Cray XT™ Series System Overview

S–2423–20

Record of Revision

Version	*Description*
1.0	December 2004 Draft documentation to support Cray XT3 early-production systems.
1.0	March 2005 Draft documentation to support Cray XT3 limited-availability systems.
1.1	June 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.1, System Management Workstation (SMW) 1.1, and UNICOS/lc 1.1 releases.
1.2	August 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.2, System Management Workstation (SMW) 1.2, and UNICOS/lc 1.2 releases.
1.3	November 2005 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.3, System Management Workstation (SMW) 1.3, and UNICOS/lc 1.3 releases.
1.4	April 2006 Supports Cray XT3 systems running the Cray XT3 Programming Environment 1.4, System Management Workstation (SMW) 1.4, and UNICOS/lc 1.4 releases.
1.5	August 2006 Supports limited availability (LA) release of Cray XT series systems running the Cray XT series Programming Environment 1.5, UNICOS/lc 1.5, and System Management Workstation 1.5 releases.
1.5	October 2006 Supports general availability (GA) release of Cray XT series systems running the Cray XT series Programming Environment 1.5, UNICOS/lc 1.5, and System Management Workstation 1.5 releases.
2.0	May 2007 Supports limited availability (LA) release of Cray XT series systems running the Cray XT series Programming Environment 2.0, UNICOS/lc 2.0, and System Management Workstation 2.0 releases.

2.0	October 2007 Supports general availability (GA) release of Cray XT series systems running the Cray XT series Programming Environment 2.0, UNICOS/lc 2.0, and System Management Workstation 3.0 releases.

Contents

Page

Preface

The information in this preface is common to Cray documentation provided with this software release.

Accessing Product Documentation

With each software release, Cray provides books and man pages, and in some cases, third-party documentation. These documents are provided in the following ways:

CrayDoc
: The Cray documentation delivery system that allows you to quickly access and search Cray books, man pages, and in some cases, third-party documentation. Access this HTML and PDF documentation via CrayDoc at the following locations:

  - The local network location defined by your system administrator
  - The CrayDoc public website: `docs.cray.com`

Man pages
: Access man pages by entering the `man` command followed by the name of the man page. For more information about man pages, see the `man`(1) man page by entering:

```
% man man
```

Third-party documentation
: Access third-party documentation not provided through CrayDoc according to the information provided with the product.

Conventions

These conventions are used throughout Cray documentation:

Convention	Meaning
`command`	This fixed-space font denotes literal items, such as file names, pathnames, man page names, command names, and programming language elements.
variable	Italic typeface indicates an element that you will replace with a specific value. For instance, you may replace *filename* with the name `datafile` in your program. It also denotes a word or concept being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a syntax representation for a command, library routine, system call, and so on.
`...`	Ellipses indicate that a preceding element can be repeated.
`name`(N)	Denotes man pages that provide system and programming reference information. Each man page is referred to by its name followed by a section number in parentheses. Enter: `% man man` to see the meaning of each section number for your particular system.

Reader Comments

Contact us with any comments that will help us to improve the accuracy and usability of this document. Be sure to include the title and number of the document with your comments. We value your comments and will respond to them promptly. Contact us in any of the following ways:

E-mail:
`docs@cray.com`

Telephone (inside U.S., Canada):
1–800–950–2729 (Cray Customer Support Center)

Telephone (outside U.S., Canada):
+1–715–726–4993 (Cray Customer Support Center)

Mail:
Customer Documentation
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120–1128
USA

Cray User Group

The Cray User Group (CUG) is an independent, volunteer-organized international corporation of member organizations that own or use Cray Inc. computer systems. CUG facilitates information exchange among users of Cray systems through technical papers, platform-specific e-mail lists, workshops, and conferences. CUG memberships are by site and include a significant percentage of Cray computer installations worldwide. For more information, contact your Cray site analyst or visit the CUG website at `www.cug.org`.

Introduction [1]

This document provides an overview of Cray XT series systems. The intended audiences are application developers and system administrators. Prerequisite knowledge is a familiarity with the concepts of high-performance computing and the architecture of parallel processing systems.

Note: Functionality marked as deferred in this documentation is planned to be implemented in a later release.

1.1 Cray XT Architecture

Cray XT series supercomputer systems are massively parallel processing (MPP) systems. Cray has combined commodity and open source components with custom-designed components to create a system that can operate efficiently at immense scale.

Cray XT series systems are based on the Red Storm technology that was developed jointly by Cray Inc. and the U.S. Department of Energy Sandia National Laboratories. Cray XT series systems are designed to run applications that require large-scale processing, high network bandwidth, and complex communications. Typical applications are those that create detailed simulations in both time and space, with complex geometries that involve many different material components. These long-running, resource-intensive applications require a system that is programmable, scalable, reliable, and manageable.

The Cray XT series consists of Cray XT3 and Cray XT4 systems. The Cray XT3 and Cray XT4 systems differ in the type and speed of their compute node components.

The major features of Cray XT series systems are:

- Cray XT series systems scale from fewer than 100 to more than 30,000 processors. The ability to scale to such proportions stems from the design of system components:
  - The basic scalable component is the *node*. There are two types of nodes. Service nodes provide support functions, such as managing the user's environment, handling I/O, and booting the system. Compute nodes run user applications.

  - Cray XT series systems use a simple memory model. Every instance of a distributed application has its own processor and local memory. Remote memory is the memory on the nodes running the associated application instances. There is no shared memory.
  - Cray XT series systems use error correction code (ECC) technology to detect and correct multiple bit data transfer errors.
  - The *system interconnection network* connects compute and service nodes. It is the data-routing resource that Cray XT series systems use to maintain high communication rates as the number of nodes increases. Most Cray XT series systems use a full 3D torus network topology; a few configurations use a torus-mesh topology.

- The development environment provides a comprehensive set of compilers, libraries, parallel programming models, debuggers, and performance measurement tools.
- The Cray XT series operating system, UNICOS/lc, is tailored to the requirements of service and compute nodes. A full-featured operating system runs on single-core or dual-core service nodes. All compute nodes run either the CNL compute node operating system or the Catamount (also known as the *quintessential kernel* or *Qk*) compute node operating system.
- Lustre is the Cray XT parallel file system. Lustre scales to thousands of clients and petabytes of data.
- Reliability, availability, and serviceability (RAS) features are designed into the system:
  - Cray XT series system cabinets have only one moving part (a blower that cools the components) and redundant power supplies, reducing the likelihood of cabinet failure.
  - Cray XT series system processor boards (called *blades*) have redundant voltage regulator modules (VRMs) or VRMs with redundant circuitry. VRMs are the solid state components most likely to fail.
  - All Cray XT series nodes are diskless; the availability of a node is not tied to the availability of a moving part.
  - Most RAID subsystems have multiple redundant RAID controllers with automatic failover capability and multiple Fibre Channel connections to disk storage.

- Cray XT series systems provide advanced system management and system administration features:
  - The Cray XT series single-system view (SSV) is a set of operating system features that provide one view of the system, significantly simplifying the administrator's tasks.
  - The Cray RAS and Management System (CRMS) monitors and manages all major Cray XT series system components. The CRMS is independent of computation and service components and has its own network.
  - A single physical Cray XT series system can be split into two or more logical machines, each operating as an independent computing resource. A logical machine must have its own compute partition and a service partition that has its own service nodes, external network connections, and I/O equipment.

    Each logical machine can be booted and dumped independently of the other logical machines. For example, a customer may create a logical machine that has nodes with memory size, processor speed, or number of cores that differ from another logical machine. A job is limited to running within a single logical machine.

    Once booted, a logical machine appears as a normal Cray XT series system to the users, limited to the hardware included for the logical machine.

    The CRMS is common across all logical machines. Because logical machines apply from the system interconnection network layer and up, the CRMS functions continue to behave as a single system for power control, diagnostics, low-level monitoring, and so on.
  - Administrators can boot either CNL or Catamount on compute nodes, but not at the same time unless the system is partitioned. If the system is partitioned, the administrator can boot CNL on one partition and Catamount on another.

    Administrators can reboot one or more compute nodes without rebooting the entire Cray XT series system. This feature allows the administrator to reboot compute nodes that have stopped running.
  - Cray XT series systems use Ethernet link aggregation (also known as *bonding* or *channel bonding*) to increase aggregate bandwidth by combining multiple Ethernet channels into a single virtual channel. Link aggregation can also be used to increase the availability of a link by using other interfaces in the bond when one of the links in that bond fails.

- The Cray I/O testing tool provides a common base for testing the I/O components on Cray products. This tool can be used for high-level diagnostic testing to determine the health of underlying I/O components.
- Cray XT series systems support automated release switching. Administrators can change the version of CRMS used on the System Management Workstation (SMW) and/or the version of UNICOS/lc used when the Cray XT series system is booted.
- Support for native IP is provided as an alternative to the IPPO implementation. Native IP is the default; it is functionally equivalent to IPPO and has significant performance advantages.

