Cray X1™ System Overview

S–2346–22

Record of Revision

Version	*Description*
1.0	June 28, 2002 Draft printing to support the Cray X1 early production systems.
2.0	December 20, 2002 Draft printing to support Cray X1 systems running the UNICOS/mp 2.0, Cray Programming Environment 4.2, Cray MPT 2.1, Cray Workstation (CWS) 2.0, and CNS 1.0 releases.
2.1	March 17, 2003 Draft printing to support Cray X1 systems running the UNICOS/mp 2.1, Cray Programming Environment 4.3, and Cray MPT 2.1 releases.
2.2	June 2003 Supports Cray X1 systems running the Cray Programming Environment 5.0, Cray MPT 2.2, and UNICOS/mp 2.2 releases.

Contents

Preface

This publication introduces Cray X1 systems. It assumes the reader is familiar with UNICOS, UNICOS/mk, or UNIX systems. This preface describes how to access Cray manuals and man pages referenced in this system overview, interpret our typographical conventions, order Cray documentation, and contact us about this document.

Accessing Cray Documentation

Each Cray X1 software release package includes the CrayDoc documentation system, a collection of open-source software components that gives you fast, easy access to and the ability to search all Cray manuals, man pages, and glossary in HTML and/or PDF format from a web browser at the following locations:

- Locally, using the network path defined by your system administrator
- On the Cray public web site at:

  `http://www.cray.com/craydoc/`

All software release packages include a software release overview that provides information for users, user services staff, and system administrators about that release. An installation guide is also provided with each software release package. Release overviews and installation guides are supplied in HTML and PDF formats as well as in printed form. Most software release packages contain additional reference and task-oriented documentation in HTML and/or PDF formats.

Man pages provide system and programming reference information. Each man page is referred to by its name followed by a number in parentheses:

`manpagename(`*`n`*`)`

where *n* is the man page section identifier:

1	User commands
2	System calls
3	Library routines
4	Devices (special files) and Protocols
5	File formats
7	Miscellaneous information

8	Administrator commands

Access man pages in any of these ways:

- Enter the `man` command to view individual man pages in ASCII format; for example:

  ```
  man ftn
  ```

  To print individual man pages in ASCII format, enter, for example:

  ```
  man ftn | col -b | lpr
  ```

- Use a web browser with the CrayDoc system to view, search, and print individual man pages in HTML format.
- Use Adobe Acrobat Reader with the CrayDoc system to view, search, and print from *collections* of formatted man pages provided in PDF format.

If more than one topic appears on a page, the man page has one primary name (`grep`, for example) and one or more secondary names (`egrep`, for example). Access the ASCII or HTML man page using either name; for example:

- Enter the command `man grep` or `man egrep`
- Search in the CrayDoc system for `grep` or `egrep`

Typographical Conventions

The following conventions are used throughout this document:

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.

Ordering Documentation

To order software documentation, contact the Cray Software Distribution Center in any of the following ways:

E-mail:
`orderdsk@cray.com`

Web:
`http://www.cray.com/craydoc/`

Click on the `Cray Publications Order Form` link.

Telephone (inside U.S., Canada):
1–800–284–2729 (BUG CRAY), then 605–9100

Telephone (outside U.S., Canada):
Contact your Cray representative, or call +1–651–605–9100

Fax:
+1–651–605–9001

Mail:
Software Distribution Center
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120–1128
USA

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:
`swpubs@cray.com`

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

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

Mail:
Software Publications
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120–1128
USA

Introduction [1]

Cray X1 systems utilize powerful vector processors, shared memory, and a modernized vector instruction set in a highly scalable configuration to provide the computational power required for advanced scientific and engineering applications. A Cray X1 system has high memory bandwidth and scalable system software, which are crucial to achieving peak and sustained performance.

This document provides you with a brief overview of the Cray X1 hardware and software capabilities and points you to other Cray documentation for detailed information. It is not intended to provide you with an exhaustive list of all software features and utilities included with your Cray X1 system.

This overview includes the following information:

- Chapter 2, page 3, Hardware Overview
- Chapter 3, page 17, Development Environment Overview
- Chapter 4, page 33, Operations Overview

1.1 Related Publications

This document includes references to other Cray documents that contain detailed information. For information about how to access these documents, see the Preface of this overview.

If you are migrating from a UNICOS or UNICOS/mk system, the following migration guides will also be of interest to you:

- *Cray X1 User Environment Differences*
- *Migrating Applications to the Cray X1 System*
- *Cray X1 System Administration Differences*

Hardware Overview [2]

A Cray X1 system combines the single-processor performance and single-shared address space of Cray parallel vector processor (PVP) systems with the high bandwidth, low latency, scalable interconnect, and scalable microprocessor-based architecture used in Cray T3E systems.

As shown in Figure 1, a Cray X1 system contains the following major functional blocks:

- The *Cray X1 mainframe* consists of nodes and a node interconnection network housed in one or more cabinets. Each node is housed on a single module and contains processors, globally addressable shared local memory, and four System Port Channel (SPC) I/O ports. Nodes communicate with each other through the node interconnection network, which, depending on the mainframe cabinet type, may include router modules.
- The *Cray X1 I/O architecture* includes the following components housed in multiple cabinets:
  - I/O drawers (IODs) that convert the SPC protocol used by the nodes to Fibre Channel protocol used by various peripheral devices
  - Cray Programing Environment Server (CPES) that runs the programming environment
  - Cray Network Subsystem (CNS) that connects the Cray X1 system to the customer’s networks
  - RAID subsystem that provides disk storage for the Cray X1 system
- The *Cray X1 support system* is used to boot, configure, troubleshoot, and monitor the Cray X1 system. This support system consists of a Cray Workstation (CWS), the System Control Facility, and several private Ethernet subnetworks.

This chapter describes these three major functional blocks. It also provides information about partitioning a system and describes the two Cray X1 system model types—air-cooled (AC) and liquid-cooled (LC) models.

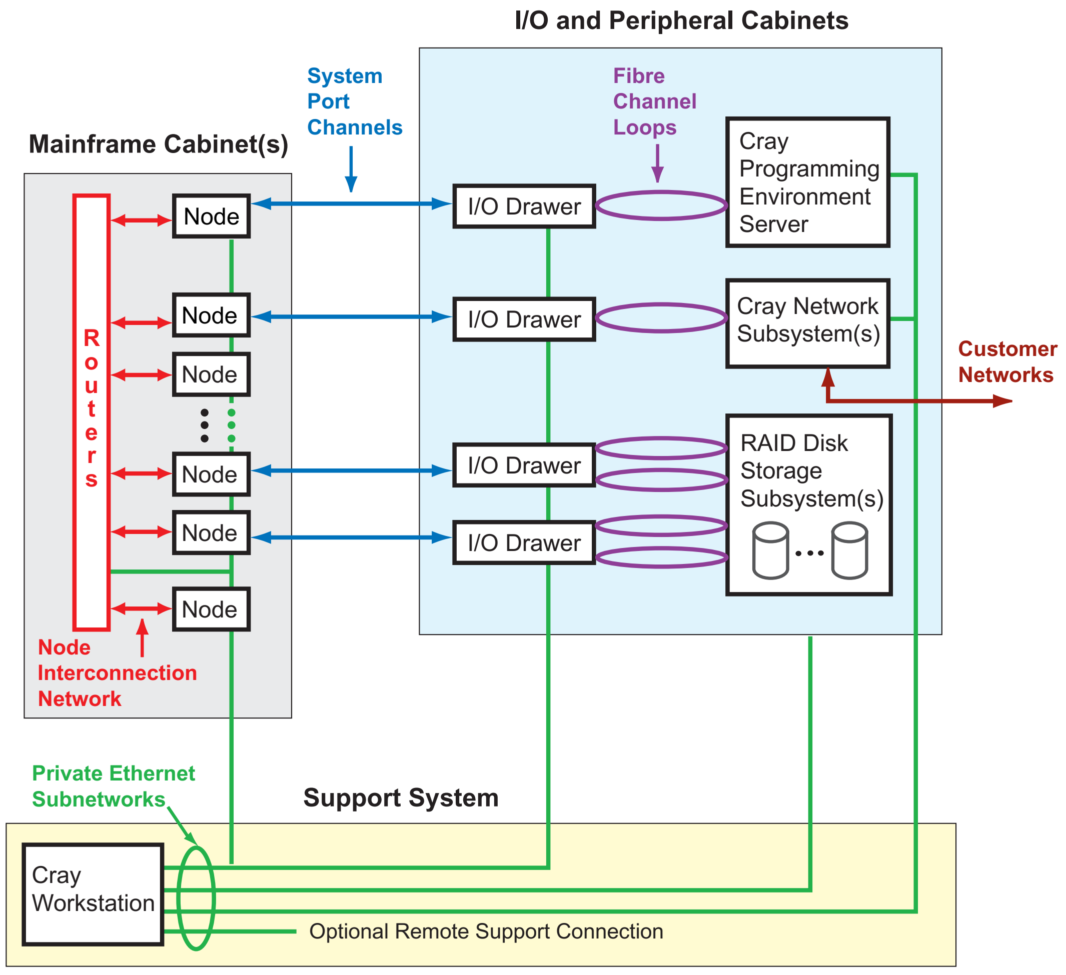
I/O and Peripheral Cabinets
System Port Channels
Fibre Channel Loops
Mainframe Cabinet(s)
Node
Node
Node
Node
Node
Node
Routers
I/O Drawer
I/O Drawer
I/O Drawer
I/O Drawer
Cray Programming Environment Server
Cray Network Subsystem(s)
Customer Networks
RAID Disk Storage Subsystem(s)
Node Interconnection Network
Private Ethernet Subnetworks
Support System
Cray Workstation
Optional Remote Support Connection

Figure 1. Cray X1 System Functional Diagram

2.1 Cray X1 Mainframe

The primary blocks of a Cray X1 mainframe are nodes and the node interconnection network housed in one or more cabinets. Nodes communicate with each other through the node interconnection network, which, depending on the mainframe cabinet type, may include router modules. Nodes and the node interconnection network are described in the following sections.

2.1.1 Nodes

The scalable building block for a Cray X1 mainframe is a node. Each node is housed on a single module called a node module.

Each *node* contains processors, globally addressable shared local memory, and four SPC I/O ports. A distributed set of routing switches controls all memory access within the node and access to the node interconnection network. The system's processing power, memory, and I/O bandwidth scale as nodes are added to the system.

Each node also contains two controllers (not shown in Figure 1) that are part of the System Control Facility. The System Control Facility is used for configuring, booting, troubleshooting, and monitoring the system. The controllers communicate with the CWS through an Ethernet connection. (For more information about the CWS, see Section 2.3.1, page 12; for more information about the System Control Facility, see Section 2.3.2, page 12.)

A Cray X1 system requires a minimum of two nodes. Physically, all nodes are the same; software controls how a node is used. The following subsections define the major functional blocks of a node.

2.1.1.1 Node Processors

Each node consists of four *multistreaming processors (MSPs)*; each MSP includes a 2-MB cache.

Both 32-bit and 64-bit arithmetic are supported. As depicted in Figure 2, a single MSP provides 12.8 GF (gigaflops) of peak computational power for 64-bit data computation (theoretically, 32-bit computation is 2X faster than 64-bit computation; however, other constraints such as memory bandwidth will reduce this peak for typical application execution).

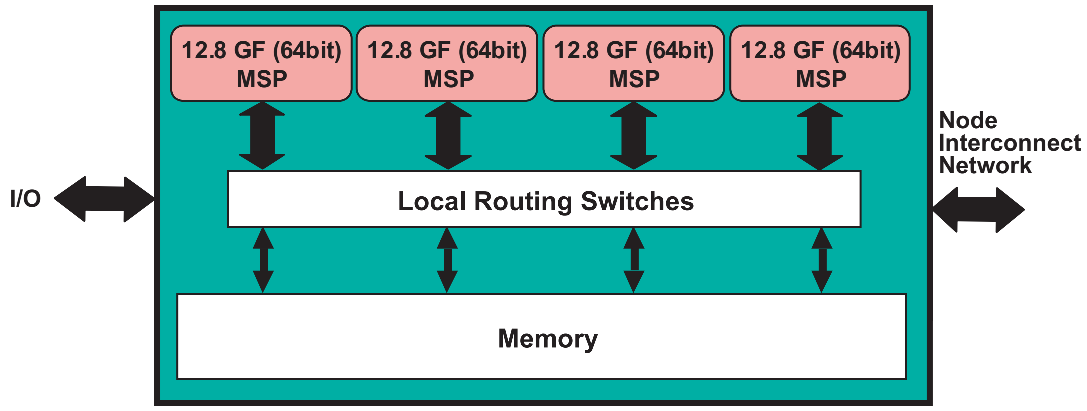

Figure 2. Cray X1 Node Module Block Diagram

Each MSP is composed of four *single-streaming processors (SSPs).* The four SSPs share the 2-MB cache of the MSP, as shown in Figure 3. Each SSP contains both a superscalar processing unit and a two-pipe vector processing unit.

