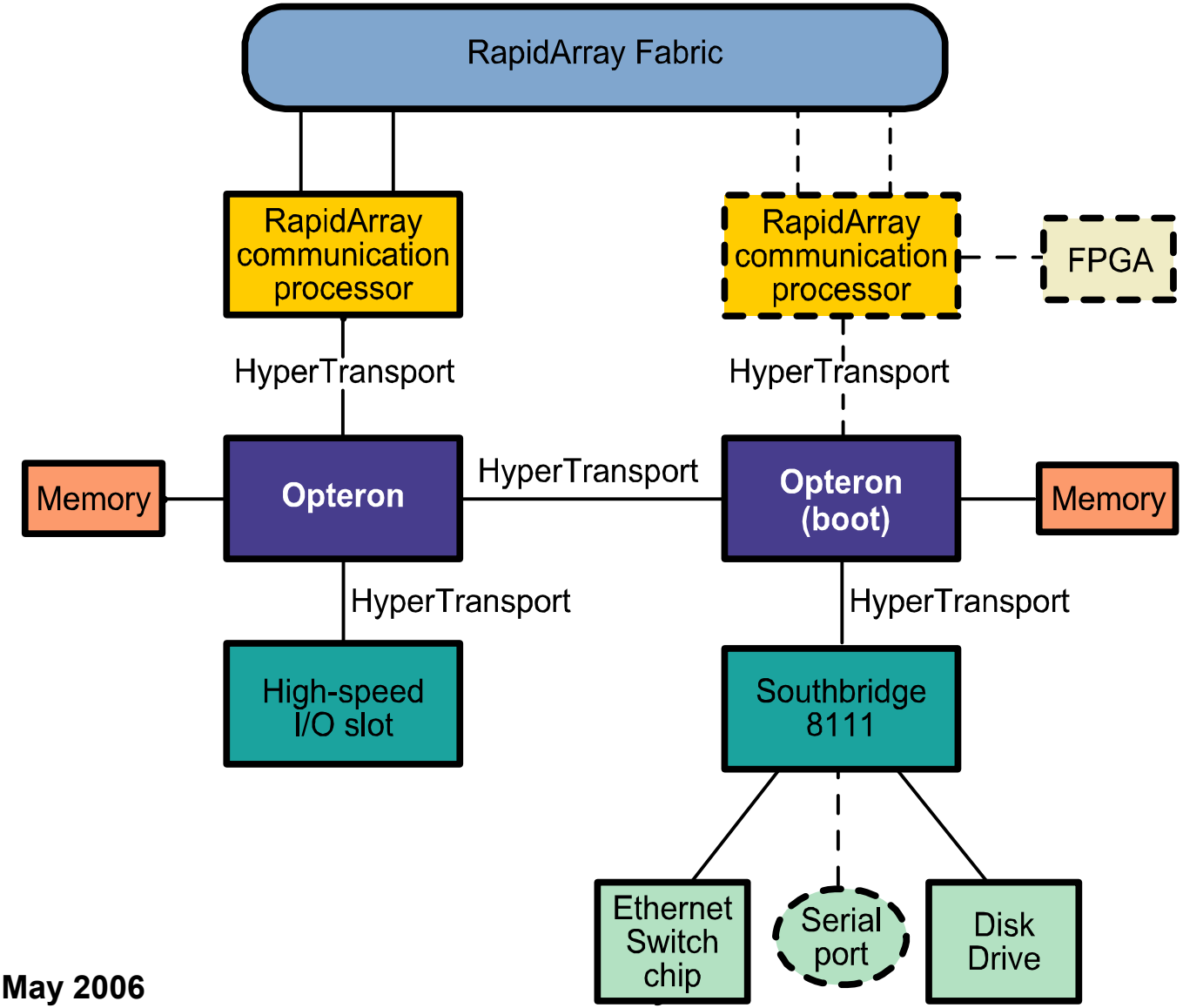
Telnet connections
(for out-of-band configuration)