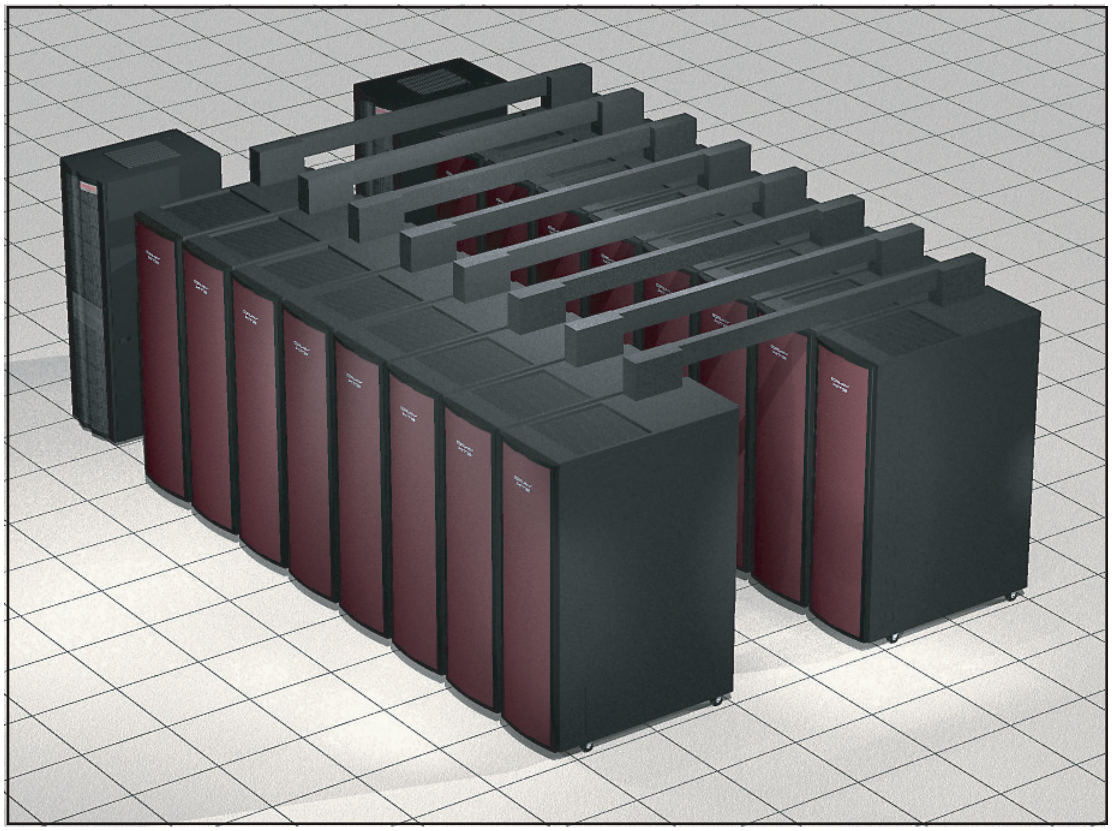

Figure 1. Cray XT Series Supercomputer System

1.2 Related Publications

The Cray XT series system runs with a combination of proprietary, third-party, and open source products, as documented in the following publications.

1.2.1 Publications for Application Developers

- *Cray XT Series System Overview* (this manual)
- *Cray XT Series Programming Environment User's Guide*
- *Cray XT Series Programming Environments Installation Guide*
- *Cray XT Series Software Release Overview*
- *PGI User's Guide*
- *PGI Tools Guide*
- *PGI Fortran Reference*
- PGI compiler commands man pages: `cc`(1), `CC`(1), `ftn`(1), `f77`(1)
- GCC manuals: `http://gcc.gnu.org/onlinedocs/`
- GCC compiler commands man pages: `cc`(1), `CC`(1), `ftn`(1), `f77`(1)
- PathScale manuals: `http://www.pathscale.com/docs.html`
- PathScale compiler commands man pages: `cc`(1), `CC`(1), `ftn`(1)
- Modules utility man pages: `module`(1), `modulefile`(4)
- Commands related to application launch: `aprun`(1), `yod`(1), `cnselect`(1)
- Parallel programming models:
  - Cray MPICH2 man pages (read the `intro_mpi`(3) man page first)
  - Cray SHMEM man pages (read the `intro_shmem`(3) man page first)
  - OpenMP documentation at `http://www.openmp.org/`
- Cray scientific library, XT-LibSci:
  - Basic Linear Algebra Subroutines (BLAS) man pages
  - LAPACK linear algebra man pages
  - ScaLAPACK parallel linear algebra man pages

  - Basic Linear Algebra Communication Subprograms (BLACS) man pages
  - Iterative Refinement Toolkit (IRT) man pages (read the `intro_irt`(3) man page first)
  - SuperLU sparse solver routines (*SuperLU Users' Guide*)
- *AMD Core Math Library (ACML)* manual
- FFTW 2.1.5 and 3.1.1 man pages (read the `intro_fftw2`(3) and/or `intro_fftw3`(3) man page first)
- Portable, Extensible Toolkit for Scientific Computation (PETSc) library, an open source library of sparse solvers. See the `intro_petsc`(3) man page and `http://www-unix.mcs.anl.gov/petsc/petsc-as/index.html`
- I/O buffering (IOBUF) routines. See the `iobuf`(3) man page
- Lustre `lfs`(1) man page
- *PBS Professional 9.0 User's Guide*
- *PBS Pro Release Overview, Installation Guide, and Administration Addendum*
- PBS Pro man pages (`pbs`(1B), `qsub`(1B), and `pbs_resources`(7B))
- TotalView documentation (`http://www.totalviewtech.com/`)
- GNU debugger documentation (see the `xtgdb`(1) man page and the *GDB User Manual* at `http://www.gnu.org/software/gdb/documentation/`).
- PAPI man pages (read the `intro_papi`(3) man page first)
- PAPI manuals (see `http://icl.cs.utk.edu/papi/`)
- *Using Cray Performance Analysis Tools*
- CrayPat man pages (read the `craypat`(1) man page first)
- Cray Apprentice2 man page (`app2`(1))
- *VisIt and SOLD on Catamount* Cray white paper at `http://crinform.cray.com/`
- UNICOS/lc man pages
- SUSE LINUX man pages

- Linux documentation (see the Linux Documentation Project at `http://www.tldp.org` and to SUSE documentation at `http://www.suse.com`)

1.2.2 Publications for System Administrators

- *Cray XT Series System Overview* (this manual)
- *Cray XT Series Software Release Overview*
- *Cray XT Series Programming Environments Installation Guide*
- *Cray XT Series Software Installation and Configuration Guide*
- *Cray XT Series System Management*
- UNICOS/lc man pages
- SUSE LINUX man pages
- CRMS man pages (`xtcli`(8) and `xtgui`(8))
- Lustre documentation (see `http://manual.lustre.org`)
- *PBS Pro Release Overview, Installation Guide, and Administration Addendum*
- *PBS Professional 9.0 Administrator's Guide*
- Linux documentation (see the Linux Documentation Project at `http://www.tldp.org` and SUSE documentation at `http://www.suse.com`).

Hardware Overview [2]

Hardware for the Cray XT series system consists of computation components, service components, the system interconnection network, RAID disk storage systems, and CRMS components. This chapter describes all hardware components except CRMS hardware. For a description of CRMS hardware, see Chapter 4, page 43.

2.1 Basic Hardware Components

The Cray XT series system include the following hardware components:

- AMD Opteron processors
- Dual in-line memory modules (DIMMs)
- Cray SeaStar chips
- System interconnection network
- RAID disk storage subsystems

2.1.1 AMD Opteron Processor

The Cray XT series system supports single-core and dual-core AMD Opteron processors. Opteron processors feature:

- Full support of the x86 instruction set.
- Full compatibility with AMD Socket 940 design (Cray XT3 systems) and AMD Socket AM2 design (Cray XT4 systems).
- Out-of-order execution and the ability to issue a maximum of nine instructions simultaneously.
- Sixteen 64-bit registers and a floating-point unit that support full 64-bit IEEE floating-point operations.
- An integer processing unit that performs full 64-bit integer arithmetic.
- Four 48-bit performance counters that can be used to monitor the number or duration of processor events, such as the number of data cache misses or the time it takes to return data from memory after a cache miss.
- A memory controller that uses error correction code (ECC) for memory protection.

Figure 2 shows the components of a single-core processor.

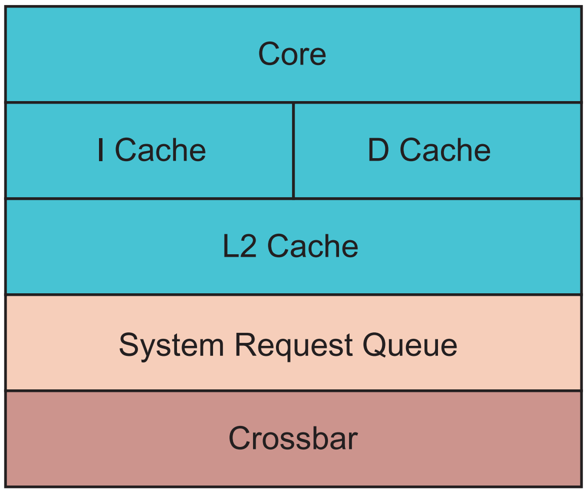

Figure 2. AMD Opteron Single-core Processor

Dual-core processors have two computational engines. Each core (also referred to as a CPU) has its own execution pipeline and the resources required to run without blocking resources needed by other processes. Because dual-core processor systems can run more tasks simultaneously, they can increase overall system performance. The trade-offs are that each core has less local memory bandwidth (because it is shared by the two cores) and less system interconnection bandwidth (which is also shared).

On compute nodes, CNL and Catamount Virtual Node (CVN) support dual-core processors. CNL uses two-way symmetric multiprocessing (SMP). CVN uses SUSE LINUX threading functions for dual-core processing.

On service nodes, UNICOS/lc uses SMP to support dual-core processing.

Figure 3 shows the components of a dual-core processor.

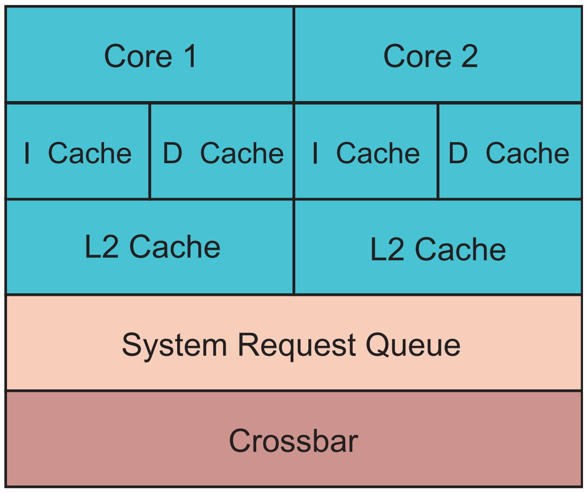

Figure 3. AMD Opteron Dual-core Processor

2.1.2 DIMM Memory

Cray XT series systems support double data rate Dual In-line Memory Modules (DIMMs). Cray XT3 systems include 512 MB, 1 GB, 2 GB, or 4 GB DDR1 DIMMs. Cray XT4 systems include 1 GB or 2 GB DDR2 DIMMs. With four DIMM slots per node, the maximum physical memory is 16 GB per node on Cray XT3 systems and 8 GB per node on Cray XT4 systems. The minimum amount of memory for service nodes is 2 GB.

Cray XT series systems use Error-Correcting Code (ECC) memory protection technology.

2.1.3 Cray SeaStar Chip

The Cray SeaStar application-specific integrated circuit (ASIC) chip is the system's message processor. Cray XT3 systems use SeaStar 1 chips. Cray XT4 systems use SeaStar 2 chips.