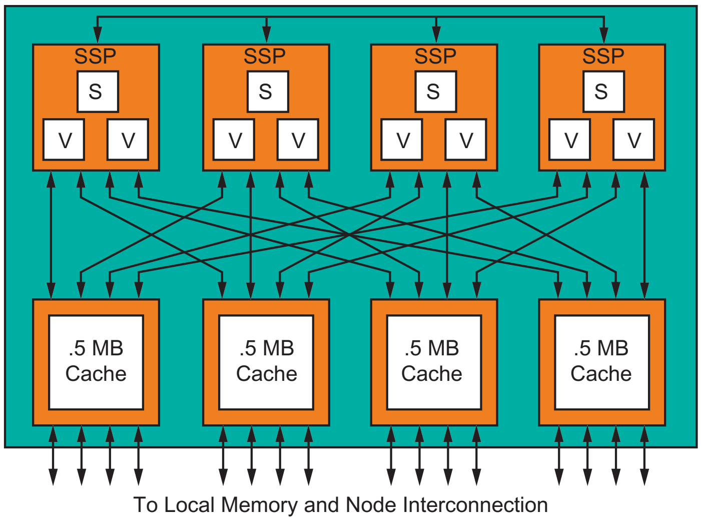

Figure 3. MSP Functional Diagram

As a result of this design, an application can run in either MSP mode or SSP mode. In *MSP mode*, an MSP provides the user-programmable processor for typical parallel applications; each MSP automatically distributes the parallel parts of a multistreaming application to its component SSPs. In *SSP mode*, each SSP runs independently of the others, executing its own stream of instructions.

Cray X1 systems implement Cray's new vector (NV-1) instruction set architecture (ISA). The NV-1 instruction set features:

- Support for 32-bit and 64-bit two's-complement integers
- Support for IEEE–754 floating-point format (both 32-bit and 64-bit formats)
- Fixed 32-bit width instructions with regular encoding
- Large register sets to reduce the number of memory accesses and to hide memory latency
- Hardware features to support improved vectorization
- Cache allocation control to support explicit communication and reduce cache pollution
- Relaxed memory ordering rules with mechanisms for explicit synchronization
- Ease of use and transition for current Cray vector processor users
- Simple data paths and simple instructions
- Support for decoupled scalar and vector execution for maximum performance

2.1.1.2 Local Memory

Each node contains globally-shared local memory. Each memory word is 72 bits wide; 64 bits are used for data and the remaining 8 bits are used for single-error-correction, double-error-detection (SECDED) protection. Each 32-bit half of a 64-bit data word can be written separately to support 32-bit operations. Local memory is accessible by all processors on the node, processors on other nodes, and all I/O devices.

Local memory latency is completely flat for all processors within a single node. Total peak local memory bandwidth for one node is 204.8 GBps, which supports network traffic and I/O without greatly impacting computational performance.

Each node contains a set of 16 memory controller chips and 32 memory daughter cards. Daughter cards with 288 Mb DRAMs (256 Mb for data, 32 Mb for error correction) are currently used, yielding a per node memory capacity of 16 GB.

Systems with larger memory size per node will be available as the appropriate Rambus DRAM technology becomes available.

Local memory can operate in two degraded modes. First, half of the memory chips on a card can be disabled to tolerate the loss of a memory chip. This degraded mode cuts the memory size in half. Second, half the daughter cards can be disabled to tolerate the failure of a daughter card. This degraded mode cuts both memory size and bandwidth in half. It is possible to degrade down to a quarter of memory on a node.

2.1.1.3 System Port Channel (SPC) I/O Ports

Each Cray X1 node has four Cray proprietary protocol SPCs that serve as I/O ports between the mainframe and an I/O drawer that resides in an I/O cabinet (see Section 2.2.1, page 9 for information about I/O drawers). I/O channel capacity scales with the node count. The peak channel bandwidth is 1.2 GBps per direction per SPC. The total I/O bandwidth per node is 4.8 GBps full duplex. Full duplex means that input and output channel bandwidth can be sustained at the same time.

The SPCs are logically accessible by all processors, but physically distributed among the nodes. All I/O channel controllers are globally addressable and controllable. There is no special relationship between an I/O channel controller and its local node other than physical proximity. Data is routed to memory without going through the processors.

2.1.2 Node Interconnection Network

Nodes communicate with each other through the *node interconnection network*, which consists of router logic on the node module, cables, and depending on the mainframe cabinet type, router modules. On smaller systems (2 to 4 nodes), the router logic on the nodes connects directly to each other. For systems with router modules, router modules connect the nodes within a mainframe cabinet and, in larger configurations, connect nodes from one mainframe cabinet to nodes in other mainframe cabinets.

Each node provides the first-level routing function. The node determines whether a memory reference is local to the node or whether the reference is to a memory location on a remote node. Remote memory references are made through the node interconnection network. Each node contains 32 network ports, each port supports 1.6 GBps peak per direction.

Configuration tables are used for data flow mapping across the node interconnection network. These tables are loaded at system boot time. They can also be reloaded in the event of a hardware failure, thus providing a means to reconfigure the node interconnection network around hardware failures.

2.2 Cray X1 I/O Architecture

The Cray X1 I/O architecture includes the following components, which are housed in multiple cabinets:

- I/O drawers that convert the SPC protocol used by the nodes to Fibre Channel protocol used by various peripheral devices. Each drawer supports four SPC channels to the mainframe and up to 16 Fibre Channels to peripheral devices.
- Cray Programming Environment Server (CPES) that runs the tools used by the programmer for program development.
- Cray Network Subsystem (CNS) that connects a Cray X1 system to the customer's networks; supported protocols include Gigabit Ethernet and HIPPI.
- RAID subsystem that provides disk storage for the Cray X1 system.

The I/O architecture components are described in the following subsections.

2.2.1 I/O Drawer

An I/O drawer resides in an I/O cabinet (IOC) and converts SPC channel protocol to Fibre Channel protocol. Each drawer supports four SPC channels using four independent circuits. Each circuit contains an I/O Channel Adapter (IOCA). The IOCA converts one SPC channel to a dual-slotted PCI-X bus. Dual-port Fibre Channel host bus adapter (HBA) cards are inserted into each PCI-X slot. Thus, each SPC channel can drive four independent Fibre Channel connections (each I/O drawer can drive up to 16 Fibre Channel connections). The Fibre Channels are configured as arbitrated loops and drive all peripheral devices. Each loop connects to a single Fibre Channel device.

The PCI-X Fibre Channel cards can be used to connect directly to RAID storage located in a peripheral cabinet (PC-20) using the normal Fibre Channel Arbitrated Loop (FCAL) protocol. These cards can also be used to make a network connection to the CNS and the CPES using Internet Protocol (IP) over Fibre Channel.

Each I/O drawer has controllers that are part of the System Control Facility, which is used for booting, configuring, troubleshooting, and monitoring purposes.

2.2.2 Cray Programming Environment Server

A Cray X1 system includes a Cray Programming Environment Server (CPES). The Cray Programming Environment compilers, loader, and performance analysis tool reside on the CPES. Program compilations are invoked by user commands on the Cray X1 mainframe and are executed transparently on the CPES. There is no user access (login) directly to the CPES. (For an overview of the development environment, see Chapter 3, page 17.) The CPES resides in a PC-20 cabinet.

2.2.3 Cray Network Subsystem

A Cray X1 system includes a Cray Network Subsystem (CNS) for network routing. In addition to bridging from Fibre Channel connections to network protocols, the CNS provides IP packet management, minimizing the impact on the Cray X1 processors. Initially, the CNS supports Gigabit Ethernet and HIPPI network types; other network types will be supported on an as-needed basis. Multiple CNSs can be configured on a Cray X1 system to scale the number of network connections required. The CNS resides in a PC-20 cabinet.

2.2.4 RAID Subsystem for Disk Storage

A Cray X1 system uses one or more RAID (redundant array of independent drives) subsystems for disk storage. The RAID subsystems are configured using various RAID levels and consist of modular bricks with full built-in redundancy. A RAID subsystem resides in a PC-20 cabinet. Each RAID subsystem includes one controller brick (C-brick) and its associated storage bricks (S-bricks). Figure 4 shows a block diagram of a RAID subsystem.

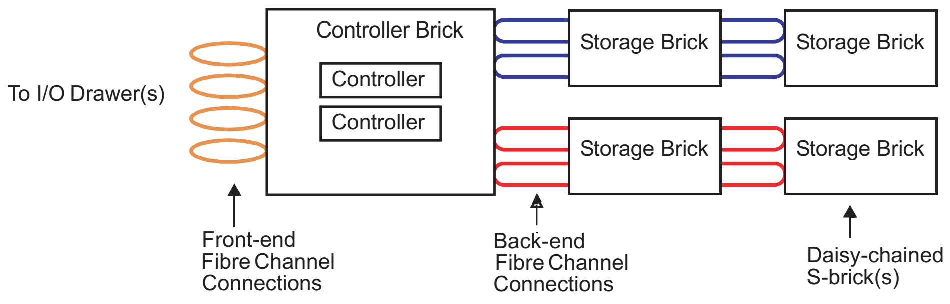

Figure 4. RAID Subsystem Block Diagram

Each C-brick has the following features:

- Four Fibre Channel front-end (or host) connections (to host bus adapters in I/O drawers)
- A pair of redundant controllers
- Four back-end connections, configured as two channel pairs (to the S-bricks)
- Ethernet ports for communication and administration

Full redundancy is built into the C-brick. Any of the four front-end connections can communicate with any of the four back-end connections.

The S-bricks are the scalable storage component. Each S-brick contains a number of dual-ported Fibre Channel disk drives. The dual ports provide another level of resiliency. The number of drives per brick and the size of each disk varies with the different S-brick models. S-bricks can be daisy-chained to increase storage capacity on a given RAID subsystem. S-bricks are software configured, allowing for flexibility in defining the size and quantity of disk devices on a RAID subsystem.

A Cray X1 system can contain multiple RAID subsystems. For sites where storage capacity is critical, fewer RAID systems with daisy-chained S-bricks may be used. For sites where disk transfer speed is important, multiple RAID systems with minimal daisy-chaining may be used.

2.3 Cray X1 Support System

The Cray X1 support system is used to boot, configure, troubleshoot, and monitor the Cray X1 system. This support system consists of a Cray Workstation (CWS), the System Control Facility, and several private Ethernet subnetworks, which are described in the following subsections.

2.3.1 Cray Workstation

Each system includes one CWS, which is used by Cray X1 system administrators and operators to configure, troubleshoot, monitor, and boot the system. The CWS communicates to all of the system components through various private Ethernet subnetworks. The CWS provides the operational interface for all system maintenance and operator functions, including consolidating log messages from the CWS, System Control Facility, CPES, UNICOS/mp, and CNS as part of the centralized logging capability. It includes the Cray Storage Management (CRAYSM) software for configuring RAID subsystems, and the interface for remote maintenance via a modem (if allowed by customer agreement with Cray).

2.3.2 System Control Facility

The System Control Facility (not shown in Figure 1) consists of controllers that reside on each Cray X1 node module, router module, I/O drawer, and the mainframe's power/cooling equipment. These controllers monitor the power and cooling subsystem and provide the operational interface to the system for initial system startup, driver interface communications to Cray X1 node and router modules, and mainframe hardware error logging.

2.3.3 Private Ethernet Subnetworks

There are several private Ethernet subnetworks in the Cray X1 system; all connect to the CWS through Ethernet switches. A brief description of each subnetwork follows:

- *System Control Facility Subnetwork*: This subnetwork connects the CWS to the controllers in node modules, router modules, mainframe power and cooling control circuits, and I/O drawers.
- *Private Administration Subnetwork*: This subnetwork connects the CWS to Ethernet ports on the CPES and CNSs.
- *RAID Subnetwork*: This subnetwork connects the CWS to Ethernet ports on RAID controller bricks. The RAID administration software uses this subnetwork to configure and to monitor the status of the RAID subsystems.

- *Customer Service Subnetwork*: This optional subnetwork allows Cray Customer Service remote access to the CWS. It is used for remote support if allowed by customer agreement with Cray.

2.4 System Partitioning

A Cray X1 system can be divided into multiple partitions. A partition is a set of nodes that functions as an independent Cray X1 system. Each Cray X1 partition requires a minimum of two node modules.