Compute Blades

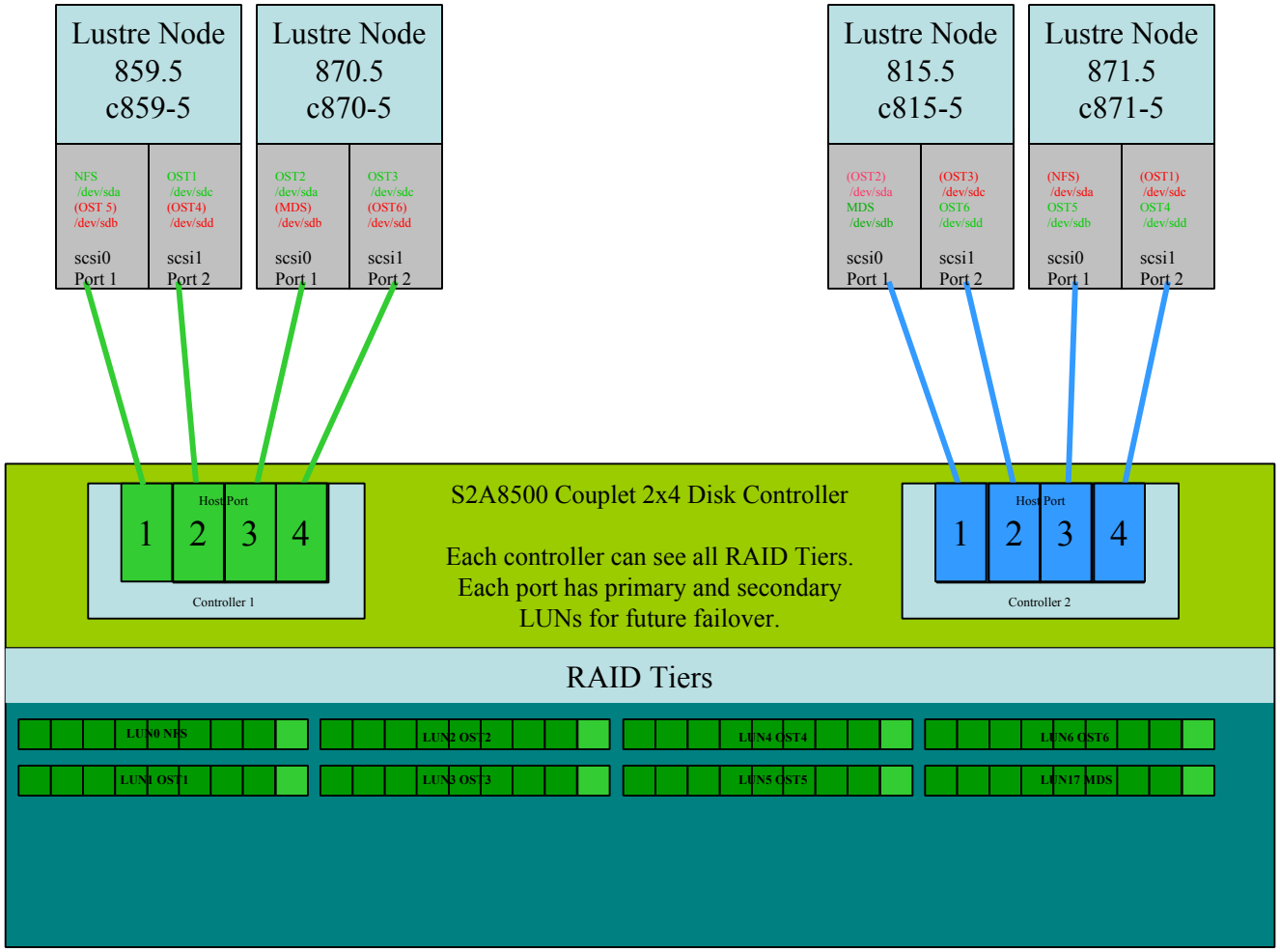
- Each chassis has 6 compute blades; each blade contains 2 Opteron sockets (single or dual core), 8 DIMM slots, and a RapidArray processor