SeaStar offloads communications functions from the AMD Opteron processor. A SeaStar chip contains:

- A HyperTransport Link, which connects SeaStar to the AMD Opteron processor.
- A Direct Memory Access (DMA) engine, which manages the movement of data to and from node memory. The DMA engine is controlled by an on-board processor.
- A router, which connects the chip to the system interconnection network. For more information, see Section 2.1.4, page 13.
- A Portals message passing interface, which provides a data path from an application to memory. Portions of the interface are implemented in Cray SeaStar firmware, which transfers data directly to and from user memory without operating system intervention.
- A link to a blade control processor (also known as an *L0 controller*). Blade control processors are used for booting, monitoring, and maintenance. For more information, see Section 4.1.3, page 45.

Figure 4 illustrates the hardware components of the Cray SeaStar chip.

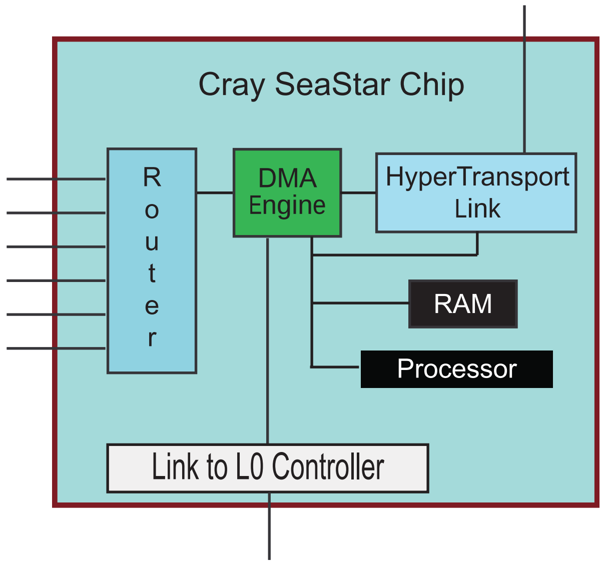

Figure 4. Cray SeaStar Chip

2.1.4 System Interconnection Network

The system interconnection network is the communications center of the Cray XT series system. The network consists of the Cray SeaStar router links and the cables that connect the compute and service nodes.

The network uses a Cray proprietary protocol to provide fast node-to-node message passing and fast I/O to and from a global, shared file system. The network enables the system to achieve an appropriate balance between processor speed and interconnection bandwidth.

2.1.5 RAID Disk Storage Subsystems

Cray XT series systems use two types of RAID subsystems for data storage. System RAID stores the boot image and system files. File system RAID holds root and user files.

Data on system RAID is not globally accessible. Data on file system RAID is globally accessible by default.

2.2 Nodes

Cray XT series processing components combine to form a node. The Cray XT series system has two types of nodes: compute nodes and service nodes. Each node is a logical grouping of a processor, memory, and a data routing resource.

2.2.1 Compute Nodes

Compute nodes run application programs. Each Cray XT3 compute node consists of a single-core or dual-core AMD Opteron processor, DIMM memory, and a Cray SeaStar chip. Each Cray XT4 compute node consists of a dual-core AMD Opteron processor, DIMM memory, and a Cray SeaStar chip.

All compute nodes in a logical system use the same processor type. Because processors are inserted into standard AMD Opteron processor sockets, customers can upgrade nodes as faster processors become available.

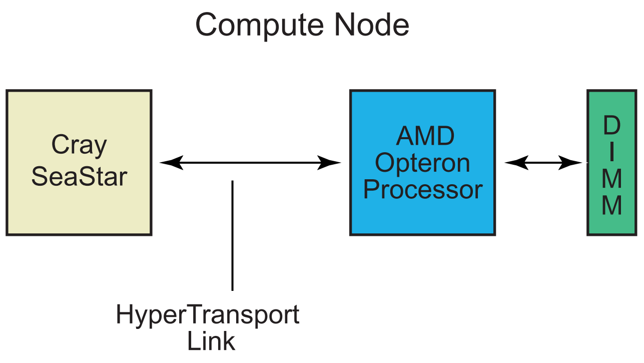

Figure 5. Cray XT Series System Compute Node

2.2.2 Service Nodes

Service nodes handle support functions such as user login, I/O, and network management. Each service node contain a single-core or dual-core processor, DIMM memory, and a SeaStar chip. In addition, each service node contains two PCI-X slots for optional interface cards.

Cray XT series systems include several types of service nodes, defined by the function they perform.

- Login nodes. Users log in to the system through login nodes. Each login node includes one or two PCI-X cards that connect to a user workstation.

  Note: Cray XT series systems also support stand-alone, cross-compiler machines. See the *Cross-compiler for Cray XT Series Systems* manual for details.

- Network service nodes. Each network service node contains a PCI-X card that can be connected to customer network storage devices.
- I/O nodes. Each I/O node uses one or two fibre channel cards to connect to RAID storage.
- Boot nodes. Each system requires one boot node. A boot node contains one fibre channel card and one PCI-X card. The fibre channel card connects to the RAID subsystem, and the PCI-X card connects to the SMW (see Chapter 4, page 43 for further information). Most systems have two boot nodes, a primary and a backup.
- Service database (SDB) node. Each SDB node contains a fibre channel card to connect to the SDB file system. The SDB node manages the state of the Cray XT system.

For a description of the types of service nodes, see Section 3.1, page 19.

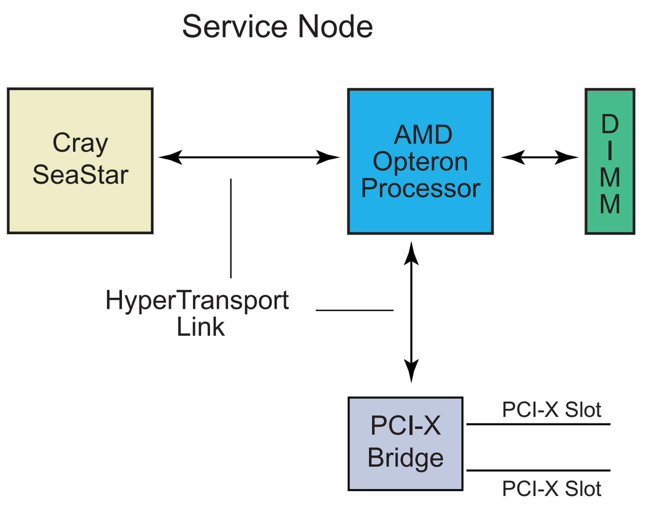

Figure 6. Service Node

2.3 Blades, Chassis, and Cabinets

This section describes the main physical components of the Cray XT series system and their configurations.

2.3.1 Blades

While the node is the logical building block of the Cray XT series system, the basic physical component and field-replaceable unit is the *blade*. There are two types of blades: compute blades and service blades.

A compute blade consists of four compute nodes, voltage regulator modules, and an L0 controller. Each compute blade within a logical machine is populated with AMD processors of the same type and speed and memory chips of the same speed.

The L0 controller is a CRMS component; for more information about CRMS hardware, see Chapter 4, page 43.

A service blade consists of two service nodes, voltage regulator modules, PCI-X cards, and an L0 controller. A service blade has four SeaStar chips to allow for a common board design and to simplify the interconnect configurations. Several different PCI-X cards are available to provide Fibre Channel, GigE, and 10 GigE interfaces to external devices.

2.3.2 Chassis and Cabinets

Each cabinet contains three vertically stacked chassis, and each chassis contains eight vertically mounted blades. A cabinet can contain compute blades, service blades, or a combination of compute and service blades. A single variable-speed blower in the base of the cabinet cools the components.

Customer-provided three-phase power is supplied to the cabinet Power Distribution Unit (PDU). The PDU routes power to the cabinet's power supplies, which distribute 48 VDC to each of the chassis in the cabinet.

All cabinets have redundant power supplies. The PDU, power supplies, and the cabinet control processor (L1 controller) are located at the rear of the cabinet.

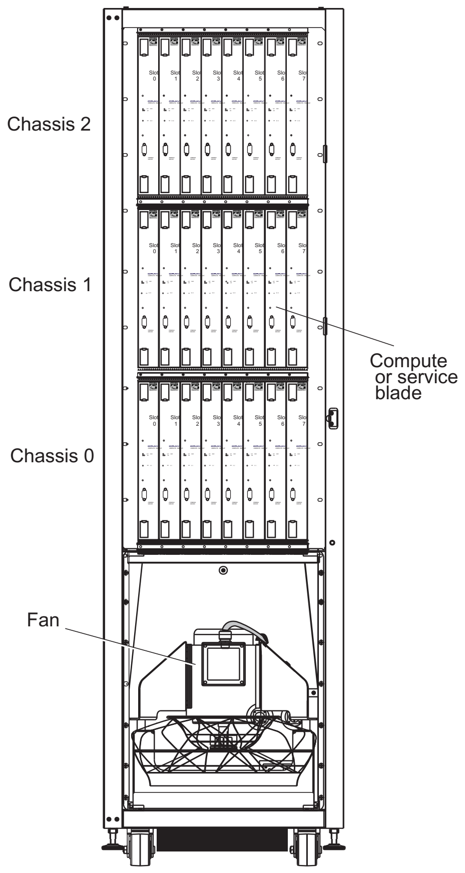

Figure 7. Chassis and Cabinet (front view)

Software Overview [3]

Cray XT series systems run a combination of Cray developed software, third-party software, and open source software. The software is optimized for applications that have fine-grain synchronization requirements, large processor counts, and significant communication requirements.

This chapter provides an overview of the UNICOS/lc operating system, the Lustre file system, the application development environment, and system administration tools. For a description of CRMS software, see Chapter 4, page 43.

3.1 UNICOS/lc Operating System

The Cray XT series operating system is UNICOS/lc. UNICOS/lc is a distributed system of service-node and compute-node components.

Service nodes perform the functions needed to support users, administrators, and applications running on compute nodes. Service nodes run a full-featured version of SUSE LINUX. Above the operating system level are specialized daemons and applications that perform functions unique to each service node. There are five basic types of service nodes: login, network, I/O, boot, and SDB service nodes; see Section 2.2.2, page 14 for further information.

Compute nodes run either the CNL or Catamount operating system.

3.1.1 CNL

CNL is a lightweight compute node operating system. It includes a run time environment based on the SUSE Linux Enterprise Server (SLES) distribution and the SLES kernel with Cray specific modifications. Cray has configured the kernel to eliminate device drivers for hardware not supported on Cray XT series systems.

CNL features:

- Scalability. Only the features required to run high performance computing applications are available on the compute nodes. Other features and services are available from service nodes.
- Minimized OS jitter. Cray has configured and tuned the kernel to minimize processing delays caused by inefficient synchronization.