Each partition is independently booted, dumped, and so on without impacting other running partitions. Users log on to a partition as if it were an independent Cray X1 machine.

Because a Cray X1 partition does not have access to I/O facilities on other partitions, each partition must have independent I/O drawers, CNSs, and RAID subsystems. If users wish to compile from a partition, that partition also requires either a separate CPES or an independent connection to a common CPES. However, all partitions connect to a common CWS.

2.5 Hardware Models

A Cray X1 system is available as either an air-cooled (AC) or a liquid-cooled (LC) model.

Both models are multicabinet systems that include one or more of the following cabinets: mainframe cabinet (MFC), I/O cabinet (IOC), and peripheral cabinet (PC-20). The mainframe cabinets differ between the model types. Both models use the same IOC and PC-20 cabinets. IOCs and PC-20s are always air cooled. The following subsections describe the two models in more detail.

2.5.1 Air-cooled Model

A Cray X1 air-cooled system requires one MFC-AC mainframe cabinet, one IOC, and one PC-20. Additional peripheral cabinets can be added, depending on the system configuration. All cabinets are air cooled; the system does not use facility water for cooling. Air-cooled systems do not require a raised computer room floor, but a raised floor is recommended.

Figure 5 shows a Cray X1 air-cooled system with one MFC-AC mainframe cabinet, one IOC, and two PC-20s.

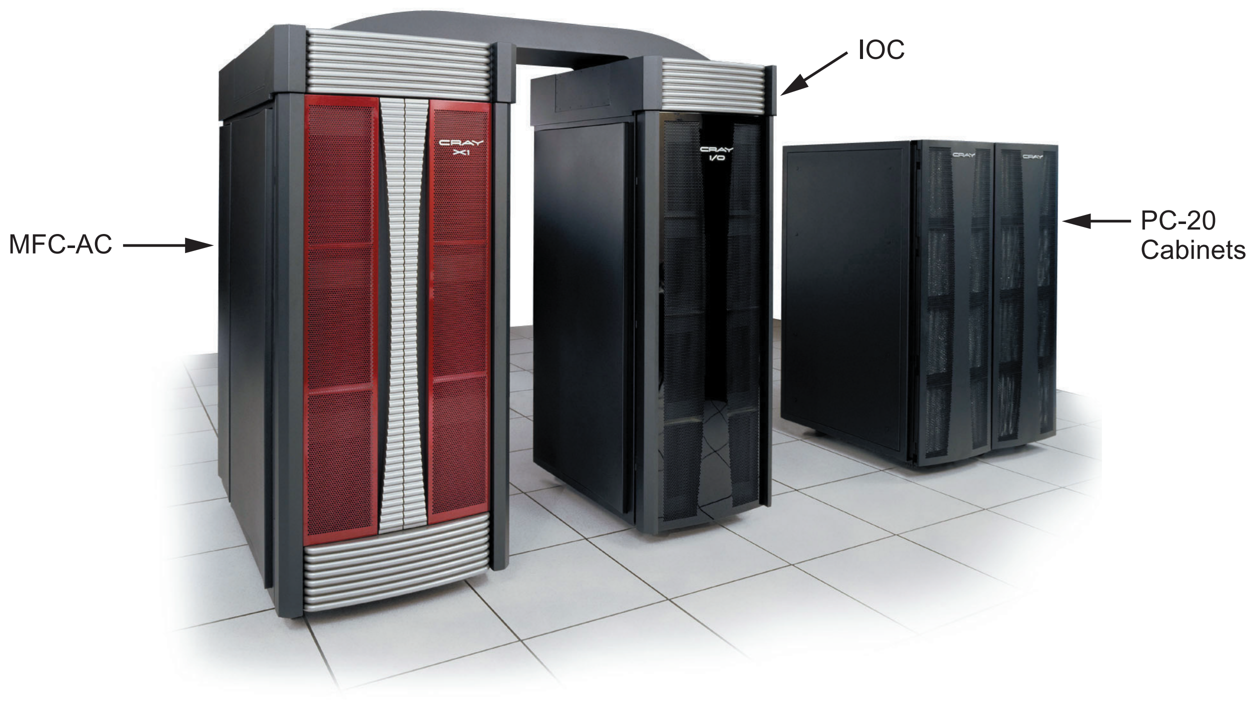

Figure 5. Cray X1 Air-cooled System

The MFC-AC and IOC are physically attached to each other. Cabling between the MFC-AC and IOCs is run through cable troughs at the top of the system. All PC-20s are freestanding; cabling between an IOC and PC-20s is typically run under the raised computer room floor.

An MFC-AC cabinet holds up to 4 node modules and their associated routers. It is intended that AC system configurations will have one to two MFC-AC cabinets, yielding systems that range from 2 to 8 nodes (8 to 32 MSPs). Based on customer demand, multicabinet systems of up to four MFC-ACs may be available in the future, yielding systems that range from 2 to 16 nodes (8 to 64 MSPs).

An IOC houses I/O drawers (IODs) that provide a bridge between the MFC-AC and its various peripheral devices. The MFC-AC communicates with an IOD using the SPC protocol. An IOD communicates with the Cray X1 system's various peripheral devices through Fibre Channel protocol. The Fibre Channels connect to/from the RAID disk storage, the CPES, and CNSs.

Each Cray X1 air-cooled system requires a minimum of one IOD in the IOC. The number of drawers is dependent on the system’s mainframe and peripheral configuration. An IOC can support up to 32 SPC mainframe connections and up to 128 Fibre Channel connections.

A PC-20 houses various peripherals, including the CPES, the CNS, and the RAID disk storage components. All Cray X1 air-cooled systems require a minimum of one PC-20 cabinet. Additional PC-20 cabinets are added based on the system’s peripheral configuration.

2.5.2 Liquid-cooled Model

A Cray X1 liquid-cooled system requires a minimum of one MFC-LC mainframe cabinet, one IOC, and one PC-20. Additional cabinets can be added, depending on the system configuration. An MFC-LC requires facility water for cooling. IOCs and PC-20s are air-cooled. Cray X1 liquid-cooled systems require a raised computer room floor for routing the facility water.

Figure 6 shows a Cray X1 liquid-cooled system consisting of one MFC-LC, one IOC, and two PC-20s.

All MFC-LCs and IOCs are physically attached to each other. Cabling between the MFC-LCs and IOCs is run through cable troughs at the top of the system. All PC-20s are freestanding; cabling between the IOC and PC-20s is typically run under the raised computer room floor.

Each MFC-LC holds up to 16 node modules and their associated routers. Cray X1 liquid-cooled systems can consist of 1 to 64 MFC-LC cabinets, yielding systems that range from 2 to 1024 nodes (8 to 4096 MSPs), with approximately 50 TFLOPS of aggregate computational power in its largest configuration.

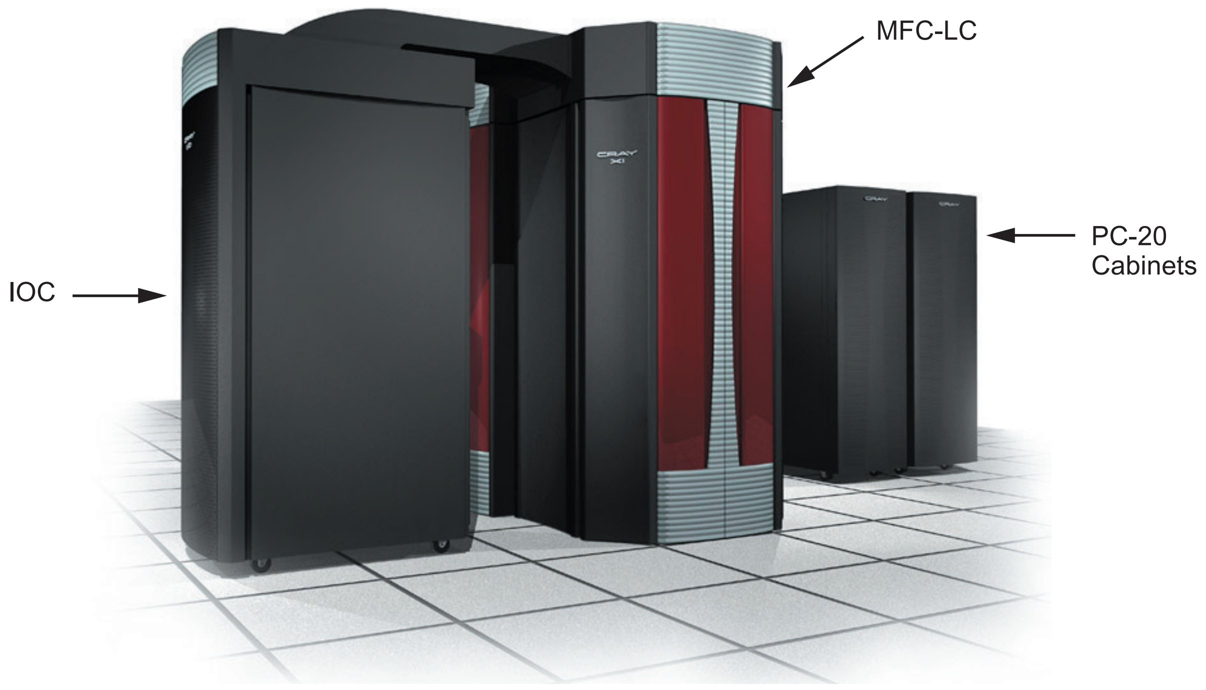

Figure 6. Cray X1 Liquid-cooled System

An IOC houses I/O drawers (IODs) that provide a bridge between the MFC-LC and its various peripheral devices. The MFC-LC communicates with an IOD using the SPC protocol. An IOD communicates with the Cray X1 system's various peripheral devices through Fibre Channel protocol. The Fibre Channels connect to/from the RAID disk storage, the CPES, and CNSs.

Each Cray X1 liquid-cooled system requires a minimum of one IOC with two IODs. The number of cabinets and drawers is dependent on the system's mainframe and peripheral configuration. An IOC can support up to 32 SPC mainframe connections and up to 128 Fibre Channel connections.

A PC-20 houses various peripherals, including the CPES, the CNS, and the RAID disk storage components. All Cray X1 liquid-cooled systems require a minimum of one PC-20 cabinet. Additional PC-20 cabinets are added based on the system's peripheral configuration.

Development Environment Overview [3]

This chapter presents an overview of the development environment for Cray X1 systems. It describes the software that is available to you to build, execute, monitor, checkpoint, debug, and optimize applications on the Cray X1 system.

If you are migrating from a UNICOS or UNICOS/mk system, you should also refer to *Cray X1 User Environment Differences* and *Migrating Applications to the Cray X1 System*.

Detailed information about optimizing your applications for use on Cray X1 systems is provided in *Optimizing Applications on the Cray X1 System*.

Note: For information about how to access Cray manuals or man pages mentioned in this chapter, see the Preface of this overview.

3.1 Development Tools

The development environment for Cray X1 systems is the software that is provided with or that runs on top of the UNICOS/mp operating system. The UNICOS/mp operating system is based on the IRIX 6.5 operating system with Cray enhancements. It is designed to sustain the high-capacity, high-bandwidth throughput generated by Cray X1 systems. A single UNICOS/mp kernel image runs on the entire system.

The software comprises:

- Commands, system calls, system libraries, special files, protocols, and file formats provided with the UNICOS/mp operating system; for information about UNICOS/mp operating system functionality, see Section 4.6.2, page 39.
- Software provided in or with the Cray Programming Environment product:
  - Cray Fortran Compiler and/or the Cray C and C++ Compilers
  - CrayTools, which includes the Cray loader and CrayPat, the performance analysis tool for Cray X1 systems
  - Libraries for application programmers
  - The Modules utility
  - X Window System X11 libraries

  - Motif (if proper licensing is in place), which allows you to develop X Window System applications on the Cray X1 system that are portable across a variety of platforms
  - Trigger commands for initiating a process or sequence of processes on the CPES
  - Cray Assembler (CAL) for Cray X1 systems

- Message Passing Toolkit (MPT) for Cray X1 systems, which is available from Cray as an optional product
- The Etnus TotalView debugger for Cray X1 systems, which is available from Cray as an optional product
- PBS Pro batch workload system for Cray X1 systems, which is available from Cray as an optional product
- Cray Open Software (COS) package for Cray X1 systems, which is available from Cray as an optional product

3.2 MSP and SSP Modes

As described in Section 2.1.1, page 5, a Cray X1 system is made up of nodes. Software defines how a node is used. A node's software-assigned "flavor" dictates the kind of processes and threads that can use its resources. The three flavors (assignable by the system administrator) are application, support, and OS. *Application nodes* are used to run user applications. *Support nodes* are used to run serial commands, such as shells, editors and other user commands, such as `ls`. *OS nodes* provide kernel-level services, such as system calls, to all support nodes and application nodes.