Compute Blade Block Diagram



Sample Cray XD1 Lustre with Planned Failover Paths



The DDN Zoning

```
#### 2/9/2006 ddn1 ##################################
                                        LUN Zoning
  Port    World Wide Name     (External LUN, Internal LUN)
 ---------------------------------------------------------
   1      21000001FF030708     000,000      007,007
   2      22000001FF030708     001,001      006,006
   3      23000001FF030708     002,002      005,005
   4      24000001FF030708     003,003      004,004
#### 2/9/2006 ddn2 ##################################
                                        LUN Zoning
  Port    World Wide Name     (External LUN, Internal LUN)
 ---------------------------------------------------------
   1      21000001FF03071E     000,000      007,007
   2      22000001FF03071E     001,001      006,006
   3      23000001FF03071E     002,002      005,005
   4      24000001FF03071E     003,003      004,004
```

SCSI Detection

```
c859-5:~ # lsscsi
[0:0:0:0]    disk    DDN      S2A 8500         5.20  /dev/sda
[0:0:0:7]    disk    DDN      S2A 8500         5.20  /dev/sdb
[1:0:0:1]    disk    DDN      S2A 8500         5.20  /dev/sdc
[1:0:0:6]    disk    DDN      S2A 8500         5.20  /dev/sdd
c870-5:~ #  lsscsi
[0:0:0:2]    disk    DDN      S2A 8500         5.20  /dev/sda
[0:0:0:5]    disk    DDN      S2A 8500         5.20  /dev/sdb
[1:0:0:3]    disk    DDN      S2A 8500         5.20  /dev/sdc
[1:0:0:4]    disk    DDN      S2A 8500         5.20  /dev/sdd
c871-5:~ # lsscsi
[0:0:0:0]    disk    DDN      S2A 8500         5.20  /dev/sda
[0:0:0:7]    disk    DDN      S2A 8500         5.20  /dev/sdb
[1:0:0:1]    disk    DDN      S2A 8500         5.20  /dev/sdc
[1:0:0:6]    disk    DDN      S2A 8500         5.20  /dev/sdd
c815-5:~ # lsscsi
[0:0:0:2]    disk    DDN      S2A 8500         5.20  /dev/sda
[0:0:0:5]    disk    DDN      S2A 8500         5.20  /dev/sdb
[1:0:0:3]    disk    DDN      S2A 8500         5.20  /dev/sdc
[1:0:0:4]    disk    DDN      S2A 8500         5.20  /dev/sdd
```

Putting it all together

Node	Port	SCSI	LUN	Device	Use
C859-5	1	0:0:0:0	0	/dev/sda	NFS
C859-5	1	0:0:0:7	7	/dev/sdb	OST5 backup
C859-5	2	1:0:0:1	1	/dev/sdc	OST1
C859-5	2	1:0:0:6	6	/dev/sdd	OST4 backup
C870-5	1	0:0:0:2	2	/dev/sda	OST2
C870-5	1	0:0:0:5	5	/dev/sdb	MDS backup
C870-5	2	1:0:0:3	3	/dev/sdc	OST3
C870-5	2	1:0:0:4	4	/dev/sdd	OST6 backup
C815-5	1	0:0:0:2	2	/dev/sda	OST2 backup
C815-5	1	0:0:0:5	5	/dev/sdb	MDS
C815-5	2	1:0:0:3	3	/dev/sdc	OST3 backup
C815-5	2	1:0:0:4	4	/dev/sdd	OST6
C871-5	1	0:0:0:0	0	/dev/sda	NFS backup
C871-5	1	0:0:0:7	7	/dev/sdb	OST5
C871-5	2	1:0:0:1	1	/dev/sdc	OST1 backup
C815-5	2	1:0:0:6	6	/dev/sdd	OST4

The Sample Cray XD1 config.sh file

Part 1 – Global setup

```
#!/bin/bash
# Example lustre_config.xml
config=${1:-lustre_config.xml}
SERVERS="c859-5 c815-5 c870-5 c871-5"
JSIZ='--journal_size 400'    # ext3 journal size
LMC=${LMC:-/usr/sbin/lmc}
STRIPE_BYTES=1048576
if [ -f $config ]
then
  rm -f $config
fi
# Add network node for each server
for s in $SERVERS; do
${LMC} -m $config --add net --node $s --nid $s --nettype
ra --port 988 || exit 1
done
```