- Minimized memory footprint. Cray has configured the kernel and the RAMFS-based root file system to use a minimum amount of memory on each node in order to maximize the amount of memory available for applications.
- Application networking (sockets)
- POSIX system calls
- POSIX threads functions

A separate service, the Application Level Placement Scheduler (ALPS), handles application launch, monitoring, and signaling and coordinates batch job processing with PBS Pro.

3.1.2 Catamount

Sandia National Laboratories developed the Catamount kernel to provide support for application execution without the overhead of a full operating system image. The Catamount kernel interacts with an application in very limited ways. This minimal interaction allows for reproducible run times and enables sites to add compute nodes without a corresponding increase in operating system overhead.

Catamount provides virtual memory addressing and physical memory allocation, memory protection, and access to the message-passing layer. Separate entities, yod and process control threads (PCTs), handle application launch and signaling, provide a scalable job loader and the process control tree, and coordinate batch job processing with PBS Pro. Each instance of a distributed application is limited to the amount of physical memory of its assigned compute node; the kernel does not support demand paged virtual memory.

3.2 Lustre File System

I/O nodes host the Lustre file system. Lustre is a high-performance, highly scalable, POSIX-compliant shared file system. Lustre is based on Linux and uses the Portals lightweight message passing API and an object-oriented architecture for storing and retrieving data.

In Catamount, I/O is possible to any file system accessible to `yod`. Lustre I/O is handled as a special case. In CNL, only I/O to Lustre file systems is supported. Files in other remote file systems cannot be accessed. One exception to the CNL restriction is stdin, stdout, and stderr, which is forwarded by `aprun`. Files local to the compute node, such as ones in `/proc` or `/tmp`, can be accessed by a CNL application.

Lustre separates file metadata from data objects. Each instance of a Lustre file system consists of Object Storage Servers (OSSs) and a Metadata Server (MDS). Each OSS hosts one or more Object Storage Targets (OSTs). Lustre OSTs are backed by RAID storage. Applications store data on OSTs, and files can be striped across multiple OSTs.

Lustre's file I/O operations are transparent to the application developer. The I/O functions available to the application developer—Fortran, C, and C++ I/O calls; C and C++ stride I/O calls (`readx`, `writex`, `ireadx`, and `iwritex`); and system I/O calls—are converted to Lustre library calls.

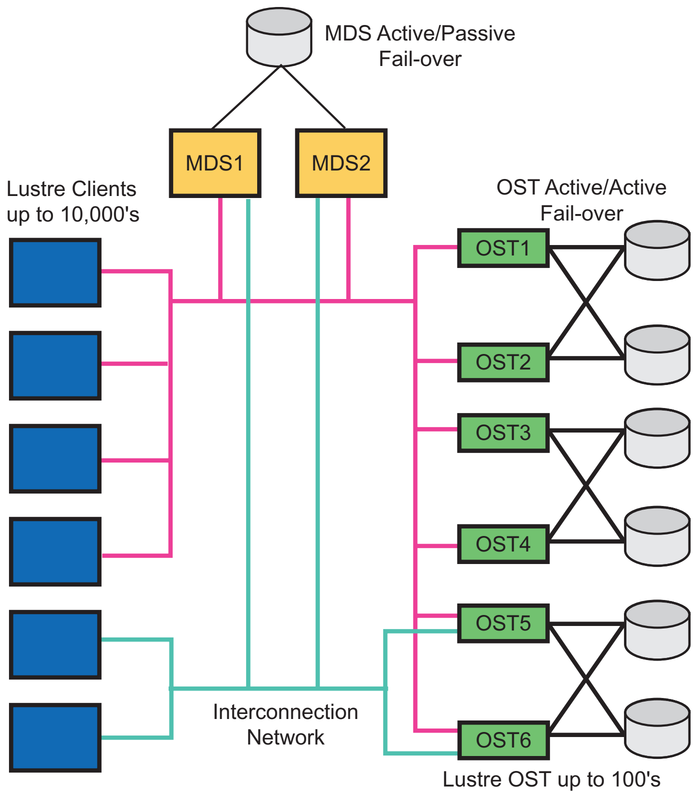
MDS Active/Passive Fail-over
MDS1
MDS2
Lustre Clients up to 10,000's
OST Active/Active Fail-over
OST1
OST2
OST3
OST4
OST5
OST6
Interconnection Network
Lustre OST up to 100's

Figure 8. Lustre Architecture

For a description of Lustre administration, see Section 3.4.3, page 38.

3.3 Development Environment

The Cray XT series application development software is the set of software products and services that programmers use to build and run applications on Cray XT series compute nodes.

3.3.1 User Environment

The user environment is similar to the environment on a typical Linux workstation. Users log in to either a Cray XT series login node or a stand-alone, cross-compiler machine. This manual describes a login node user environment; for information about the cross compiler environment, see the *Cray XT Series Programming Environments Installation Guide* manual.

Before starting to develop applications, the user:

1. Sets up a secure shell. The Cray XT series system uses `ssh` and `ssh`-enabled applications for secure, password-free remote access to login nodes. Before using `ssh` commands, the user needs to generate an RSA authentication key.
2. Loads the appropriate modules. The Cray XT series system uses the Modules utility to support multiple versions of software, such as compilers, and to create integrated software packages. As new versions of the supported software become available, they are added automatically to the Programming Environment, and earlier versions are retained to support legacy applications. By specifying the module to load, the user can choose the default or another version of one or more Programming Environment tools.

For details, see the *Cray XT Series Programming Environment User's Guide* and the `module`(1) and `modulefile`(4) man pages.

3.3.2 Compiling Programs

The Cray XT series system Programming Environment includes C, C++, and Fortran compilers from The Portland Group (PGI), the GNU Compiler Collection (GCC), and PathScale.

The compilers translate C, C++, and Fortran source programs into Cray XT series object files. Interlanguage communication functions enable developers to create Fortran programs that call C or C++ routines and C or C++ programs that call Fortran routines.

The command used to invoke a compiler is called a *compilation driver*; it can be used to apply options at the compilation unit level. Fortran directives and C or C++ pragmas apply options to selected portions of code or alter the effects of command-line options.

Before compiling programs, users need to make sure the target architecture is set correctly. The target architecture is used by the compilers and linker in creating executables to run on either CNL or Catamount compute nodes. The target architecture is not the operating system currently running on compute nodes; that is determined by the system administrator at boot time. For details, see *Cray XT Series Programming Environment User's Guide*.

In addition to the PGI, GCC, and PathScale compilers, the Cray XT Programming Environment includes the Java compiler for developing applications to run on service nodes. For details, see `http://java.sun.com/javase/6/docs/`.

3.3.2.1 PGI Compiler Commands

The following PGI compiler commands are available:

PGI Compiler	**Command**
C	`cc`
C++	`CC`
Fortran 90/95	`ftn`
FORTRAN 77	`f77`

Note: Users should not invoke a PGI compiler directly using the `pgcc`, `pgCC`, `pgf95`, or `pgf77` command. The resulting executable will not run on the Cray XT series system.

The `cc`(1), `CC`(1), `ftn`(1), and `f77`(1) man pages contain information about the compiler driver commands, while the `pgcc`(1), `pgCC`(1), `pgf95`(1), and `pgf77`(1) man pages describe the PGI compiler command options.

For further information, see the *Cray XT Series Programming Environment User's Guide*.

3.3.2.2 GCC Compiler Commands

The Programming Environment includes the 4.1.1 and 3.2.3 versions of GCC. GCC 4.1.1 includes the gfortran compiler, and GCC 3.2.3 includes the g77 compiler. The following GCC compiler commands are available:

GCC Compiler	**Command**
C	`cc`
C++	`CC`
Fortran 90/77 (GCC 4.1.1)	`ftn`
FORTRAN 77 (GCC 3.2.3)	`f77`

Note: Users should not invoke a GCC compiler directly using the `gcc`, `g++`, `gfortran`, or `g77` command. The resulting executable will not run on the Cray XT series system.

The `cc`(1), `CC`(1), `ftn`(1), and `f77`(1) man pages contain information about the compiler driver commands, while the `gcc`(1), `g++`(1), `gfortran`(1), and `g77`(1) man pages describe the GCC compiler command options.

For further information, see the *Cray XT Series Programming Environment User's Guide*.

3.3.2.3 PathScale Compiler Commands

The following PathScale compiler commands are available:

PathScale Compiler	**Command**
C	`cc`
C++	`CC`
Fortran 90/95/77	`ftn`

Note: Users should not invoke a PathScale compiler directly using the `pathcc`, `pathCC`, or `path95` command. The resulting executable will not run on the Cray XT series system.

The `cc`(1), `CC`(1), and `ftn`(1) man pages contain information about the compiler driver commands, while the `pathcc`(1), `pathCC`(1), `path95`(1), and `eko`(7) man pages contain descriptions of the PathScale compiler command options.

For further information, see the *Cray XT Series Programming Environment User's Guide*.

3.3.3 Using Library Functions

Developers can use C, C++, and Fortran library functions and functions from the following libraries:

- GNU C Language Runtime Library (glibc) functions. More glibc functions are supported in CNL than Catamount. For details, see the *Cray XT Series Programming Environment User's Guide*.
- Cray MPICH2, Cray SHMEM, and OpenMP functions. MPICH2 and SHMEM use Portals functions for message passing; the Portals interface is transparent to the application programmer.

  MPICH2 is an implementation of MPI-2 by the Argonne National Laboratory Group. The dynamic process (spawn) functions in Cray MPICH2 are not supported at this time, but otherwise the libraries are fully MPI 2.0 compliant.

  Cray SHMEM routines are similar to the Cray MPICH2 routines; they pass data between cooperating parallel processes. Cray SHMEM routines can be used in programs that perform computations in separate address spaces and that explicitly pass data to and from different processing elements in the program. The Fortran module for Cray SHMEM is not supported. Developers should use the `INCLUDE mpp/shmem.fh` statement instead.

  OpenMP is an industry-standard, portable model for shared memory parallel programming. In addition to library routines, OpenMP provides Fortran directives and C and C++ pragmas. The PGI, GCC, and PathScale compilers support OpenMP.