A Cray X1 system requires a minimum of two nodes. Each system typically uses at least one node that is designated as both an OS node and a support node; this node is often called a *system node*.

As shown in Figure 7, each node has four *multistreaming processors (MSPs)*; each MSP includes a 2-MB cache. Each MSP is composed of four processing elements called *single-streaming processors (SSPs)*. Each SSP contains both a superscalar processing unit and a two-pipe vector processing unit, depicted as S and V, respectively in Figure 7. The four SSPs share the 2-MB cache of the MSP.

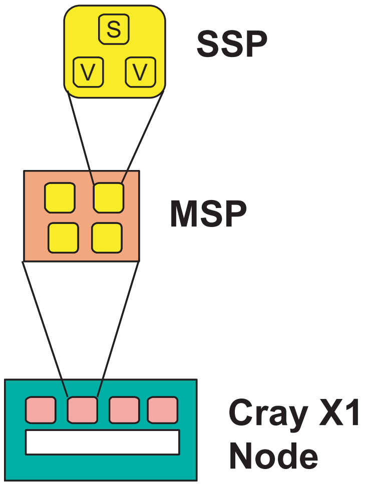

Figure 7. Cray X1 Node Makeup

An application can run in either MSP mode or SSP mode. The most powerful way to use an MSP is to tightly couple the four SSPs to work together; you can use distributed programming models to use multiple MSPs. This is called *MSP mode*. In *SSP mode*, each SSP runs independently of the others, executing its own stream of instructions; you can use distributed programming models to use multiple SSPs.

By default, an application runs in MSP mode. You designate SSP mode by using the `-O ssp` option (Fortran) or the `-h ssp` option (C or C++). You must compile and link using this option to get an SSP executable.

You can compile routines to be single-streamed for MSP mode executables, and you can call a single-streamed routine from a streamed region using Cray Streaming Directives.

Note: Routines compiled for SSP mode cannot be linked with routines compiled for MSP mode.

For additional information, see the `ftn`(1), `cc`(1), and `CC`(1) man pages, the *Cray Fortran Compiler Commands and Directives Reference Manual*, and the *Cray C and C++ Reference Manual*.

3.3 Programming Models

High-level parallelization of codes is supported on Cray X1 systems by both shared memory and distributed memory models and some hybrid combinations of certain models.

Table 1 shows the memory models supported on Cray X1 systems.

Table 1. Memory Models Supported on Cray X1 Systems

Distributed Memory Models	Shared Memory Models
Message Passing Interface (MPI)	POSIX threads (C and C++)
Unified Parallel C (UPC) (subset)	(Deferred implementation) OpenMP (Fortran, C, and C++)
Co-array Fortran (CAF)	
Shared Memory (SHMEM)	

Distributed memory models imply anywhere from no visibility of address space (as with MPI standards) to full visibility (as with UPC and CAF) between tasks or threads of execution. However, all distributed memory models require explicit definition and referencing of data objects not local to a task or thread of execution. These distributed models are well-suited for a non-uniform memory access (NUMA) system like Cray X1 systems.

Shared memory models are shared in the sense that the address space is visible to all tasks or threads within that model. Shared memory model address space is limited to the memory on single nodes. Single nodes support uniform memory access (UMA), which provides optimal OpenMP performance. The shared memory models supported on Cray X1 systems are POSIX threads and OpenMP (support for OpenMP on Cray X1 systems is deferred).

CSDs are provided as a way to access SSPs from an MSP application. CSDs are supported for C, C++, and Fortran.

It is possible to mix shared and distributed memory models or different distributed memory models in the same application. In such “hybrid” models, the shared memory model is seen as being applied to a single process of the distributed memory model. In other words, the distributed memory model application should be viewed as composed of multiple processes, each of which is composed of multiple shared memory threads. All the threads of a single process must execute on the same Cray X1 node.

Each model has pros and cons that should be understood by the developer. All higher level parallelization models should be viewed as optional and secondary to vectorization and multistreaming.

3.4 Vectorization and Multistreaming

The Cray compilers automatically translate code to make optimal use of machine resources. Register allocation, scheduling, vectorization, and multistreaming are all performed by default, and they are carefully balanced to give the best possible performance. Compiler vectorization provides loop level parallelization of operations and uses the vector processing hardware (execution pipes, load buffers, and functional unit groups).

Multistreaming code generation by the compiler permits execution across the four SSPs of an MSP on a block of code. Vectorization and multistreaming can be intermixed, and multistreaming often extends outside loop boundaries.

3.5 Accessing a Cray X1 System

A set of commands, including all commands defined by POSIX 1003.2–1992, is provided. As shown in Figure 8, page 22, you log in to a Cray X1 system from your network-connected system using standard `telnet` or `rlogin` commands, as permitted by your local site. In addition, your site administration may offer or require logins through the open secure shell command, `ssh`. Related commands for file transfers, `ftp`, `rcp`, `sftp`, and `scp`, are also supported. (The `ssh`, `sftp`, and `scp` commands are available with the Cray Open Software (COS) package.)

You enter commands in a standard terminal window on the Cray X1 system. The system supports `sh` (defaults to `ksh`), `csh`, and `tcsh` user shells. The Cray Network Subsystem provides a transparent connection between users on a network-connected system and the Cray X1 system. In addition, the X Window System client commands and libraries and Motif are available for ease of use.

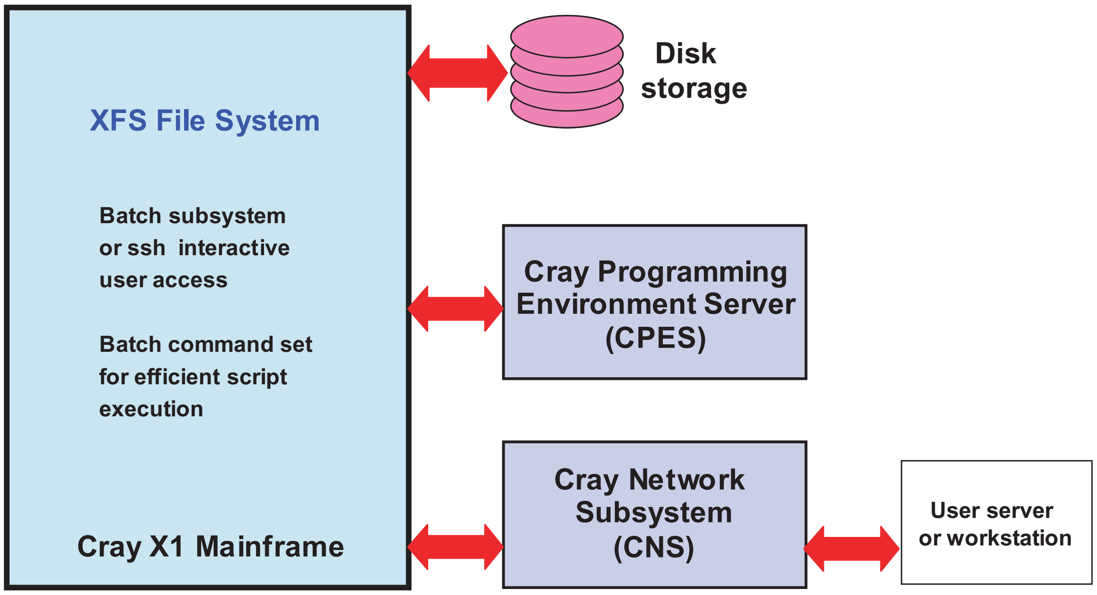
Disk storage
XFS File System
Batch subsystem or ssh interactive user access
Batch command set for efficient script execution
Cray X1 Mainframe
Cray Programming Environment Server (CPES)
Cray Network Subsystem (CNS)
User server or workstation

Figure 8. Accessing the Cray X1 System

If the PBS Pro batch management system is running on your Cray X1 system, PBS Pro has the capability to manage both batch and interactive sessions. An interactive session allows you to have the standard input, output, and error streams of the job connected to the terminal session in which the job is running. For additional information, see the *PBS Pro User's Guide*.

System and user files are stored on disks on a Cray X1 system. The file management system for these files is the XFS file system, which is a journaling, mature file system with support for very large individual files and file systems. Data files and temporary storage may be maintained on Cray X1 disk storage or on disks mounted using NFS.

The role of the Cray Programming Environment Server (CPES) is discussed in Section 3.12, page 27.

3.6 Modules Package

The Modules package is available on Cray X1 systems; it allows you to dynamically modify your user environment by using module files. The user interface to this package is the `module` command. The `module` command provides a number of capabilities to the user, including loading or unloading a module file, listing which module files are loaded, determining which module files are available, and others.

You use a module file to define information specific to an application or collection of applications. For example, to define the paths, command names, and other environment variables to use the Cray Programming Environment, use the `PrgEnv` module file, which contains the base information needed for application compilations. The `mpt` module file sets a number of environment variables needed for message passing and data passing application development.

An example use of the `module` command is as follows:

```
module load PrgEnv
```

After the Modules package has been loaded to your user environment, documentation about the use of the `module` utility is available to you on the `module`(1) and `modulefile`(4) man pages.

Note: The `PrgEnv` module file must be loaded before any compilations, assemblies, or loads can be executed on a Cray X1 system.

For information about compiling and loading your code, see Section 3.12, page 27.

3.7 Cray Fortran Compiler and CAF Support

This section briefly describes the Cray Fortran Compiler and the Co-array Fortran extension for distributed memory parallel processing.

3.7.1 Cray Fortran Compiler

The Cray Fortran Compiler translates Fortran programs into Cray X1 object files. The Cray Fortran Compiler fully supports the Fortran language through the Fortran 95 Standard, which is the International Organization of Standards ISO/IEC 1539–1:1997. Selected features from the proposed Fortran 2000 Standard are also supported.

The compiler command is `ftn`. (The `f90` command is replaced on Cray X1 systems by the `ftn` command.) The user interface to the compiler is very similar to Cray T3E and Cray parallel vector processing (PVP) Fortran compilers. The majority of compiler options and directives remain applicable to Cray X1 systems. For specifics, see *Cray X1 User Environment Differences* and *Migrating Applications to the Cray X1 System*.

Fortran programs running on Cray T3E systems that require high scalability and Fortran programs running on Cray T90 and Cray SV1 systems should easily port to a Cray X1 system. The Cray Fortran Compiler and libraries support 8-bit, 16-bit, 32-bit, and 64-bit integer and logical data types; 32-bit, 64-bit, and 128-bit floating-point data; and character data. Previous Cray PVP Fortran compilers did not support 8-bit, 16-bit, or 32-bit arithmetic data types.

The compiler will vectorize and stream based on the compiler options that you set and the directives you use.

Detailed information about the `ftn` command is available on the `ftn`(1) man page and in the *Cray Fortran Compiler Commands and Directives Reference Manual*. The *Fortran Language Reference Manual* describes the Fortran language as implemented by the Cray Fortran Compiler.

3.7.2 Co-array Fortran (CAF)

Co-array Fortran is a syntax extension of Fortran 95 for distributed memory parallel processing. Co-array Fortran was implemented on Cray T3E systems and is available on Cray X1 systems.

Co-array Fortran adopts the single-program-multiple-data (SPMD) programming model. A Co-array Fortran program is interpreted as if it were replicated a number of times and all copies were executed concurrently. Each copy has its own set of data objects and is termed an image. Simple syntax allows images to reference or define data objects in other images. Intrinsic subroutines are provided to synchronize images. Co-array Fortran can be used as an alternative to message-passing methods such as MPI and SHMEM.

For more information, see the *Fortran Language Reference Manual, Volume 3*.

3.8 Cray C/C++ Compilers and UPC Support

This section briefly describes the Cray C and C++ compilers and the UPC extension of the C programming language.

3.8.1 Cray C and C++ Compilers

The Cray C and C++ compilers translate C and C++ programs into Cray X1 object files. The `cc`, `CC`, `c89`, and `cpp` commands work with the C and C++ compilers:

- The C++ compiler, which is invoked by the `CC` command, conforms to the International Standards Organization (ISO) standard ISO / IEC 14882:1998, with some exceptions.
- The C compiler, which is invoked by the `cc` command, conforms to the standard ISO / IEC 9899:1999 (C99).
- The `c89` command functions as defined by POSIX 1003.2–1992.
- The `cpp` command invokes only the preprocessor component of the Cray Standard C compiler.

The C and C++ compilers and libraries support the LP64 data type model, which includes 16-bit, 32-bit, and 64-bit data types (in addition to byte-oriented types such as char). Previous Cray PVP compilers did not support 16-bit or 32-bit arithmetic data types.