The Sample Cray XD1 config.sh file

Part 2 – MDS setup

```
# Add network node for a generic client
${LMC} -m $config --add net --node client --nid '*' --
nettype ra --port 988 || exit 1
# Configure MDS Primary Server
${LMC} -m $config --add mds --mds mds1 --node c815-5
--group mds --fstype ldiskfs --dev /dev/sdb $JSIZ
--failover --timeout 300 || exit 10
# Configure MDS Failover Server
${LMC} -m $config --add mds --mds mds1 --node c870-5
--group mds --fstype ldiskfs --dev /dev/sdb $JSIZ
--failover --timeout 300 || exit 10
# Configure LOV
${LMC} -m $config --add lov --lov lov1 --mds mds1
--stripe_sz $STRIPE_BYTES --stripe_cnt 2 || exit 20
```

The Sample Cray XD1 config.sh file

Part 3 – OST setup

```
# Configure OSTs: Primary and Failover for each, the
# order of the OST's in the following is important
# because this determines the write/read order, the
# OSTs are staggered to get better load balancing.
# The order is OST1,OST6,OST2,OST5,OST3,OST4

${LMC} -m $config --add ost --ost ost1 --node c859-5
--group ost1 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdc $JSIZ || exit 21

${LMC} -m $config --add ost --ost ost1 --node c871-5
--group ost1 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdc $JSIZ || exit 21
```

The Sample Cray XD1 config.sh file

Part 3 – OST setup continued

```
${LMC} -m $config --add ost --ost ost6 --node c815-5
--group ost6 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdd $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost6 --node c870-5
--group ost6 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdd $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost2 --node c870-5
--group ost2 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sda $JSIZ || exit 21
```

The Sample Cray XD1 config.sh file

Part 3 – OST setup continued

```
${LMC} -m $config --add ost --ost ost2 --node c815-5
--group ost2 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sda $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost5 --node c871-5
--group ost5 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdb $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost5 --node c859-5
--group ost5 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdb $JSIZ || exit 21
```

The Sample Cray XD1 config.sh file

Part 3 – OST setup continued

```
${LMC} -m $config --add ost --ost ost3 --node c870-5
--group ost3 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdc $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost3 --node c815-5
--group ost3 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdc $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost4 --node c871-5
--group ost4 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdd $JSIZ || exit 21
${LMC} -m $config --add ost --ost ost4 --node c859-5
--group ost4 --lov lov1 --fstype ldiskfs
--mountfsoptions extents,mballoc --failover
--timeout 300 --dev /dev/sdd $JSIZ || exit 21
```

The Sample Cray XD1 config.sh file

Part 3 – MDS and client mount point setup

```
# Add mount point for mds
${LMC} -m $config --add mtpt --node c815-5
--path /mnt/lustre --mds mds1 --lov lov1
--clientoptions async || exit 30

# Add mountpoint for clients
${LMC} -m $config --add mtpt --node client
--path /mnt/lustre --mds mds1 --lov lov1
--clientoptions async || exit 30
```

Kernel Modules Prior to Starting Lustre

```
ho618n5:~ # lsmod
Module                  Size  Used by
nfsd                  111112  5
exportfs                7296  1 nfsd
mptscsih               39088  0
sg                     43192  0
st                     43428  0
sd_mod                 21824  7
sr_mod                 18980  0
mptbase                47008  1 mptscsih
scsi_mod              133120  5 mptscsih,sg,st,sd_mod,
                                sr_mod
freq_table              5696  0
ipt_LOG                 6784  1
ipt_owner               4672  2
iptable_filter          3840  1
ip_tables              19392  3 ipt_LOG,ipt_owner,
                                iptable_filter
```

Kernel Modules After Starting Lustre

```
c815-5:~ # lsmod
Module                  Size  Used by
llite                 661560  2
mds                   695676  2
lov                   438420  4 llite
osc                   200844  18
mdc                   148908  3 llite
obdfilter             338244  1
fsfilt_ldiskfs         43208  2
ldiskfs               183476  3 fsfilt_ldiskfs
jbd                    75592  2 fsfilt_ldiskfs,ldiskfs
ost                   153820  1
ptlrpc               1212140  7 llite,mds,lov,osc,mdc,
                               obdfilter,ost
obdclass             2750276  9 llite,mds,lov,osc,mdc,
                               obdfilter,
                               fsfilt_ldiskfs,
                               ost,ptlrpc
```