  OpenMP applications can be used in hybrid MPI/OpenMP applications but cannot cross node boundaries. OpenMP is supported on CNL but not Catamount. For further information, see the *Cray XT Series Programming Environment User's Guide* and the *OpenMP Application Program Interface* at `http://www.openmp.org/`.
- Cray XT LibSci scientific libraries. XT-LibSci contains:
  - Basic Linear Algebra Subroutines (BLAS)
  - LAPACK linear algebra routines
  - ScaLAPACK parallel linear algebra routines

  - Basic Linear Algebra Communication Subprograms (BLACS), a set of communication routines used by ScaLAPACK and the user to set up a problem and handle the communications.
  - Iterative Refinement Toolkit (IRT), a library of factorization routines, solvers, and tools that can be used to solve systems of linear equations more efficiently than the full-precision solvers in Cray XT-LibSci or ACML.
  - SuperLU, a set of routines that solve large, sparse, nonsymmetric systems of linear equations. XT-LibSci library routines are written in C but can be called from Fortran, C, or C++ programs.

- 64-bit AMD Core Math Library (ACML), which includes:
  - A suite of Fast Fourier Transform (FFT) routines for single-precision, double-precision, single-precision complex, and double-precision complex data types.
  - Fast scalar, vector, and array math transcendental library routines optimized for high performance.
  - A comprehensive random number generator suite.

- The Programming Environment includes the 2.1.5 and 3.1.1 releases of FFTW. FFTW is a C subroutine library with Fortran interfaces for computing the discrete Fourier transform in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, such as the discrete cosine/sine transforms). The Fast Fourier Transform (FFT) algorithm is applied for many problem sizes. Distributed memory parallel FFTs are available only in FFTW 2.1.5.

  For further information, see the `intro_fftw2`(3) and `intro_fftw3`(3) man pages.

- Portable, Extensible Toolkit for Scientific Computation (PETSc), an open source library of sparse solvers.

- IOBUF, an optional I/O buffering library that can reduce the I/O wait time for Catamount programs that read or write large files sequentially. IOBUF intercepts standard I/O calls, such as read and open, and replaces the stdio (glibc, libio) layer of buffering with an additional layer of buffering, thus improving program performance by enabling asynchronous prefetching and caching of file data. IOBUF can also gather run time statistics and print a summary report of I/O activity for each file.

  IOBUF is not needed for CNL jobs.

3.3.4 Linking Applications

After correcting compilation errors, the developer again invokes the compilation driver, this time specifying the application's object files (`*filename*.o`) and libraries (`*filename*.a`) as required.

The linker extracts the library modules that the program requires and creates the executable file (named `a.out` by default).

3.3.5 Running Applications

There are two methods of running applications: interactively and through application launch commands in batch job scripts. The user can run an application interactively by:

- Using the `aprun` or `yod` command. The `aprun` command launches applications on compute nodes running CNL. The `yod` command launches applications on compute nodes running Catamount.
- Using the PBS Pro `qsub -I` command to initiate an interactive session and then running an application interactively using the `aprun` or `yod` command.

The developer uses PBS Pro to run batch jobs. PBS Pro is a networked subsystem for submitting, monitoring, and controlling a workload of batch jobs. A batch job is typically a shell script and attributes that provide resource and control information about the job. Batch jobs are scheduled for execution at a time chosen by the subsystem according to a defined policy and the availability of resources.

A PBS Pro server maintains job queues. Each queue holds a group of jobs available for execution and each job has a set of user-specified resource requirements.

The PBS Pro scheduler is the policy engine that examines the ready-to-run jobs and selects the next job to run based on a set of criteria. Once the required compute nodes are reserved, PBS Pro processes the job script and transfers control to yod or aprun.

Note: If your site has installed another batch system, please contact the appropriate vendor for the necessary installation, configuration, and administration information. For example, contact Cluster Resources, Inc. (`http://www.clusterresources.com/`) for documentation specific to Moab products.

Developers who want to launch an application only on nodes with certain attributes can use the `cnselect` command. Among the selectable node attributes are node ID, OS class (CNL or Catamount), number of cores (single or dual), amount of node memory, page size, and CPU clock rate.

The `cnselect` utility uses the service database (SDB) to generate a candidate list of nodes. Developers can include this list on `aprun` or `yod` commands to launch applications on compute nodes with those attributes.

3.3.5.1 Running Applications Interactively on CNL

To run an interactive job under CNL, the user enters the `aprun` command, specifying the executables that are part of the job and the number of processors or number and candidate list of processors on which they will run.

The `aprun` client sends the request to the apsys server, which forwards it to the `apsched` agent running on a service node. The `apsched` agent gets the compute node placement list, reserves the nodes needed for the application, and relays the placement list to `aprun`. The `aprun` client then sends the placement list and the executable binary data to the `apinit` daemon running on the first node assigned to the application.

On each node allocated to the application, an apinit daemon creates an application shepherd to manage the processes of the application on that node. The application shepherd on the first assigned node propagates the node placement list and the executable to all other nodes allocated to the job and passes control to the application.

While the application is running, application shepherds monitor the application processes. If the `aprun` client or an application instance catches a signal, the signal is propagated to all processing elements. Applications running under CNL rely on `aprun` to manage the stdin, stdout, and stderr streams and handle signal management.

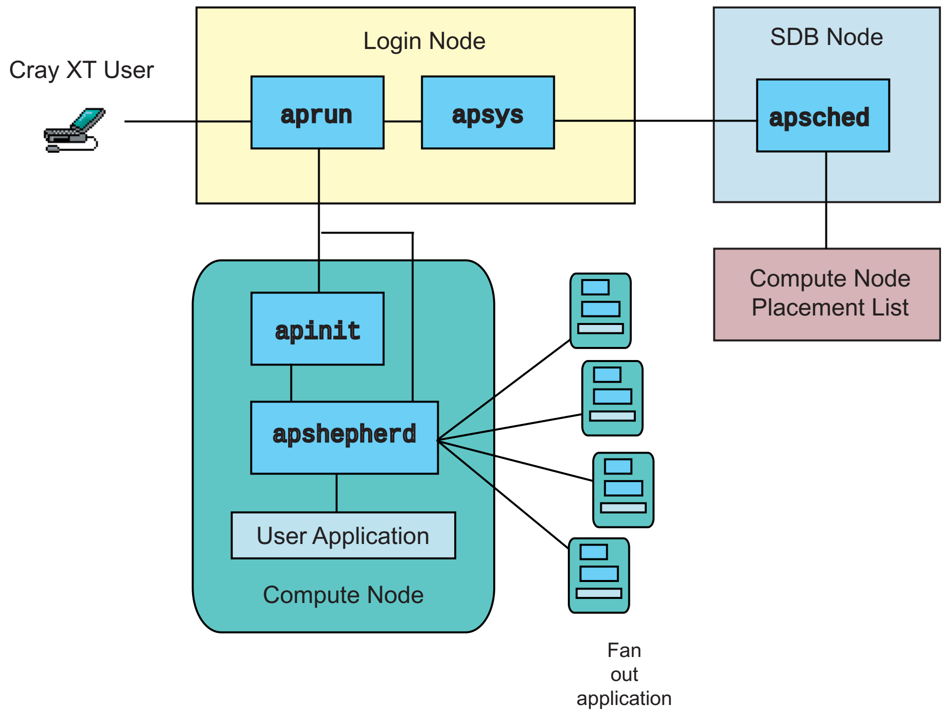

Figure 9. Running Applications Interactively on CNL

3.3.5.2 Running Batch Jobs on CNL

The Cray XT series system uses PBS Pro to launch batch jobs running under CNL. The user creates a script containing `aprun` commands, then enters the PBS Pro `qsub` command to submit the job to a PBS Pro server.

The PBS Pro server uses the apbasil interface to reserve the nodes required for the job, then processes the job scripts. When PBS Pro encounters the `aprun` command(s) in the script, control is transferred to ALPS for application propagation and launch.

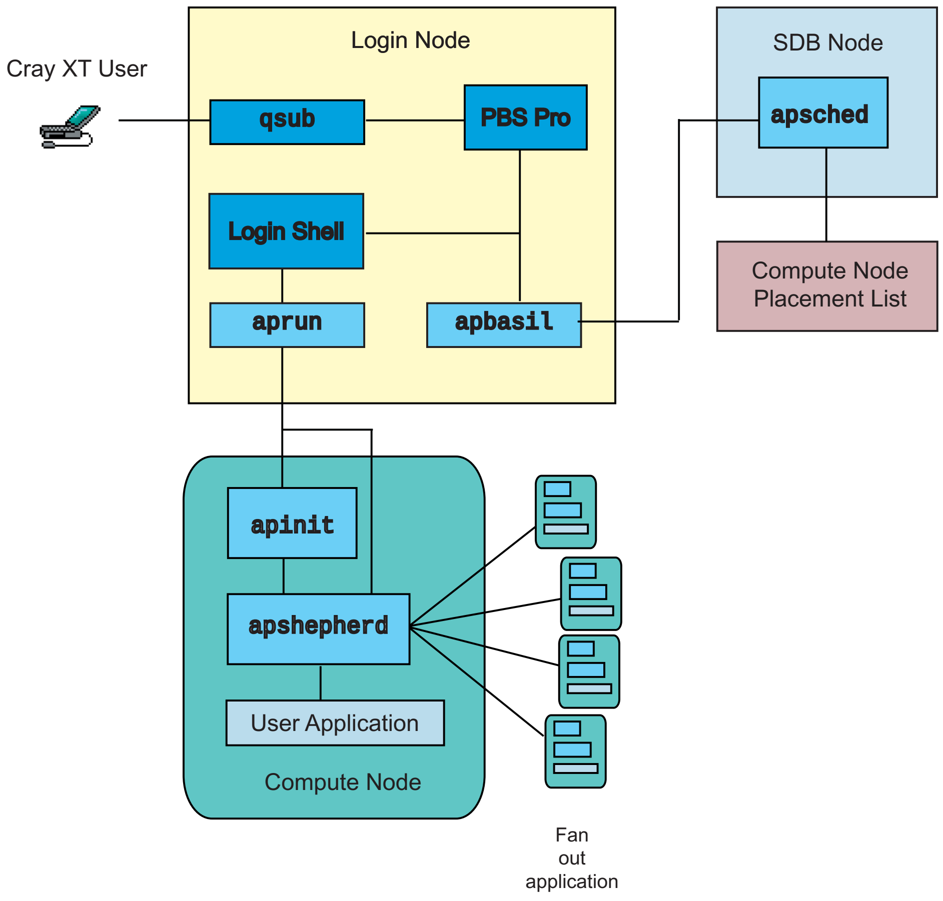

Figure 10. Running Batch Jobs on CNL

3.3.5.3 Running Applications Interactively on Catamount

To run an application interactively on Catamount, the user enters the `yod` command, specifying all executables that are part of the job and the number of processors or number and candidate list of processors on which they will run. The `yod` utility uses the Compute Processor Allocator (CPA) to get a list of nodes on which to run the job. It then contacts the process control thread (PCT) for the first node on the list. After connecting with the PCT, yod maps the executables into its address space and sets up a memory region for the PCT to fetch the executable.