The compiler will vectorize and stream based on the compiler options that you set and the directives you use.

Detailed information about the `cc` and `CC` compilers is available on the `cc`(1) and `CC`(1) man pages and in the *Cray C and C++ Reference Manual*. If you are migrating code from a UNICOS or UNICOS / mk system, you should also refer to *Cray X1 User Environment Differences* and *Migrating Applications to the Cray X1 System*.

3.8.2 Unified Parallel C (UPC)

UPC is a parallel extension of the C programming language intended for multiprocessors with a common global address space. A subset of UPC is available on Cray X1 systems.

For more information, see the *Cray C and C++ Reference Manual*.

3.9 Libraries

Procedures and functions that programmers frequently use are provided by Cray and made readily available in libraries. To use one of these library routines, you need only refer to it, and the routine is incorporated into your program automatically.

The library routines that are provided with Cray X1 systems relate to:

- UNICOS/mp operating system; for information about these library routines, see the `intro`(3) man page.
- Fortran procedures
- C and C++ functions (Edison Design Group based and Dinkumware C++)
- Fortran and C/C++ intrinsic procedures
- Science (primarily fast Fourier transforms (distributed memory version is deferred), BLAS, BLACS, LAPACK, ScaLAPACK) and mathematics; for information about these routines, see the following man pages: `intro_fft`(3s), `intro_blas1`(3s), `intro_blas2`(3s), `intro_blas3`(3s), `intro_lapack`(3s), `intro_blacs`(3s), `intro_scalapack`(3s), `intro_libsci`(3s), and `intro_libm`(3m).
- Utilities, including but not limited to flexible file I/O (FFIO), I/O, numeric conversion, error message handling, and sort and search; for information about these utilities, see the `intro_ffio`(3f), `intro_io`(3f), `intro_conversion`(3f), and `intro_sortsearch`(3f) man pages.
- Message Passing Interface (compliant with MPI-1; conforming to the MPI 1.2 specification; also providing MPI I/O and MPI one-sided communications features specified in the MPI-2 standard); for information about this library, see the `intro_mpi`(1) and `intro_mpt`(1) man pages.
- SHMEM (shared memory access); for information about this library, see the `intro_shmem`(3) man page.
- POSIX threading library (threading routines are not available to programs that use more than one node because shared memory model address space is limited to the memory on single nodes); for information about the `libpthread` POSIX threading library, see the `pthreads`(3p) man page.

3.10 Assembly Language Codes

The Cray X1 assembler assembles Cray Assembly Language files into Cray X1 object files. (The Cray X1 Assembly Language and the Cray X1 assembler are both referred to as CAL.) The Cray X1 assembler allows for explicit definition of data types as 32-bit or 64-bit and as integer or floating point. In addition, it uses hexadecimal for addresses and displayed machine code. The assembler recognizes a set of pseudo instructions, macros, micros, and opdef commands and directives.

For futher information about the Cray X1 assembler, see the `as`(1) man page and the *Cray Assembly Language (CAL) for Cray X1 Systems Reference Manual*.

3.11 Cray X1 Loader

The Cray X1 loader combines relocatable object files and library modules. It produces an output file that can be executed on Cray X1 systems. It is recommended that you execute the loader by using the compiler commands, instead of using the `ld` command to load your application. The files specified on the command line can be either object files or object libraries produced by the `ar` command.

The Cray X1 loader supports the most commonly used Cray PVP and Cray T3E loaders command line options. The Cray X1 loader does not support directives.

For further information about using the Cray X1 loader, see the `ftn`(1), `CC`(1), and `ld`(1) man pages.

3.12 Triggering the Compilation and Loading of Your Code

You log on to the Cray X1 system (or submit batch jobs to the Cray X1 system). When you execute a limited set of commands, called *trigger* commands, the Cray X1 mainframe initiates a process or process sequence on the CPES. The `cc`, `CC`, and `ftn` commands and certain tools like the CrayPat performance analysis tool are initiated by links to the trigger executable; trigger execution is transparent. When you compile and load your code, the compile and load commands actually trigger the compile and load processes to be performed on the CPES. Files are accessed between the Cray X1 mainframe and the CPES by NFS mounted file systems. The user commands that trigger CPES actions are `ar`, `as`, `cc`, `CC`, `c89`, `cpp`, `c++filt`, `ftn`, `ftnlx`, `ld`, `nm`, `pat`, `pat_build`, `pat_help`, `pat_report`, and `remps`.

3.13 Running the Cray Programming Environments on a Different Platform

The Cray Programming Environment (Fortran, C, and C++) also runs on other platforms (also known as a cross compiler). If your site has the proper licensing in place, you might choose to use one of these other platforms. The advantages of using this compiler environment include faster compile time and access to the Cray Programming Environment when the Cray X1 system is not available to you. Supported platforms are listed in the *Cray Programming Environment Releases Overview and Installation Guide*.

3.14 Executing Your Program

After the loading step is completed, your program is ready to execute on the Cray X1 mainframe. Use the `ftn`, `cc`, or `CC` command options to designate the execution mode and the number of processors needed to execute your program.

You can also use the `aprun` or `mpirun` command to launch your program (use `mpirun` for programs that use the MPI memory model). And you can set the `AUTO_APRUN_OPTIONS` environment variable to specify options for the `aprun` command before running your application. (An application is any group of one or more processes bound together by a single application ID (`apid`).)

Applications launched using the `aprun` (or `mpirun`) command create an *ApTeam* of processes. An ApTeam can be a single execution thread or a large number of processes and can use shared memory, distributed memory, or a hybrid model for parallel execution.

By using the `aprun` (or `mpirun`) command, you can designate more specifics about how and where you want your application to execute, which may be helpful for testing and development purposes. For example, you can designate that your application be placed on specific nodes. Also, Cray X1 systems allocate memory in units of pages; the base page size is 64K (65,536 bytes). The `aprun` and `mpirun` default to a page size of 16M (16,777,216 bytes) and allow you to request even larger page sizes. For more information about page size specification and other options, see the `aprun`(1) (and `mpirun`(1)) man page.

Once an application is launched, it is placed under control of the UNICOS/mp application placement scheduler tool, `psched`, which manages all hardware resources (memory and processors) on application nodes. An application is scheduled based upon how it was compiled.

Note: You can build commands written in C/C++ by using the `-h command` compiler option (this is often referred to as *command mode*). These commands are executed on a support node. The purpose of this command-mode execution is to provide rapid, interactive execution of small, generally single-threaded programs on a Cray X1 system. This command option is not supported in Fortran.

3.14.1 Accelerated and Flexible Modes

An application may execute in one of two modes: accelerated or flexible. Applications executing in *accelerated mode* perform in a predictable period of processor time, although their wall clock time may vary depending on I/O usage, network use, and/or whether any oversubscription occurs on the relevant nodes.

Due to the characteristics of the memory address space, applications executing in accelerated mode must run on logically contiguous nodes.

Applications executing in *flexible mode* may run on noncontiguous nodes, and they perform in a less predictable amount of processor time due to the flexible network topology resulting from using noncontiguous nodes.

3.14.2 Virtual and Physical Memory

A multinode application (composed of several cooperating processes) must fit in available physical memory. All other processes (including single-node applications and commands) may use more virtual memory than available physical memory by using demand paging.

Multinode applications have different virtual memory capabilities than other processes because a different mechanism is used to translate a remote (off-node) virtual memory reference to a physical address than is used to translate a local reference.

As in other virtual memory architectures, a Cray X1 system employs a Translation Lookaside Buffer (TLB) to translate local virtual memory references to physical addresses by means of a translation table stored in the TLB. When a requested page is not currently contained in the TLB, a TLB miss occurs, which causes UNICOS/mp to add a new entry to the TLB. This generally causes the overwriting of an older entry in the TLB. Sometimes the action of creating a new TLB entry requires UNICOS/mp to assign a new physical page to the process (demand paging).

Multinode applications use the Remote Translation Table (RTT) (or a software equivalent if executing in flexible mode) to reference off-node memory. Because RTT faults occur too late in the processor pipeline to be precise, restartable faults, it is not possible to do demand paging, copy-on-write, or memory-mapped files in multinode applications. Instead, each page must be preallocated to create the appropriate entries in the RTT before remote processes reference the pages. This preallocation is performed automatically by multinode applications using one of the high-level programming models. The TLB is still used for local memory references.

The RTT minimizes *thrashing* of the local TLB by permitting a hardware/software mechanism other than a TLB to perform the translation to physical memory addresses. The RTT can significantly speed up local memory references by preventing such thrashing.

UNICOS/mp supports standard UNIX user limits, which provide separate per-process limits for virtual memory size and memory resident size. UNICOS/mp also supports distinct limits for commands versus `psched`-controlled applications. The virtual memory limit is usually set larger than physical memory, so it does not come into play for multinode applications. The memory resident size limit is honored for all processes, including multinode applications.

3.14.3 Batch System

Although both interactive and batch access are provided, the user environment is strongly oriented toward batch use. Cray offers the PBS Pro batch workload management system. The PBS Pro implementation on UNICOS/mp systems uses a command-line API with commands that will be familiar to former NQS/NQE users. Optionally, PBS Pro allows you to run your job interactively; the job is queued and scheduled as any PBS Pro batch job, but when executed, the standard input, output, and error streams of the job are connected through `qsub` to the terminal session in which `qsub` is running. When the job begins execution, all input to the job is from the terminal session in which `qsub` is running.

PBS Pro interacts with the UNICOS/mp `psched` tool and uses its resource management and job placement capabilities. Either through the native capabilities of PBS Pro or in cooperation with the `psched` tool, PBS Pro maintains queues of jobs by resource type and priority and supports preemptive executions, job dependencies, and checkpoint/restart capabilities. The package enforces site policy limits for users and jobs. For simple executions, the `qsub` and `qstat` PBS Pro commands may suffice. For additional information, see the `qsub`(1) and `qstat`(1) PBS Pro man pages and the *PBS Pro User Guide*.

3.15 Monitoring Your Program

The `psview` command provides status information for one or all applications running on the mainframe. The `apkill` command sends a kill signal to an ApTeam. For more information about these commands, see the `psview`(1) and `apkill`(1) man pages.

You can also monitor activity on the CPES by executing certain commands that run remotely, such as the `remps`(1) command, which is documented on the `remps`(1) man page.

3.16 Checkpointing Your Program

UNICOS/mp provides a checkpoint/restart software management tool, `cpr`, to checkpoint a process or a set of processes that is executing and to restart it later. You must be the owner of a process or set of processes being checkpointed, although system administrators and operators may checkpoint/restart any process. You can checkpoint both MSP mode and SSP mode applications.

Options to the `cpr` command let you create, query, restart, and delete checkpoints. By using the `cpr -p` *id*`[:`*type*`]` command, you can specify the process or set of processes to checkpoint. The *id* can be one of the following types:

- `PID` (UNIX process; this is the default)
- `GID` (UNIX process group ID)
- `SID` (UNIX process session ID)
- `HID` (Process hierarchy (tree) rooted at that PID; this type checkpoints all associated shells and child processes of that PID)
- `APT` ApTeams (application team ID)

The `cpr` tool works with the `psched` daemon to restart ApTeams on the application nodes.

For additional information about using the UNICOS/mp checkpoint/restart tool, see the `cpr`(1) man page.

3.17 Debugging Your Program

Cray offers the TotalView debugger from Etnus, LLC. Etnus TotalView is a scalable debugger that is designed to debug parallel applications programmed in a variety of parallel programming models. Both the command line interface and the graphical user interface are supported on UNICOS/mp systems to control debugging sessions. The Etnus TotalView debugger can be used to debug MSP mode or SSP mode applications.

For information about the optional Etnus TotalView debugger for UNICOS/mp systems, see the *TotalView Release Overview, Installation Guide, and User's Guide Addendum for Cray X1 Systems*.

Note: The GNU `gdb` debugger is offered from Cray as an unsupported debugger in the Cray Open Software (COS) package.

3.18 Improving Your Application's Performance

The Cray X1 performance analysis tool, CrayPat, allows you to perform profiling, sampling, and tracing experiments on an instrumented application and to analyze the results of those experiments; no recompilation is needed to produce the instrumented program. In addition, CrayPat provides access to all hardware performance counters. CrayPat is a single, integrated application performance analysis tool. (The Cray PVP and Cray MPP performance tools `perfview`, `flowview`, `procview`, `profview`, `jumpview`, `atexpert`, `apprentice`, and `pat` are not available on Cray X1 systems.) CrayPat can be used with MSP mode or SSP mode applications.

The `pat` command is a trigger command; you launch it on the Cray X1 mainframe, and it executes on the CPES. The instrumented application executes on the Cray X1 mainframe. (For information about triggering your code, see Section 3.12, page 27.)