Kernel Modules After Starting Lustre

```
lvfs                    58156  11 llite,mds,lov,osc,
                                  mdc,obdfilter,
                                  fsfilt_ldiskfs,
                                  ost,ptlrpc,obdclass
kranal                 155340  1
portals                125100  2  ptlrpc,kranal
libcfs                 101080  13 llite,mds,lov,osc,
                                  mdc,obdfilter,
                                  fsfilt_ldiskfs,ost,
                                  ptlrpc,obdclass,lvfs,
                                  kranal,portals
autofs                  19904  1
```

Failover

- Clients have both servers in the configuration data
- After disconnect, clients will try both until it succeeds
- What will this mean?
  - Clients are prepared for automatic failover out of the box
  - Server failover is unchanged from current process

Failing Over an OST

- Important: two servers must never simultaneously use the same device
  - If you have set up Lustre to start automatically with an **`init`** script, disable it on the failing server

```
$ insserv -r <init_script_name>
```

  - Use **`amshutdown`** to stop the primary (failing) server

```
$ amshutdown -i <chassis_number.node>
```

  - Start the service on the failover server

```
$ lconf --service=<failed_OST>
  /etc/lustre/lustre_config.xml
```

Failing Back

- The failed node has replaced
  - Start the primary node
  - On the backup node

```
$ lctl
 > cfg_device <failed ost>
 > cleanup failure
 > detach
 .. repeat for other OSTs on this node
 > quit
```

  - On the primary node

```
$ lconf --service=<ost>
  /etc/lustre/lustre_config.xml
$ insserv <init_script_name>
```

General Lustre Discussion

- CUG 2006

Lustre Start and Stop

- Start the OSTs before the MDS
  - To **initialize** Lustre

    `lconf --reformat /etc/lustre/config.xml`
  - To start Lustre

    `lconf /etc/lustre/config.xml`
- Stop the MDS before the OSTs
  - Unmount (**`umount`**) any client mounts
  - Stop Lustre on the MDS and OSSs

    `lconf --cleanup /etc/lustre/config.xml`
  - Other options:

    `-d` is the same as **`--cleanup`**

    `-f` is force

Mount command for clients

- Mount command line

```
mount 8:/nid00008/client -o nettype=cray_kern_nal
   /mnt/lustre
```

- /etc/fstab entry

```
8:/nid00008/client /lus/nid00008 lustre
   nettype=cray_kern_nal,noauto 0 0
```

Using the `df` Command

- Using the standard df command or any of its options may not tell the true story
  - Future release of Lustre will resolve this issue

```
> df -t lustre
Filesys 1K-blocks       Used  Available Use% Mounted
client 4111676960 2735793896 1167022064  71%   /l/n8
client  822335392  494463248  286099944  64%  /l/n36
>
```

Note: The above output from `df` was modified to fit on this slide

Filesystem space from the client view

- To find out more about the filesystem:

```
$ cd /proc/fs/lustre/lov/nid00008_lov-000001001dad4800
$ ls -1
   activeobd
   blocksize
   desc_uuid
   filesfree
   filestotal
   kbytesavail
   kbytesfree
   kbytestotal
   numobd
   stripecount
   stripeoffset
   stripesize
   stripetype
   target_obd
   uuid
```

Filesystem space from the client view

- The file **kbytestotal** contains the filesystem size

```
$ cat kbytestotal
4111676960
```

- The file **kbytesfree** is what is available plus the root reserve

```
$ cat kbytesfree
1542579044
```

- The file **kbytesavail** is what is available to use
  - Differences between free and available are the reserved root space

```
$ cat kbytesavail
1167022064
```

Filesystem space from the client view

- To see the individual OSTs

```
$ cd /proc/fs/lustre/osc/OSC_nid00008_ost1_MNT_client…
```

- Here the **kbytestotal, kbytesfree, and kbytesavail** are for the OST
  - You may discover that one of the OSTs is full
  - A future release will have a better tool, something like

```
$ ????
/lus/nid00008        3921 GB total, 1469 GB free, 5 OSTs
   MDS nid00008      3024 MB cache,   40 MB read ahead
   LOV nid00008         4 KB block, 1024 KB stripe size
   OST ost1    nid00011     784 GB total,  228 GB free
   OST ost3    nid00011     784 GB total,  352 GB free
   OST ost2    nid00032     784 GB total,  332 GB free
   OST ost4    nid00032     784 GB total,  338 GB free
   OST ost5    nid00035     784 GB total,  216 GB free
```