In response to the load request, the PCT propagates the executable to all compute nodes selected to run that program. After the first PCT has finished loading the executable, it maps its memory for load propagation, contacts the next set of PCTs, and forwards the load request. This process is repeated until all compute nodes have loaded. After all application images have been loaded, the PCT starts the executable.

On dual-core processor systems, there is only one instance of the kernel and the PCT per node. The PCT runs only on the "master" CPU, though parts of Catamount do run on the "subordinate" CPU. The PCT is able to schedule user processes on either core.

While the application is running, yod manages the stdin, stdout, and stderr streams; manages signals and system call forwarding; and participates in cleanup when the application terminates.

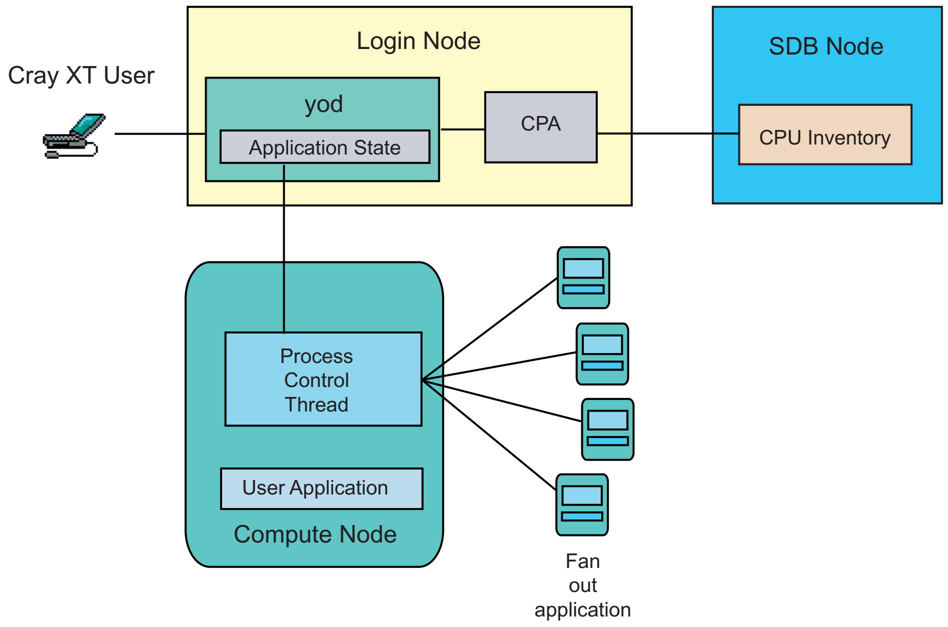

Figure 11. Running Applications Interactively on Catamount

3.3.5.4 Running Batch Jobs on Catamount

The Cray XT series system uses PBS Pro to launch batch jobs running under Catamount. The user creates a script containing the `yod` command(s) to run the application, then enters the PBS Pro `qsub` command to submit the job to a PBS Pro server. PBS Pro uses the catnip interface to reserve the nodes required for the job, then processes the job scripts. When PBS Pro encounters the `yod` command(s) in the script, control is transferred to yod for application propagation and launch.

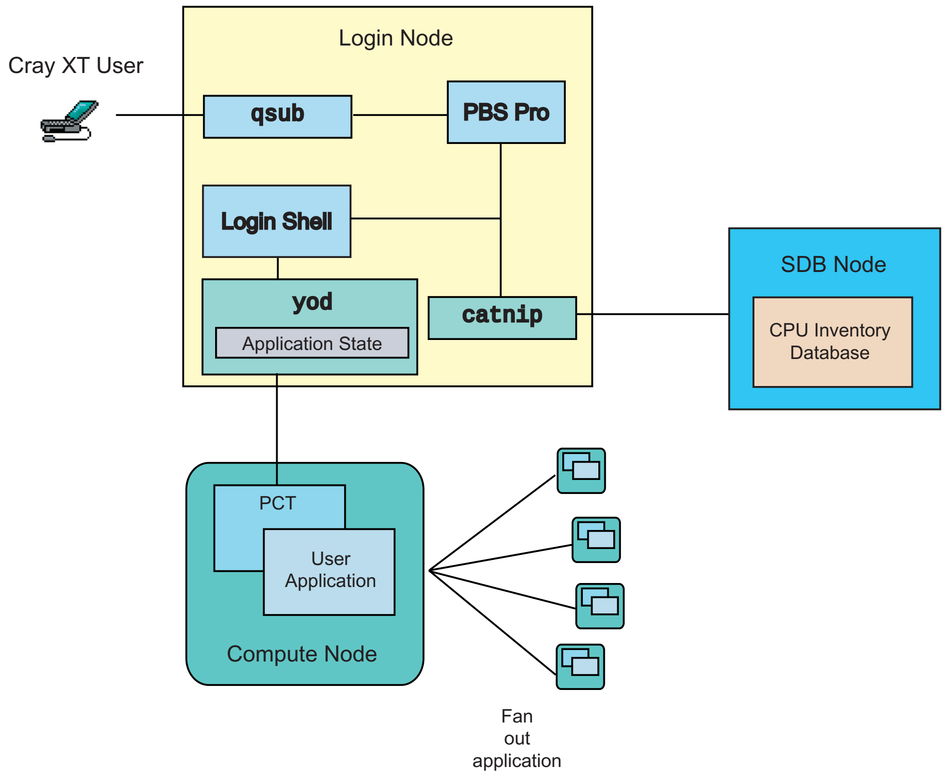

Figure 12. Running Batch Jobs on Catamount

3.3.6 Debugging Applications

The Cray XT series system supports the TotalView debugger for single-process and mutiprocess debugging and the GNU debugger, `xtgdb`, for single-process debugging.

The TotalView debugger, available from TotalView Technologies, LLC, provides source-level debugging of applications. It is compatible with the PGI, GCC, and PathScale C, C++, and Fortran compilers.

TotalView can debug applications running on 1 to 4096 compute nodes, providing visibility and control into each process individually or by groups. It also supports access to MPI-specific data, such as the message queues.

TotalView typically is run interactively. To debug a program using TotalView, the developer invokes TotalView either from the graphical interface (`totalview`) or the command line (`totalviewcli`). TotalView parses the command to get the number of nodes requested, then makes a node allocation request. TotalView directs `aprun` or `yod` to load but not start the application. The `aprun` or `yod` utility loads the application onto the compute nodes, after which TotalView can perform initial setup before instructing `aprun` or `yod` to start the application.

For more information about TotalView, see the *Cray XT Series Programming Environment User's Guide*, the `totalview`(1) man page, and TotalView documentation at `http://www.totalviewtech.com/Documentation/`. To find out what levels of the compilers TotalView supports, see the *TotalView Platforms and System Requirements* document at the TotalView website.

The developer can use `xtgdb` to debug single-process programs running under CNL; `xtgdb` is not supported on Catamount. For more information about `xtgdb`, see the `xtgdb`(1) man page and gdb documentation at `http://www.gnu.org/software/gdb/gdb.html`.

3.3.7 Monitoring and Managing Applications

For CNL applications, ALPS provides commands for monitoring and managing applications. The `apstat` command reports the status of applications, including the number of processing elements, number of threads, a map of the application's address space, and a map showing the placement of team members. The `apkill` command sends a kill signal to an application team.

For more information, see the `apstat`(1) and `apkill`(1) man pages.

3.3.8 Measuring Performance

The Cray XT series system provides tools for collecting, displaying, and analyzing performance data.

3.3.8.1 Performance API

The Performance API (PAPI) from the University of Tennessee and Oak Ridge National Laboratory is a standard interface for accessing hardware performance counters. A PAPI event set maps AMD Opteron processor hardware counters to a list of events, such as Level 1 data cache misses, data translation lookaside buffer (TLB) misses, and cycles stalled waiting for memory accesses. Developers can use the API to collect data on those events.

3.3.8.2 CrayPat

CrayPat is a performance analysis tool. The developer can use CrayPat to perform sampling and tracing experiments on an instrumented application and analyze the results of those experiments.

Sampling experiments capture information at user-defined time intervals or when a predetermined event occurs, such as the overflow of a user-specified hardware performance counter. Tracing experiments capture information related to both predefined and user-defined function entry points, such as the number of times a particular MPI function is called and the amount of time the program spends performing that function.

The developer uses the `pat_build` command to instrument programs. No recompilation is needed to produce the instrumented program. Alternatively, the developer can use the `pat_hwpc` command to instrument the program for collecting predefined hardware performance counter information, such as cache usage data.

After instrumenting a program, the developer sets environment variables to control run time data collection, runs the instrumented program, then uses the `pat_report` command to either generate a report or export the data for use in other applications.

3.3.8.3 Cray Apprentice2

Cray Apprentice2 is an interactive X Window System tool for displaying performance analysis data captured during program execution.

Cray Apprentice2 identifies many potential performance problem areas, including the following conditions:

- Load imbalance
- Excessive serialization
- Excessive communication
- Network contention
- Poor use of the memory hierarchy
- Poor functional unit use

Cray Apprentice2 has the following capabilities:

- It is a post-execution performance analysis tool that provides information about a program by examining data files that were created during program execution. It is not a debugger or a simulator.
- Cray Apprentice2 displays many types of performance data contingent on the data that was captured during execution.
- It reports time statistics for all processing elements and for individual routines.
- It shows total execution time, synchronization time, time to execute a subroutine, communication time, and the number of calls to a subroutine.

3.3.9 Optimizing Applications

The PGI, GCC, and PathScale compilers provide compiler command options, directives, and pragmas that the developer can use to optimize code.

For further information, see the PGI compiler documentation at `http://www.pgroup.com`, the GCC compiler documentation at `http://gcc.gnu.org/`, or the PathScale compiler documentation at `http://www.pathscale.com/docs.html`.

In addition, see the *Software Optimization Guide for AMD64 Processors* at `http://www.amd.com/`.