Note: The `pat` command is currently deferred. You must run the `pat_build` command on your `a.out` file to instrument your program; the resulting instrumented program is put in an `inst.out` file. Executing the instrumented program creates a data file containing the results of the experiment. You can then run the `pat_report` command to view or dump the data file.

The `pat_hwpc` utility collects hardware performance counter statistics for an executed command. The `pat_hwpc` utility executes a program and writes out the values of the specified hardware performance counters. Alternately, a process that is already executing may have its counters captured.

For additional information about CrayPat, see the `pat`(1), `pat_build`(1), `pat_hwpc`(1), `pat_report`(1), and `counters`(5) man pages; *Optimizing Applications on the Cray X1 System*; and `pat_help`, which contains examples.

Operations Overview [4]

This chapter provides the following operations overview information for Cray X1 system administrators:

- A list of the software release packages for Cray X1 systems and the related administration documentation
- Brief descriptions of software functionality with references to Cray X1 system administration documentation for detailed information

This chapter addresses system administrators who are experienced UNICOS or UNICOS/mk system administrators or have administered other systems based on the UNIX operating system. For information about MSP and SSP concepts, see Section 3.2, page 18.

If you are migrating from a UNICOS or UNICOS/mk system, *Cray X1 System Administration Differences* will also be of interest to you.

Note: For information about how to access Cray manuals or man pages mentioned in this chapter, see the Preface of this overview.

4.1 Software Release Packages

Cray X1 system administration encompasses:

- Cray Workstation (CWS) software, which also includes RAID administration
- Cray Programming Environment Server (CPES) software
- Cray Programming Environment software, which runs on the Cray Programming Environment Server (CPES)
- Cray Network Subsystem (CNS) software
- UNICOS/mp operating system software
- Any optional software products running on your Cray X1 system

As a system administrator, you will need to understand the software products that are provided in the following release packages for Cray X1 systems:

- Cray Workstation (CWS) release package, which includes the following CWS administration documentation:
  - *Cray Workstation (CWS) Release Overview*
  - *Cray Workstation (CWS) Installation Guide*
  - *Cray X1 Configuration and CWS Administration*
  - *Man Page Collection: Cray Workstation (CWS)*
- Cray Programming Environment Server (CPES) release package, which includes the following CPES administration documentation:
  - *Cray Programming Environment Server (CPES) Release Overview and Installation Guide*
  - *Cray Programming Environment Server Administration*
- Cray Fortran Programming Environment release package and/or the Cray C and C++ Programming Environment release package, which include administration (trigger) scripts, module files, the *Cray Programming Environment Releases Overview and Installation Guide*, and the *Man Page Collection: Trigger Commands*
- Modules release package (also included with the Cray Programming Environment release packages)
- Cray Network Subsystem (CNS) release package, which includes the *Cray Network Subsystem (CNS) Software Release Overview and Installation Guide*
- UNICOS/mp release package, which includes the following UNICOS/mp administration documentation:
  - *UNICOS/mp Release Overview*
  - *UNICOS/mp Installation Guide*
  - *UNICOS/mp System Administration*
  - *UNICOS/mp Disks and File Systems Administration*
  - *UNICOS/mp Networking Facilities Administration*
  - *Man Page Collection: UNICOS/mp Administrator Commands*

– *Man Page Collection: UNICOS/mp File Formats and Special Files*

– *Man Page Collection: UNICOS/mp User Commands*

– *Man Page Collection: UNICOS/mp System Calls*

– *Man Page Collection: UNICOS/mp Library Routines*

If your site also chooses to use any of the following products, you should also know the administration needs for these software packages:

- Message Passing Toolkit (MPT) release package, which includes the *Cray Message Passing Toolkit Release Overview*
- Etnus TotalView debugger release package, which includes the *TotalView Release Overview, Installation Guide, and User's Guide Addendum for Cray X1 Systems* and TotalView documentation from Etnus LLC
- PBS Pro batch system release package for UNICOS/mp systems, which includes the *PBS Pro Release Overview, Installation Guide, and Administration Addendum for Cray Systems* and PBS Pro documentation from Altair Grid Technologies, L.L.C.
- Cray Open Software release package, which includes the *Cray Open Software Release Overview and Installation Guide* and documentation about each included software package

4.2 Cray Workstation (CWS) Functionality

You use the CWS to perform administration functions, such as configuring, operating, administering, monitoring, booting, halting, dumping, and diagnosing your Cray X1 system.

The CWS is also used to install CWS and UNICOS/mp software using the Common Installation Tool (CIT). (All Cray X1 software release packages may be installed using CIT.)

The UNICOS/mp system uses RAID (redundant array of independent drives) devices exclusively for storage. Configuring and managing UNICOS/mp RAID controllers is done primarily through Cray storage management (CRAYSM) tools on the CWS. (For additional information about RAID devices and disk storage, see Section 2.2.4, page 10.)

The CWS has private network connections to the System Control Facility, the Cray Programming Environment Server (CPES), the Cray Network Subsystem (CNS), the RAID disk storage, and an optional customer service network used for remote support by Cray Customer Service. (For additional information about CWS private network connections, see Section 2.3, page 11.)

In addition, there is a centralized logging capability; the CWS consolidates log messages from the CWS, System Control Facility, CPES, UNICOS/mp, and CNS.

For detailed information about CWS functions and administration, see the *Cray X1 Configuration and CWS Administration* and *UNICOS/mp Disks and File Systems Administration*.

4.3 Cray Programming Environment Server (CPES) Functionality

The Cray Programming Environment compilers, loader, and performance analysis tool reside on the CPES. Program compilations are invoked by user (trigger) commands on the Cray X1 mainframe and are executed on the CPES in a way that is transparent to the user. There is no user access (login) directly to the CPES. All users requiring access to the Cray Programming Environment must have accounts set up on the CPES to match their accounts on the Cray X1 system (same uids). The CPES has a Gigabit Ethernet connection so that file systems on other servers that are needed for compiling can be mounted on both the Cray X1 mainframe and on the CPES.

The CPES resides in a PC-20 peripheral cabinet. It runs the Solaris operating system.

For detailed information about CPES functions and administration, see the *Cray Programming Environment Releases Overview and Installation Guide*.

Note: The Cray Programming Environment (Fortran, C, and C++) also runs on other platforms (also known as a cross compiler). If your site has the proper licensing in place, you might choose to use one of these other platforms so that your users will have faster compile time and will have access to the Cray Programming Environment when the Cray X1 system is not available to them. Supported platforms are listed in the *Cray Programming Environment Releases Overview and Installation Guide*.

4.4 Trigger Environment

Included with the Programming Environment release package are the trigger environment commands. When a user enters the name and options of a CPES-hosted command on the command line of the Cray X1 mainframe, a trigger environment command executes, setting up an environment for the CPES-hosted command. This trigger environment command duplicates the portion of the current working environment on the Cray X1 mainframe that relates to the Programming Environment. The user interacts with the system as if all elements of the Programming Environment are hosted on the Cray X1 mainframe. Commands entered on the Cray X1 system trigger the execution of the corresponding CPES-hosted commands that have the same names.

A template `trigexecd.cfg` file is provided with the Programming Environment release, which the administrator needs to edit to ensure the trigger environment commands will function properly. For additional information, see the *Man Page Collection: Trigger Commands*.

4.5 Cray Network Subsystem (CNS) Functionality

The CNS is a specialized router on the Transmission Control Protocol / Internet Protocol (TCP / IP) network. Access to that network from a Cray X1 system is controlled entirely by the CNS software. The specialized function of the CNS is to off-load the Cray X1 mainframe resources that would otherwise be spent handling IP networking traffic. This improves the TCP bandwidth for each connection between the Cray X1 mainframe and front-end host systems.

Initially, the CNS supports Gigabit Ethernet and HIPPI network types; other network types will be supported on an as-needed basis. Multiple CNSs can be configured on a Cray X1 system to scale the number of network connections required.

The CNS resides in a PC-20 peripheral cabinet. It runs Cray-specific software with Red Hat Linux software.

For detailed information about the CNS, see the *Cray Network Subsystem (CNS) Software Release Overview and Installation Guide*.

4.6 UNICOS/mp Functionality

The UNICOS / mp operating system consists of a kernel that is based on the IRIX 6.5 operating system kernel with Cray enhancements that provide for scalability and resource scheduling.

The following sections briefly describe UNICOS / mp key functionality; detailed information is provided in the UNICOS / mp system administration documentation and UNICOS / mp man pages.

4.6.1 Key IRIX Functionality in the UNICOS/mp Operating System

Key IRIX functionality in the UNICOS/mp operating system includes:

- An environment that is based on IRIX, with command, utility, and operating system interfaces as defined by POSIX 1003.1–1990 and POSIX 1003.2–1992.
- A single-system image operating system.
- Hardware error reporting.
- A checkpoint/restart software management tool, `cpr`, to suspend a process or a set of processes that is executing and restart it later. System administrators, operators, and the owner of a process or set of processes may checkpoint the targeted process or processes. For information about using the UNICOS/mp checkpoint/restart facility, see the `cpr`(1) man page.

  Note: The PBS Pro batch system product available from Cray supports checkpointing and restarting of batch jobs. Documentation is provided in the PBS Pro software package if your site is using this optional batch system.

- The XFS journaling file system and the XLV volume manager for defining and managing logical volumes (file systems) on the Cray X1 system. Detailed information is provided in *UNICOS/mp Disks and File Systems Administration*.
- Backup and restore of XFS file systems through the `xfsdump` and `xfsrestore` utilities, using local or remote drives. You can back up file systems, directories, and/or individual files, and then restore file systems, directories, and files independently of how they were backed up. The `xfsdump` utility also allows you to back up "live" (mounted, in-use) file systems. Additional information is provided in the `xfsdump`(8) and `xfsrestore`(8) man pages and in *UNICOS/mp System Administration*.
- Network File System (NFS) to make mounted file systems available from the CPES or any other accessible server. A UNICOS/mp system can function as both an NFS client and an NFS server. Automounting of NFS file systems is supported. Information about NFS is provided in *UNICOS/mp Networking Facilities Administration*.
- Remote Procedure Call (RPC) protocol, which NFS requires for session layer services.
- Domain Name System (DNS) client; the Internet uses DNS to map names to IP addresses. Information is provided in *UNICOS/mp Networking Facilities Administration*.

- A UNICOS/mp system can function as a Network Information Service (NIS) client. NIS is a network-based information service and an administrative tool. It allows centralized database administration and a distributed lookup service. NIS supports multiple databases based on regular text files. For information about NIS, see *UNICOS/mp Networking Facilities Administration*.
- Transmission Control Protocol/Internet Protocol (TCP/IP) support, including the socket interface for network communications, `ftp`, `telnet`, and `rsh`. Information about TCP/IP is provided in *UNICOS/mp Networking Facilities Administration* and in the related man pages.
- The Network Time Protocol (NTP), which provides the mechanisms to synchronize time and coordinate time distribution in a large, diverse internetwork. The NTP protocol is implemented on UNICOS/mp systems by the `ntpd` daemon process. For information about NTP, see *UNICOS/mp Networking Facilities Administration* and the `ntpd`(8) man page.
- Standard UNIX SVR4 accounting evaluation tools such as `acctcom`, `acctcms`, and `acctmerg`. These tools enable the collection of application-wide data. Additional information is provided in *UNICOS/mp System Administration* and in the related man pages.
- System activity monitoring and reporting. System activity data is accessed on UNICOS/mp systems by using the `sar`(1) and `timex`(1) utilities. The `sar` (System Activity Reporter) command can be used to track the use of CPU time, number and type of processors, memory, and I/O to secondary storage and over network connections. Additional information is provided in *UNICOS/mp System Administration* and in the `sar`(1) and `timex`(1) man pages.
- Security features in the following forms: `/etc/passwd` file for storing user passwords and a protected `/etc/shadow` file for storing encrypted passwords; access control lists (ACLs); and granular privilege mechanism, which divides the power of the superuser into discrete units of privilege called capabilities.

4.6.2 Cray Added Functionality of Interest to All Users

Cray has added the following functionality to the UNICOS/mp operating system.

4.6.2.1 Accelerated Execution for Distributed Memory Applications

The UNICOS/mp operating system supports accelerated execution for distributed memory applications, which is called accelerated mode. This acceleration is supported by special Cray X1 memory mapping hardware through which all remote memory is completely mapped to minimize Translation Lookaside Buffer (TLB) misses and to guarantee peak application performance in tightly synchronized environments.

4.6.2.2 Distributed Memory Message Passing

Distributed memory message passing is managed through direct reads and writes as applications share a globally mapped address space across all nodes on which they execute. This feature is unique to UNICOS/mp systems.