Lustre Configuration

- Common configuration modifications
  - Secondary group support
    - Add **`--group_upcall`** arg to mds **`lmc`** command line, specify path to l_setgroups
  - Reduce root reserved space
    - Add **`--mkfsoptions`** to ost **`lmc`** command line, specify e.g. **`-m 1`** to reduce “wasted” space
    - By default 5% of the OST space is reserved for root
    - Use **`tune2fs`** to adjust the root reserve on an exiting filesystem

Adding OSTs

- OSTs must be added offline
  - Must be added at the end of the list
  - Modify the configuration script
  - Generate a new XML file
  - Make the new OST (**`mkfs`**)
  - On the MDS node

```
$ lconf --write_conf --verbose
  /etc/lustre/config_file.xml
```

  - Start Lustre
- The simplest way to redistribute existing files is "cp"

Lustre Recovery

- When Lustre does not get shutdown cleanly, on the next reboot it enters recovery mode.
- This can take in excess of 45 minutes to complete
- On the MDS check

`/var/log/messages`

`/proc/fs/lustre/mds/nid00008_mds/recovery_status`

Lustre Recovery

```
$ cat recovery_status
  status: COMPLETE
  recovery_start: 1146139145
  recovery_end: 1146139898
  recovered_clients: 2
  unrecovered_clients: 0
  last_transno: 420386998
  replayed_requests: 0
```

Lustre Status

- Checking that lustre is up and running ok
- On each node look at /proc/fs/lustre/devices
  - The MDS node should look something like this:

```
c815-5
 0 UP obdfilter ost6 ost6_UUID 265
 1 UP ost OSS OSS_UUID 3
 2 UP mdt MDT MDT_UUID 3
 3 UP mds mds1 mds1_UUID 264
 4 UP lov lov_mds1
 5 UP osc OSC_c815-5_ost1_mds1
 6 UP osc OSC_c815-5_ost6_mds1
 7 UP osc OSC_c815-5_ost2_mds1
```

Lustre Status cont

- Should look something like this for node with just OSTs

```
c859-5
  0 UP obdfilter ost1 ost1_UUID 265
  1 UP ost OSS OSS_UUID 3
----------------------------------------------------

 c870-5
  0 UP obdfilter ost2 ost2_UUID 265
  1 UP ost OSS OSS_UUID 3
  2 UP obdfilter ost3 ost3_UUID 265
----------------------------------------------------

 c871-5
  0 UP obdfilter ost4 ost4_UUID 265
  1 UP ost OSS OSS_UUID 3
  2 UP obdfilter ost5 ost5_UUID 265
```

File Striping and Inheritance

- The default stripe width is set in the configuration file
- Lustre distributes files across all OSTs
- Users can create files and directories with various striping characteristics
  - New files inherit the striping of the parent directory
  - Striping across more OSTs generally leads to higher peak performance on large files
    - But may not be best for small files
    - Striping to more LUNs may increase risk of failure
  - CANNOT change stripe pattern on existing file
  - CAN change stripe pattern on directory

File Striping and Inheritance

- Without proper controls your filesystem can be ‘out of space’ with most of your LUNs (OSTs) empty
  - User writes a very large file to a single OST

Using `lfs`

- The **`lfs`** command is used to create files or directories with specific striping characteristics
  - The form of the command is:

```
lfs find [--obd <uuid>] [--quiet | --verbose]
  [--recursive] <dir|file>

lfs getstripe <file-name>

lfs setstripe <filename> <stripe-size>
  <start-ost> <stripe-cnt>
```

lfs Command Examples

```
/lus/nid00008> mkdir rick
/lus/nid00008> lfs getstripe rick
OBDS:
0: ost1_UUID ACTIVE
1: ost2_UUID ACTIVE
2: ost3_UUID ACTIVE
3: ost4_UUID ACTIVE
4: ost5_UUID ACTIVE
/lus/nid00008>
```