3.3.10 Using Data Visualization Tools

Cray XT series systems support the VisIt data visualization and graphical analysis tool. VisIt was developed by the Department of Energy (DOE) Advanced Simulation and Computing Initiative (ASCI) and is distributed through Lawrence Livermore National Laboratory (`http://www.llnl.gov/visit`).

VisIt provides an extensible interface for creating, manipulating, and animating 2D and 3D graphical representations of data sets ranging in size from kilobytes to terabytes.

VisIt can be run on CNL or Catamount nodes. To run VisIt on Catamount nodes, sites install a socket off-load (SOLD) facility. For more information about configuring and using VisIt with the SOLD facility, see the *VisIt and SOLD on Catamount* Cray white paper.

3.4 System Administration

The system administration environment provides the tools that administrators use to manage system functions, view and modify the system state, and maintain system configuration files. System administration components are a combination of Cray XT series system hardware, SUSE LINUX, Lustre, and Cray XT series system utilities and resources.

Note: For information about standard SUSE LINUX administration, see `http://www.tldp.org` or `http://www.suse.com`. For details about Lustre functions, see the *Cray XT Series System Management* manual and `http://manual.lustre.org/`.

Many of the components used for system administration are also used for system monitoring and management (such as powering up and down and monitoring the health of hardware components). For details, see Chapter 4, page 43.

3.4.1 System Management Workstation

The System Management Workstation (SMW) is a server and display that provides a single-point interface to an administrator's environment. The SMW provides a terminal window from which the administrator performs tasks like adding user accounts, changing passwords, and monitoring applications. The SMW accesses system components through the administration / CRMS network; it does not use the system interconnection network.

3.4.2 Shared-root File System

The Cray XT series system has a shared-root file system where the root directory is shared read-only on the service nodes. All nodes have the same default directory structure. However, the `/etc` directory is specially mounted on each service node as a node-specific directory of symbolic links. The administrator can change the symbolic links in the `/etc` directory by the process of specialization, which changes the symbolic link to point to a non-default version of a file. The administrator can specialize files for individual nodes or for a class (type) of nodes.

The administrator's interface includes commands to view file layout from a specified node, determine the type of specialization, and create a directory structure for a new node or class based on an existing node or class. For details, see the *Cray XT Series System Management* manual.

3.4.3 Lustre File System Administration

Lustre is optimized for large-scale, serial access typical of parallel programming applications.

When a file is created, the client contacts a metadata server (MDS), which creates an inode for the file. The inode holds metadata rather than references to the actual data. The MDS handles namespace operations, such as opening or closing a file, managing directory listings, and changing permissions. The MDS contacts Object Storage Targets (OSTs) to create data objects. The OSTs handle block allocation, enforce security for client access, and perform parallel I/O operations to transfer file data. The administrator can create and mount more than one instance of Lustre. One MDS plus one or more OSTs make up a single instance of Lustre and are managed as such.

Objects allocated on OSTs hold the data associated with the file. Once a file is created, read and write operations take place directly between the client and the OST, bypassing the MDS. The OSTs use the ldiskfs file system, a modified version of the ext3 file system, for backend storage. These file systems are used to store Lustre file and metadata objects and are not directly visible to the user. Lustre does not support building a parallel file system with NFS file systems.

The administrator configures Lustre as a parallel file system by creating multiple object storage targets (OSTs). The file system optimizes I/O by striping files across many RAID storage devices.

The administrator can configure a default system-wide striping pattern at file system creation time. The administrator specifies:

- The default number of bytes stored on each OST
- The default number of OSTs across which each file is striped

The Lustre lfs utility enables the developer to:

- Create a file with a specific striping pattern
- Find the striping pattern of existing files

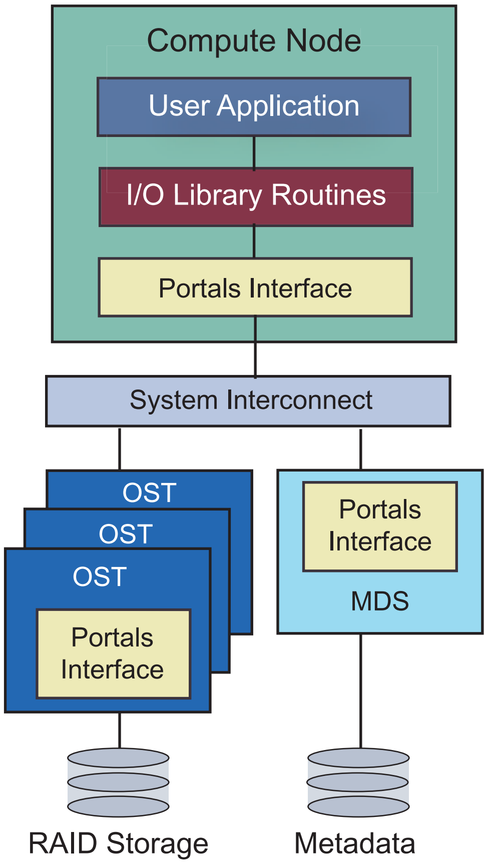

Figure 13. Lustre File System

Lustre configuration information is maintained in the service database (SDB). For details, see the *Cray XT Series System Management* manual and the *Cray XT Series Software Installation and Configuration Guide*.

3.4.4 Configuration and Source Files

The administrator use the boot node to view files, maintain configuration files, and manage the processes of executing programs. Boot nodes connect to the SMW and are accessible through a login shell.

The `xtopview` utility runs on boot nodes and allows the administrator to view files as they would appear on any node. The `xtopview` utility also maintains a database of files to monitor as well as file state information such as checksum and modification dates. Messages about file changes are saved through a Revision Control System (RCS) utility.

3.4.5 System Log

Once the system is booted, console messages are sent to the system log and are written to the boot RAID system. System log messages generated by service node kernels and daemons are gathered by syslog daemons running on all service nodes. Kernel errors and panic messages are sent directly to the SMW.

The administrator can configure the syslog daemon to write the messages to different files, sorted by message generator or degree of importance.

3.4.6 Service Database

A database node hosts the Service Database (SDB), which is accessible from every service processor. The SDB, implemented in MySQL, contains the following information:

- Node attributes used by the application launch utilities to schedule jobs. Node attributes include the number of cores present on a processor, the processor clock speed, the amount of memory available to the processor, the architecture type of the node processor, and the type of UNICOS/lc kernel running on the node.
- Accounting information (processor usage, disk storage usage, file I/O demand, application execution time, and so on). The information is accessed by accounting software that accumulates resource usage information, system availability, and system utilization and generates system accounting reports.
- System configuration tables that list and describe the configuration files.

Cray XT series system accounting is a group of utilities that run on a service node. Accounting information is collected in the SDB. The administrator uses accounting commands to generate the following reports from the SDB:

- Job and system use
- All jobs in a time period
- System-wide statistics
- Raw accounting data
- System accounting error messages

3.4.7 System Activity Reports

The `sar`(1) command collects, reports, or saves system activity information for service nodes. To get system activity information (such as accounting information) for Catamount compute nodes, use the `xtgenacct` command instead. For more information, see the `sar`(1) and `xtgenacct`(8) man pages.

Cray RAS and Management System (CRMS) [4]

The Cray RAS and Management System (CRMS) is an integrated, independent system of hardware and software that monitors Cray XT series system components, manages hardware and software failures, controls startup and shutdown processes, manages the system interconnection network, and displays the system state to the administrator. The CRMS interfaces with all major hardware and software components of the system.

Because the CRMS is a completely separate system with its own processors and network, the services that it provides do not take resources from running applications. In addition, if a component fails, the CRMS continues to provide fault identification and recovery services and enables the functioning parts of the system to continue operating.

For detailed information about the CRMS, see the *Cray XT Series System Management* manual.

4.1 CRMS Hardware

The hardware components of CRMS are the CRMS network, the System Management Workstation (SMW), the blade control processors (L0 controllers), and the cabinet control processors (L1 controllers). CRMS hardware monitors compute and service node components, operating system heartbeats, power supplies, cooling fans, voltage regulators, and RAID systems.

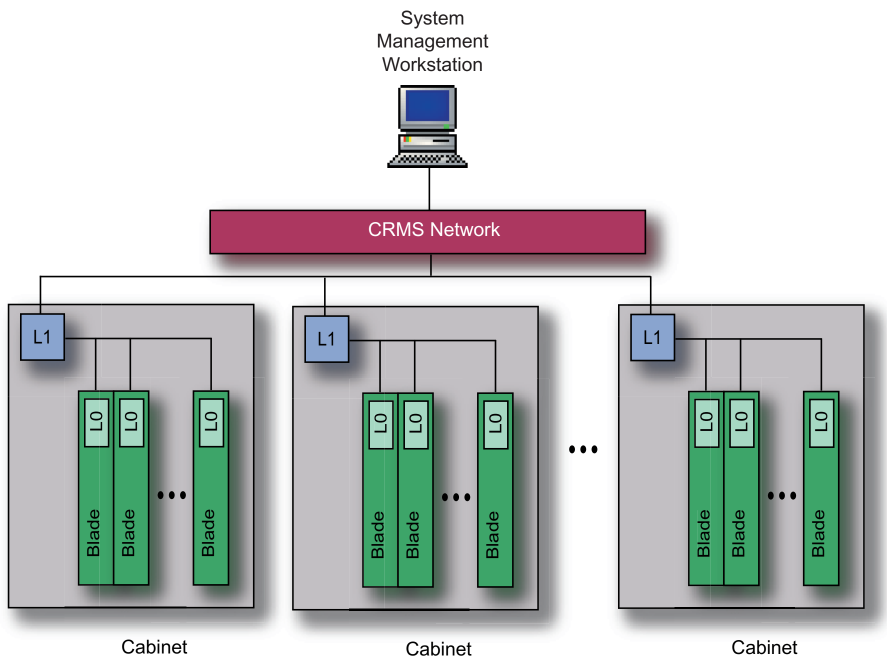

Figure 14. CRMS Components

4.1.1 CRMS Network

The CRMS network is an Ethernet connection between the SMW and the components that the CRMS monitors. The network's function is to provide an efficient means of collecting status from and broadcasting messages to system components. The CRMS network is separate from the system interconnection network.

Traffic on the CRMS network is normally low, with occasional peaks of activity when major events occur. Even during peak activity, the level of traffic is well within the capacity of the network. There is a baseline level of traffic to and from the hardware controllers. All other traffic is driven by events, either those due to hardware or software failures or those initiated by the administrator. The highest level of network traffic occurs during the initial booting of the entire system as console messages from the booting images are transmitted onto the network.