4.6.2.3 Application Migration

A requirement for accelerated applications is that all nodes allocated to the application are logically contiguous. This is not a requirement for applications run in flexible mode, but their performance should be improved if their nodes are contiguous. The placement process prefers contiguous allocation but will fragment a flexible application if that is necessary to find it a place to run.

The contiguity requirement of accelerated applications can lead to situations where the occupied nodes and available nodes are scattered in such a way that no waiting application can be placed anywhere in the domain. Fragmentation of this kind lowers the utilization of the machine by leaving portions of it effectively unavailable. Migration is used to increase the size of contiguous free space. Migration moves applications within the domain to eliminate allocation holes. Application migration is managed by the UNICOS/mp application placement scheduler, `psched`.

For additional information, see *UNICOS/mp System Administration*.

4.6.2.4 Interactive and Batch Processing

UNICOS/mp systems support both interactive and batch processing.

Cray offers the PBS Pro product as the batch system for UNICOS/mp systems. PBS Pro is the professional version of the Portable Batch System (PBS), which is a flexible resource and workload management system. PBS Pro works with the UNICOS/mp `psched` tool so that you can initiate and manage the workload of your users' computational jobs in accordance with your site scheduling policies.

PBS Pro software is licensed to Cray customers directly from Cray, and Cray provides the PBS Pro software package, documentation, and support directly to our customers.

Detailed information about the optional batch system offered by Cray is provided in the *PBS Pro Release Overview, Installation Guide, and Administration Addendum for Cray Systems*.

4.6.2.5 Application Launch and Query Commands

A user can use the `ftn`, `cc`, or `CC` command-line option `-X` *npes* to designate the number of processors needed to execute a program. A user can also use the `aprun` or `mpirun` command to launch a program; the `mpirun` command is used for programs that use the MPI memory model. For more information, see the `ftn`(1), `cc`(1), `CC`(1), and `aprun`(1) (`mpirun`(1)) man pages.

The `psview` command uses the `psched` daemon to display the status of applications. For detailed information, see the `psview`(1) man page.

4.6.2.6 Large Pages

A user is allowed to specify page size by using the `aprun` (`mpirun`) command. The larger the page size, the fewer TLB misses occur, providing better application performance. For information about how to set the page size, see the `aprun`(1) (`mpirun`(1)) man page.

4.6.2.7 Application Monitoring

The UNICOS/mp `psview`, `apstat` and `snflv` commands provide information about running applications and node configuration. The `apkill` command is used to send a kill signal to an ApTeam. These commands are documented on the `apstat`(1), `psview`(1), `snflv`(8), and `apkill`(1) man pages.

Activity on the CPES can be monitored by executing certain commands that run remotely, such as the `remps`(1) command, which is documented on the `remps`(1) man page.

4.6.2.8 X Window System Client and Libraries

UNICOS/mp supports the X Window System client (X11R6.3); the essential X Window System libraries are provided with the Cray Programming Environment releases. For additional information, see the *Cray Programming Environment Releases Overview and Installation Guide*.

4.6.2.9 Motif Version 2.1

Motif version 2.1 is supported and, if licensed, is provided with the Cray Programming Environment releases. For additional information, see the *Cray Programming Environment Releases Overview and Installation Guide*.

4.6.3 Additional Cray Added Functionality of Interest to System Administrators

This section describes additional software functionality that Cray includes for UNICOS/mp system administration.

4.6.3.1 Software Installation

The Common Installation Tool (CIT) is used to install UNICOS/mp from the CWS. UNICOS/mp installation information is provided in the *UNICOS/mp Installation Guide*.

4.6.3.2 UNICOS/mp Storage

The UNICOS/mp system uses RAID devices for storage. UNICOS/mp treats RAID volumes as a single logical volume and communicates to it only through a RAID controller. In effect, the entire physical disk infrastructure is invisible and under total control of the RAID subsystem; UNICOS/mp sees only virtual devices. The RAID subsystem configuration and management utilities are part of the Cray Workstation (CWS) release; for additional information, see *UNICOS/mp Disks and File Systems Administration*.

4.6.3.3 UNICOS/mp File System Hierarchy

The file system hierarchy for UNICOS/mp was implemented to closely comply with the Filesystem Hierarchy Standard (FHS) Version 2.2. The UNICOS/mp file system hierarchy is documented in *UNICOS/mp System Administration*.

4.6.3.4 Security

In addition to the security features listed in Section 4.6.1, page 38, the OpenSSH and OpenSSL optional security software products are supported on UNICOS/mp systems. These open-source products are included in the Cray Open Software (COS) package. For additional information, see the *Cray Open Software Release Overview and Installation Guide*.

4.6.3.5 Network Routing

Network traffic is forwarded from a UNICOS/mp system to the Cray Network Subsystem (CNS) for network routing. The customer network uses the CNS as a gateway (router) to a UNICOS/mp system. For information about the CNS, see the *Cray Network Subsystem (CNS) Software Release Overview and Installation Guide*.

4.6.3.6 Application Placement Scheduling Mechanism

The UNICOS/mp operating system employs an enhanced version of the UNICOS/mk `psched` mechanism. The UNICOS/mp `psched` daemon supports placement of applications according to site policy, including time sharing of node memory and gang scheduling for high processor use. In addition, the UNICOS/mk Global Resource Management functionality is included as part of the UNICOS/mp `psched` daemon. The UNICOS/mp `psched` daemon is configured through the `/etc/psched.conf` file as well as through the `psmgr`(8) command.

The UNICOS/mp `psched` daemon schedules and places all applications on nodes; it includes the following major functions to support application workloads:

- Allocation of work to application nodes
- Load balancing work among the application nodes
- Gang scheduling (context switching) applications when nodes are oversubscribed

There is also a recovery feature to restore application placement scheduling if the daemon is killed or fails plus an interface to the checkpoint and restart facility so that a restarted application is properly allocated to the application nodes.

The `psched` daemon assigns applications to processing resources. The `psched` daemon adheres to the restrictions set by the configuration of gates and limits. When an application is launched using the `aprun` or `mpirun` command, `psched` compares the `aprun` command options and requirements of the application to the configured limits and gates in order to appropriately place each application.

An administrative interface using the `psmgr` command allows configuration of the `psched` functions. A set of displays for both administrators and users is provided by using the `psview` command.

Lower-level processor and memory scheduling is done by the UNICOS/mp kernel. The `psched` daemon provides information to the kernel through the `apteamctl`(2) interface. Using this information, the kernel allocates the required resources to serve the applications.

Detailed information about the UNICOS/mp application placement scheduler is provided in *UNICOS/mp System Administration*.

4.6.3.7 UNICOS/mp Accounting

Cray has added accounting capabilities to the UNICOS/mp operating system that allow you to display account IDs and consolidate by account ID.

In addition, Cray provides application accounting. Application information is collected on a per-application basis for application flags, user ID, start time (that is, when the user hits `Enter`), launch time (that is, when the application was started by `psched`), elapsed time, connect time (different from elapsed time if gang scheduled), application name, and application width and depth. The application accounting record can be viewed by using the `acctcom -A` command. Additional information is provided in *UNICOS/mp System Administration* and the `acctcom`(1) man page.

4.6.3.8 Resource Limits

The UNICOS/mp operating system allows a system administrator to dynamically assign process resource limits. Limits are defined in two dimensions: service provider type (batch or interactive) and placement class (command or application). Both maximum and initial values may be specified for each limit.

The `limit_mkdb` command is used to maintain the limit database. Each application is assigned a limit from the limits database when it is launched.

The `userlimit` command is used to see the limits imposed on individual users.

Additional information is provided in the `limit_mkdb`(8) and `userlimit`(1) man pages and in *UNICOS/mp System Administration*.

4.6.3.9 System Resiliency

UNICOS/mp provides the following system resiliency features; additional information is provided in *Cray X1 Configuration and CWS Administration* and in *UNICOS/mp System Administration*:

- **System diagnostics**. System diagnostics are included with a Cray X1 system.

- **Running a system that has a failed SSP or MSP**. A Cray X1 system can be run in a degraded mode with failing SSPs and MSPs removed from the configuration.
- **Disabling an SSP or an MSP on an application node**. The `mpadmin`(8) command allows a system administrator to disable an SSP or an MSP on a application node without requiring a system reboot.
- **Capability to run with degraded memory**. Cray X1 systems allow memory on a node to be degraded in case of memory failures. Local memory can operate in two degraded modes. First, half of the memory chips on a card can be disabled to tolerate the loss of a memory chip. This degraded mode cuts the memory size in half. Second, half the daughter cards can be disabled to tolerate the failure of a daughter card. This degraded mode cuts both memory size and bandwidth in half. It is possible to degrade down to one quarter of memory on a node module.
- **Automatic disk failover**. A Cray X1 system automatically uses the alternate path to a disk device if the first path fails. Manual intervention may be required to correct disk configuration in the event of a disk failover.

4.6.3.10 Partitioning a System

It is possible to run a single Cray X1 system as two or more independent systems (partitions). Each partition is booted and shut down independently. A Cray X1 partition does not have access to I/O facilities on other partitions. Therefore, each partition must have independent I/O drawers, CNS(s), and RAID subsystems. Each partition also requires either a separate CPES or an independent connection to a common CPES. All partitions share a single CWS. Hardware and software failures in one partition do not affect other partitions. Partitioning can take place under the following restrictions:

- Individual nodes cannot be split between partitions.
- A partition must be composed of a distinctly defined set of nodes.
- Each partition requires a minimum of 2 nodes.
- Partitioning or repartitioning requires a full system reboot.

For additional information about partitioning a Cray X1 system, see *Cray X1 Configuration and CWS Administration*.

4.6.3.11 System Maintenance Utilities

The following Cray X1 system maintenance utilities are provided with UNICOS/mp:

- **Log files**. As part of the centralized logging capability, the CWS consolidates log messages from the CWS, System Control Facility, CPES, UNICOS/mp, and CNS.
- **System dump analysis**. In UNICOS/mp, the `crashmp` command allows you to analyze mainframe system dumps generated by the `dumpmf` command. (The `crashmp` command is based on the IRIX `icrash` command, modified to include UNICOS/mk capabilities. The `dumpmf` command is based on the UNICOS/mk command.) For more information about system dump analysis, see *UNICOS/mp System Administration* and the `crashmp`(8) and `dumpmf`(8) man pages.
- **Tunable kernel parameters**. The `systune` utility lets you view and change your tunable kernel parameters. For additional information, see the `systune`(8) man page and *UNICOS/mp System Administration*.

Glossary

accelerated mode

An application may execute in one of two modes: accelerated or flexible. Applications running in accelerated mode perform in a predictable period of processor time, though their wall clock time may vary depending on I/O usage, network use, and/or whether any oversubscription occurs on the relevant nodes. Due to the characteristics of the memory address space, accelerated applications must run on logically contiguous nodes.

application

One or more computer programs that perform specific tasks for a user; these programs use the services of and are under the control of an operating system.

application node

A node that is used to run user applications. Application nodes are best suited for executing parallel applications and are managed by the strong application placement scheduling and gang scheduling mechanism `psched`. See also *OS node*; *support node*.

ApTeam

Applications launched via `aprun` or `mpirun` create an ApTeam of processes. An ApTeam can be a single execution thread or a large number of processes and can use shared memory, distributed memory, or a hybrid model for parallel execution.

array

A data structure that contains a series of related data items arranged in rows and columns for convenient access. The C shell and the `awk`(1) command can store and process arrays. A Fortran array is a set of scalar data, all of the same type and type parameters. The rank of an array is at least 1 and at most 7. Arrays may be used as expression operands, procedure arguments, and function results, and they may appear in input/output (I/O) lists.

assembler

A computer program that creates a machine language program from a symbolic language program. The assembler substitutes machine operation codes for symbolic operation codes and substitutes absolute or relocatable addresses for symbolic instructions.

assembly language

A programming language that uses symbolic notation to represent machine language instructions. It is a low-level language that is closely related to the internal architecture of the computer on which it runs.

automounting

Automatic mounting of a remote file system (using NFS) when the file system is accessed.

batch job

1. A file of commands that will be executed in batch mode. 2. A group of commands that produces a specific piece of work that the user requires to be executed in a single execution sequence.

brick

A hardware term for four grouped node modules. On the Cray X1 mainframe chassis, a brick is designated as either V or W. See also *brick-pair*.

brick-pair

Two bricks (the space required for 8 node modules) in the front or back half (V-side or W-side for notational reference) of a liquid-cooled Cray X1 mainframe cabinet. Air-cooled Cray X1 systems have only the V-side brick.

buffer

In software, a block of memory that the system uses to store data temporarily before it is transmitted to another device or location.