lfs Command Examples

```
/lus/nid00008> touch rick/one
/lus/nid00008> touch rick/two
/lus/nid00008> lfs getstripe ../rick
OBDS:

...

../rick/one
     obdidx       objid       objid         group
          3     8627206   0x83a406             0
          4     8330251   0x7f1c0b             0
../rick/two
     obdidx       objid       objid         group
          0     8387590   0x7ffc06             0
          1     9040714   0x89f34a             0
/lus/nid00008>
```

lfs Command Examples

```
/lus/nid00008> lfs setstripe rick 0 0 3
/lus/nid00008> touch rick/three
/lus/nid00008> touch rick/four
/lus/nid00008> lfs find rick
...
rick/ default stripe_count:3 stripe_size:0
   stripe_offset:0
   rick/three
       obdidx        objid        objid       group
            0      8387591    0x7ffc07           0
            1      9040715    0x89f34b           0
            2      8586362    0x83047a           0
   rick/four
       obdidx        objid        objid       group
            0      8387592    0x7ffc08           0
            1      9040716    0x89f34c           0
            2      8586363    0x83047b           0
```

lfs find

- Problem: if an OST fails, which files are affected?
- Answer: **`lfs find`** will tell you
- Used to make a list of files (partially) residing on an OST
- Impossible to tell if there was actual data on a failed OST
  - Particularly for a small file

```
lfsfind --recursive --obd ost1-home /databarn
```

Filesystem Checking

- If filesystem damage is suspected, run **`e2fsck`** on the affected nodes (MDS and OSS(s))
  - **`e2fsck`** must be run with the filesystem unmounted
  - Run e2fsck –n first to determine the extent of the problem
  - Make sure your using a Lustre aware version

```
/lus/nid00008> which e2fsck
/sbin/e2fsck
/lus/nid00008> module load e2fs
/lus/nid00008> which e2fsck
/opt/e2fs/1.35/sbin/e2fsck
/lus/nid00008>
```

Filesystem Checking

- You will need to create a database for e2fsck that is accessible from the client
- In most cases, ext3 can't tell that it's corrupt until it's too late
  - ext3 will mark the disk corrupt (read only) if it figures it out
    - Then you won't be able to start until you run e2**fsck**

Filesystem Checking

- `lfsck` has a mode to correct distributed state
  - This involves a fair amount of downtime
  - Run it only if you believe that disk corruptions lost a lot of stripes
  - lfsck is run while the Lustre filesystems are mounted

Lustre Debug

- Something goes wrong – what now?
  - Check with user, job output may contain clues
  - Check console (**`syslog`** messages file) for associated nodes (client, MDS, OSSs)
  - Check for dumped lustre logs (to **`/tmp`** on server nodes, **`/tmp/lustre-log-`**…)
  - If persistent condition, capture thread tracebacks with **`sysrq`**
  - Dump lustre kernel log

Lustre Debug `sysrq`

- On node of interest:

```
# cat /proc/sys/kernel/sysrq
```

  - if value is 0, set to 1 to enable

```
# echo 1 > /proc/sys/kernel/sysrq
```

  - To dump tracebacks to console

```
# echo t > /proc/sysrq-trigger
```

Lustre Debug

- On MDS node:

```
# lctl clear
```

  - run command(s) on clients

```
# lctl debug_kernel SOME_PATH
```

  - file at SOME_PATH contains debug log info

```
# lctl debug_file SOME_PATH OTHER_PATH
```

  - file at OTHER_PATH contains formatted log info

/proc Layout

- **/proc/fs/lustre** contains a directory per device type
  - Each of which contains a directory per device per filesystem
- **/proc/sys/portals**
  - debug and subsystem_debug: control debug logging
- **/proc/sys/lustre**
  - timeout: how long to wait? defaults to 100 seconds

/proc Layout

/proc/sys/portals/debug

- Default value 0x33f0480

D_IOCTL
D_WARNING
D_DLMTRACE
D_ERROR
D_EMERG
D_HA
D_RPCTRACE
D_VFSTRACE
D_CONFIG
D_CONSOLE

/proc Layout

- All are in the **/proc/fs/lustre/osc/device** directories
- **max_dirty_mb**: write path will block after x MB of dirty data
- **max_pages_per_rpc**: 256 maximum (default)
- **max_rpcs_in_flight**: beware of flooding the servers
- Per-OSC **rpc_stats** holds a wealth of information