4.1.2 System Management Workstation

The System Management Workstation (SMW) is the administrator's single-point interface for booting, monitoring, and managing Cray XT series system components. The SMW consists of a server and a display device. Multiple administrators can use the SMW, locally or remotely over an internal LAN or WAN.

Note: The SMW is also used to perform system administration functions (see Section 3.4, page 37).

4.1.3 Hardware Controllers

At the lowest level of the CRMS are the L0 and L1 controllers that monitor the hardware and software of the components on which they reside.

Every compute blade and service blade has a blade control processor (L0 controller) that monitors the components on the blade, checking status registers of the AMD Opteron processors, the Control Status Registers (CSRs) of the Cray SeaStar chip, and the voltage regulation modules (VRMs). The L0 controllers also monitor board temperatures and the UNICOS/lc heartbeat.

Each cabinet has a cabinet control processor (L1 controller) that communicates with the L0 controllers within the cabinet and monitors the power supplies and the temperature of the air cooling the blades. Each L1 controller also routes messages between the L0 controllers in its cabinet and the SMW.

4.2 CRMS Software

The CRMS software consists of software monitors; the administrator's CRMS interfaces; and event probes, loggers, and handlers. This section describes the software monitors and administrator interfaces. For a description of event probes, loggers, and handlers, see Section 4.3, page 47.

4.2.1 Software Monitors

The System Environment Data Collections (SEDC) CRMS manager monitors the system health and records the environmental data (such as temperature) and the status of hardware components (such as power supplies, processors, and fans). SEDC can be set to run at all times or only when a client is listening. By default, SEDC is configured to scan the system hardware components automatically.

Resiliency communication agents (RCAs) run on all compute nodes and service nodes. RCAs are the primary communications interface between the node's operating environment and the CRMS components external to the node. They monitor software services and the operating system instance on each node.

Each RCA provides an interface between the CRMS and the system processes running on a node for event notification, informational messages, information requests, and probing. The RCA also provides a subscription service for processes running on the nodes. This service notifies the current node of events on other nodes that may affect the current node or that require action by the current node or its functions.

Each RCA generates a periodic heartbeat message, enabling CRMS to know when an RCA has failed. Failure of an RCA heartbeat is interpreted as a failure of the UNICOS/lc operating system on that node.

RCA daemons running on each node start a CRMS process called *failover manager*. If a service fails, the RCA daemon broadcasts a service-failed message to the CRMS.

Failover managers on other nodes register to receive these messages. Each failover manager checks to determine if it is the backup for any failed services that relate to the message and, if it is, directs the RCA daemon on its node to restart the failed service.

4.2.2 CRMS Administrator Interfaces

The CRMS provides both a command-line and a graphical interface. The `xtcli` command is the command line interface for managing the Cray XT series system from the SMW. The `xtgui` command launches the graphical interface. In general, the administrator can perform any `xtcli` function with `xtgui` except boot.

The SMW is used to monitor data, view status reports, and execute system control functions. If any component of the system detects an error, it sends a message to the SMW. The message is logged and displayed for the administrator. CRMS policy decisions determine how the fault is handled. The SMW logs all information it receives from the system to the SMW disk to ensure the information is not lost due to component failures.

4.3 CRMS Actions

The CRMS manages the startup and shutdown processes and event probing, logging, and handling.

The CRMS collects data about the system (event probing and logging) that is then used to determine which components have failed and in what manner. After determining that a component has failed, the CRMS initiates some actions (event handling) in response to detected failures that, if left unattended, could cause worse failures. The CRMS also initiates actions to prevent failed components from interfering with the operations of other components.

4.3.1 System Startup and Shutdown

The administrator starts a Cray XT series system by powering up the system and booting the software on the service nodes and compute nodes. Booting the system sets up the system interconnection network. Starting the operating system brings up the RCA and CPA.

A script, set up by the administrator, shuts the system down.

For logical machines, the administrator can boot, run diagnostics, run user applications, and power down without interfering with other logical machines as long as the CRMS is running on the SMW and the machines have separate file systems.

For details about the startup and shutdown processes, see the *Cray XT Series System Management* manual and the `xtcli`(8) man page.

4.3.2 Event Probing

The CRMS probes are the primary means of monitoring hardware and software components of a Cray XT series system. The CRMS probes that are hosted on the SMW collect data from CRMS probes running on the L0 and L1 controllers and RCA daemons running on the compute nodes. In addition to dynamic probing, the CRMS provides an offline diagnostic suite that probes all CRMS-controlled components.

4.3.3 Event Logging

The event logger preserves data that the administrator uses to determine the reason for reduced system availability. It runs on the SMW and logs all status and event data generated by:

- CRMS probes
- Processes communicating through RCA daemons on compute and service nodes
- Other CRMS processes running on L0 and L1 controllers

Event messages are time stamped and logged. Before the message reaches the input queue of the event handler, an attempt is made to recover from a failure. If a compute or service blade fails, the CRMS notifies the administrator.

4.3.4 Event Handling

The event handler evaluates messages from CRMS probes and determines what to do about them. The CRMS is designed to prevent single-point failures of either hardware or system software from interrupting the system. Examples of single-point failures that are handled by the CRMS system are:

- Compute node failure. A failing compute node is automatically isolated and shut down and the user job fails; the rest of the system continues running and servicing other applications.
- Power supply failure. Power supplies have an N+1 configuration for each chassis in a cabinet; failure of an individual power supply does not cause an interrupt of a compute node.

In addition, the CRMS broadcasts failure events over the CRMS network so that each component can make a local decision about how to deal with the fault. For example, both the L0 and L1 controllers contain code to react to critical faults without administrator intervention.

Glossary

blade

1) A field-replaceable physical entity. A service blade consists of AMD Opteron processors, memory, Cray SeaStar chips, PCI-X cards, and a blade control processor. A compute blade consists of AMD Opteron processors, memory, Cray SeaStar chips, and a blade control processor. 2) From a system management perspective, a logical grouping of nodes and blade control processor that monitors the nodes on that blade.

blade control processor

A microprocessor on a blade that communicates with a cabinet control processor through the CRMS network to monitor and control the nodes on the blade. See also *blade*; *Cray RAS and Management System (CRMS)*.

cabinet control processor

A microprocessor in the cabinet that communicates with the CRMS through the CRMS network to monitor and control the devices in a system cabinet. See also *Cray RAS and Management System (CRMS)*.

cage

A chassis on a Cray XT series system. See *chassis*.

Catamount

The operating system kernel developed by Sandia National Laboratories and implemented to run on Cray XT series single-core compute nodes. See also *Catamount Virtual Node (CVN)*, *compute node*, and *CNL*.

Catamount Virtual Node (CVN)

The Catamount kernel enhanced to run on dual-core Cray XT series compute nodes.

chassis

The hardware component of a Cray XT series cabinet that houses blades. Each cabinet contains three vertically stacked chassis, and each chassis contains eight vertically mounted blades. See also *cage*.

CNL

A Cray XT series compute node operating system; sites can use it as an alternative to Catamount.

compute blade

See *blade*.

compute node

Runs a kernel and performs only computation. System services cannot run on compute nodes. See also *node*; *service node*.

compute processor allocator (CPA)

A program that coordinates with yod to allocate processing elements.

Cray RAS and Management System (CRMS)

A system of software and hardware that implements reliability, availability, and serviceability (RAS) and some system management functions. The CRMS components use a private Ethernet network, not the system interconnection network. See also *system interconnection network*.

Cray SeaStar chip

The component of the system interconnection network that provides message routing and communication services. See also *system interconnection network*.

CrayDoc

Cray's documentation system for accessing and searching Cray books, man pages, and glossary terms from a Web browser.

deferred implementation

The label used to introduce information about a feature that will not be implemented until a later release.

distributed memory

The kind of memory in a parallel processor where each processor has fast access to its own local memory and where to access another processor's memory it must send a message through the interprocessor network.

dual-core processor

A processor that combines two independent execution engines ("cores"), each with its own cache and cache controller, on a single chip.

interconnect

See *system interconnection network*.

L0 controller

See *blade control processor*.

L1 controller

See *cabinet control processor*.

logical machine

An administrator-defined portion of a physical Cray XT series system, operating as an independent computing resource.

login node

The service node that provides a user interface and services for compiling and running applications.

metadata server (MDS)

The component of the Lustre file system that stores file metadata.

module

See *blade*.

Modules

A package on a Cray system that enables you to dynamically modify your user environment by using module files. (This term is not related to the module statement of the Fortran language; it is related to setting up the Cray system environment.) The user interface to this package is the `module` command, which provides a number of capabilities to the user, including loading a module file, unloading a module file, listing which module files are loaded, determining which module files are available, and other such capabilities.

node

For UNICOS/lc systems, the logical group of processor(s), memory, and network components acting as a network end point on the system interconnection network. See also *processing element*.

object storage target (OST)

The component of the Lustre file system that handles file activities.

parallel processing

Processing in which multiple processors work on a single application simultaneously.

Portals

A message-passing interface that enables scalable, high-performance network communication between nodes. Applications communicating at the user level link to the Cray MPICH2 or Cray SHMEM library. The Portals interface is transparent to the application programmer.

processing element

The smallest physical compute group. There are two types of processing elements: a compute processing element consists of an AMD Opteron processor, memory, and a link to a Cray SeaStar chip. A service processing element consists of an AMD Opteron processor, memory, a link to a Cray SeaStar chip, and PCI-X links.

reliability, availability, serviceability (RAS)

System hardware and software design that achieves increased system availability by avoiding or recovering from component and system failures.

resiliency communication agent (RCA)

A communications interface between the operating environment and the CRMS. Each RCA provides an interface between the CRMS and the processes running on a node and supports event notification, informational messages, information requests, and probes. See also *Cray RAS and Management System (CRMS)*.

service blade

See *blade*.

service database (SDB)

The database that maintains the global system state.

service node

A node that performs support functions for applications and system services. Service nodes run SUSE LINUX and perform specialized functions. There are six types of predefined service nodes: login, IO, network, boot, database, and syslog.

specialization

The process of setting files on the shared-root file system so that unique files can be present for a node or for a class of node.

system interconnection network

The high-speed network that handles all node-to-node data transfers.

System Management Workstation (SMW)

The workstation that is the single point of control for the CRMS and system administration. See also Cray RAS and Management System (CRMS).

UNICOS/lc

The operating system for Cray XT series systems.

Index