C-brick

The modular controller component of a RAID subsystem, also called a controller brick. It is located in a PC-20. A C-brick can have multiple associated RAID storage components (S-bricks).

C/C++ compilers

Part of the C/C++ Programming Environment, the Cray C and C++ compilers translate C and C++ programs, respectively, into Cray X1 object files. The `cc` (C compiler), `CC` (C++ compiler), `c89` (POSIX 1003.2-1992), and `cpp` (preprocesssor only) commands work with the C and C++ compilers.

cache

In a processing unit, a high-speed buffer that is continually updated to contain recently accessed contents of memory. Its purpose is to reduce access time.

client

A requesting machine. (Normally, a server is the supplying machine.)

co-array

A syntactic extension to Fortran that offers a method for programming data passing; a data object that is identically allocated on each image and can be directly referenced syntactically by any other image.

command line

A text string that is composed of the command name itself, followed by options and arguments to the command. It can be a sequence of nonblank arguments separated by blanks or tabs entered by a user.

Common Installation Tool (CIT)

A graphical user interface (GUI) for loading software. The `cit` and `setup` commands invoke CIT.

Cray Network Subsystem (CNS)

A specialized router (gateway) providing IP (Internet Protocol) network connectivity between a Cray mainframe and customer networks. The CNS also provides Transmission Control Protocol (TCP) assist functionality that can enhance TCP performance in Cray systems.

Cray Open Software (COS)

A collection of public-domain software that is an optional product available for porting to certain Cray systems. COS provides options that are either not available in or behave differently than Cray operating system utilities. Using

COS also makes the transition to a new Cray system easier for programs that already use the utilities included in COS.

Cray Programming Environment Server (CPES)

A server in the Cray X1 system that runs the Programming Environment software.

Cray Storage Management (CRAYSM)

Software on the Cray Workstation (CWS) that is used for configuring the RAID subsystem.

Cray Workstation (CWS)

The system operation, administration, and maintenance workstation or workstations for the Cray X1 system.

CrayDoc

Cray's documentation system for accessing and searching Cray books, man pages, and glossary terms in HTML and/or PDF format from a web browser. CrayDoc runs on any operating system based on a UNIX or Linux operating system.

CrayPat

The primary high-level tool for identifying opportunities for optimization on the Cray X1 system. CrayPat allows you to perform profiling, sampling, and tracing experiments on an instrumented application and to analyze the results of those experiments; no recompilation is needed to produce the instrumented program. In addition, the CrayPat tool provides access to all hardware performance counters.

Customer Service subnetwork

The Cray X1 system internal Ethernet network that connects the Cray Workstation to the access point to be used by Cray Customer Service for remote access to your Cray Workstation. It requires an Ethernet connection for an ISDN router.

data type

A means to categorize data and determine which operations can be applied to the data to get desired results. The Fortran language provides five intrinsic data types: real, integer, complex, logical, and character, and it lets you define additional types.

deferred implementation

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

distributed memory

1. Memory in which each processor has a separate share of the total memory. 2. Memory that is physically distributed among several modules.

Domain Name System (DNS)

An Internet service that translates domain names into IP addresses.

environment

1. The set of hardware and software products on which an application is being run. 2. A set of values supported by the shell used to pass information between processes (specifically, from parent to child).

environment variable

A variable that stores a string of characters for use by your shell and the processes that execute under the shell. Some environment variables are predefined by the shell, and others are defined by an application or user. Shell-level environment variables let you specify the search path that the shell uses to locate executable files, the shell prompt, and many other characteristics of the operation of your shell. Most environment variables are described in the ENVIRONMENT VARIABLES section of the man page for the affected command.

Etnus TotalView

A symbolic source-level debugger designed for debugging the multiple processes of parallel Fortran, C, or C++ programs.

experiment

A predefined entity that specifies the type of performance data to collect.

file system

A collection of files and meta information about the files so that they can be accessed and controlled. A file system is mounted to connect it to the overall file system hierarchy and to make it accessible. The root file system is always mounted; other file systems are mounted as needed.

flexible file I/O (FFIO)

A method of I/O, sometimes called layered I/O, wherein each processing step requests one I/O layer or grouping of layers. A layer refers to the specific type of processing being done. In some cases, the name corresponds directly to the name of one layer. In other cases, however, specifying one layer invokes the routines used to pass the data through multiple layers.

flexible mode

An application may execute in one of two modes: accelerated or flexible. Applications running in flexible mode may run on noncontiguous nodes; they perform in a less predictable amount of processor time than applications running in accelerated mode due to the exclusive use of source processor address translation. See also *accelerated mode.*

floating point

A method of representing data that contains a decimal point.

fragmentation

The process in which computer memory allocation results in scattered blocks of used and free memory. The same problem also can occur on block storage data devices.

function

In C, any called code; in Fortran, a sequence of instructions that returns a value.

functional unit

1. A combination of elements grouped to perform a generally elementary computer function. 2. Hardware within a processor that performs specialized functions. Functional units perform arithmetic, logical, and shift operations on data items.

gateway

A function, normally hardware, that translates protocols so that otherwise disparate entities can communicate.

Gigabit Ethernet

An IEEE 802.3 transmission standard and the implementation of that standard that provides a data rate of 1 Gbps (referred to as 1000Base-T).

granular privilege mechanism

A security feature that divides the power of the superuser into discrete units of privilege called capabilities.

I/O

1. Input/output; the transfer of data between devices. 2. The hardware that stores or moves data to/from a processing system.

I/O Channel Adapter (IOCA)

A field programmable gate array, resident on an IOB, that converts proprietary System Port Channel transactions into standard PCI-X bus transactions.

I/O drawer (IOD)

The hardware enclosure that contains two I/O boards (IOBs) and a Cray L1 controller assembly. Each I/O drawer supports System Port Channels (SPCs) to the Cray X1 mainframe cabinet and Fibre Channel connections to I/O devices. (Also known as an I/O module.)

instruction set

The low-level, basic set of the instructions executed by a computer.

internetwork

Two or more local-area networks (LANs) interconnected through a host or a router acting as a gateway.

LAPACK

A public-domain library of subroutines for solving dense linear algebra problems, including systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems.

latency

1. The time required for a requested disk sector to be positioned under the head on a disk. It is usually stated as average latency, which is the time of one-half revolution. 2. The period of time that starts when a processor requests data and ends when the data is available. 3. The time it takes for a packet to cross a network connection, from sender to receiver. 4. The period of time that a frame is held by a network device before it is forwarded.

library

In software, a collection of routines and functions stored in one or more files with the operating system or programming environment. Programming environment libraries link with programs, either at run time, loading, or compiling, to form a complete executable.

load

To create a binary executable file (an executable) from a binary relocatable object file (the object). This process adds library subprograms to the object and resolves the external references among subprograms. Executable files and the libraries and data they access are loaded into memory during the load step. Links are created among modules that must access each other. The command that performs a load is called a link-edit loader, or simply a loader.

load balancing

A process for ensuring that each processor in a job performs equal work.

loader

A generic term for the system software product that loads a compiled or assembled program into memory and prepares it for execution.

message passing

A programming method in which explicit messages (containing data) are sent between tasks. Cray has implemented the message-passing programming method through the Message Passing Interface and the shared memory (SHMEM) routines.

Message Passing Interface (MPI)

A widely accepted standard for communication among nodes that run a parallel program on a distributed-memory system. MPI is a library of routines that can be called from Fortran and C programs.

Message Passing Toolkit (MPT)

A Cray product that consists of the Message Passing Interface and shared distributed memory (SHMEM) data-passing routines.

module file

A metafile that defines information specific to an application or collection of applications. (This term is not related to the module statement of the Fortran language; it is related to setting up the Cray X1 system environment.) For example, to define the paths, command names, and other environment variables to use the Cray Programming Environment, you use the module file `PrgEnv`, which contains the base information needed for application compilations. The module file `mpt` sets a number of environment variables needed for message passing and data passing application development.

Modules

A package on the Cray X1 system that allows 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 X1 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 others.

MSP mode (multistreaming mode)

Processing where an SSP operates as part of a multistreaming processor to execute parts of a parallel program that have been assigned to it.

multichip module (MCM)

The packaging that contains a multistreaming processor (MSP) and resides on a node module assembly. The MCM contains four processor chips (P-chips), four cache chips (E-chips), and I/O connections (two I-chips).

multistreaming processor (MSP)

A basic programmable computational unit of a Cray X1 system. Each MSP is analogous to a traditional processor and is composed of four single-streaming processors (SSPs) and E-cache that is shared by the SSPs. See also *node*; *SSP*; *MSP mode*; *SSP mode*.

network file system (NFS)

A file system that is exported from a remote server and mounted on other hosts across a network.

Network Time Protocol (NTP)

The protocol that allows synchronizing the clocks in the System Controller and Cray mainframe by pointing the mainframe clocks to the CWS. NTP is used by the CWS to keep the Cray L1 controllers synchronized. This provides consistent time stamps among the various logs for use in diagnosing system problems.

node

The configurable scalable building block for a Cray X1 mainframe. The actual hardware contents of a node are housed in four multichip modules (MCMs). This is the conceptual or software configuration view of a hardware unit called a node module. Physically, all nodes are the same; software controls how a node is used, such as for an OS node, application node, or support node. See also *application node*; *MCM*, *MSP*, *node module*; *OS node*; *SSP*; *support node*.

node flavor

Physically, all nodes are the same; software controls how a node is used. A node's software-assigned flavor dictates the kind of processes and threads that can use its resources. The three assignable node flavors are application, OS, and support. See also *application node*; *OS node*; and *support node*.

node module

The physical node in a Cray X1 system. See *node*.

OpenMP

An industry-standard, portable model for shared memory parallel programming.

OS node

The node that provides kernel-level services, such as system calls, to all support nodes and application nodes. See also *application node*; *node*; *support node*.

page size

The unit of memory addressable through the Translation Lookaside Buffer (TLB). On a Cray X1 system, the base page size is 65,536 bytes, but larger page sizes (up to 4,294,967,296 bytes) are also available.

partition

A portion of a computer system that can be independently operated, booted, dumped, and so on without impact on other running partitions.

PC-20

The peripheral cabinet that can contain either RAID C-bricks and RAID S-bricks or a Cray Programming Environment Server (CPES), peripherals for the CPES, and the Cray Network Subsystem (CNS).

Private Administration subnetwork

The Cray X1 system internal Ethernet network that connects the Cray Workstation to the Cray Programmer Environment Server and the Cray Network Subsystem. This network provides access to the hardware that is used to configure and operate the programming environment and a customer's private network.

psched

The Cray X1 application placement tool. It provides job placement, load balancing, and gang scheduling for all applications placed on application nodes.

RAID

A standard term for a disk configuration that uses multiple drive spindles to provide data redundancy and / or error correction. There are seven levels of RAID architecture, 0 through 6, of which RAID-3 and RAID-5 are the most common. RAID-3 and RAID-5 each use one extra disk to store parity information needed to recreate data in the event of a single disk failure. RAID-3 uses a dedicated parity disk and is typically faster for throughput-oriented applications, such as file transfer and other sequential applications. RAID-5 distributes the parity information across all disks in the array and is typically faster for transaction processing and other random access applications.

RAID subnetwork

The Cray X1 system internal Ethernet that connects the CWS and the RAID subsystem. The Cray storage management software (CRAYSM) uses this network to configure and monitor the RAID subsystem.

RAID subsystem

The Cray X1 disk storage system, including RAID controllers, C-bricks, S-bricks, and the Fibre Channel using SCSI protocol paths from the I/O chassis to the various devices handling I/O.

Rambus

1. A company that develops and licenses memory technology. 2. The memory chips and memory technology that Rambus developed.

reboot

The process of stopping and booting the operating system from the hard drives at the point at which the failure occurred; usually occurs by human intervention.

router

A device that decodes and passes network-layer packets between different nodes on a network. Its purpose is to provide the physical and logical route from one network to another.

router module

Physical hardware that connects node modules.

routine

A section of a program that performs a particular task. It could be a function or a subroutine in Fortran. In C and C++, it is a function.

routing

The process of finding a path, or route, for data to travel from source to destination.

S-brick

The modular, scalable storage component of a RAID subsystem, also called a storage brick. It is located in a PC-20. Each S-brick contains a number of Fibre Channel disk drives. The number of drives per S-brick and the size of each disk varies with the different S-brick modules.

scalability

The ability to increase the resources of a system and keep the work accomplished proportional.

scalable

Capable of being increased in size, or capable of delivering an increase in performance that is proportional to an increase in size.

scheduling

The compiler process of deciding the order of the calculations in a program and which processes will execute them.

SHMEM

A library of optimized functions and subroutines that take advantage of shared memory to move data between the memories of processors. The routines can either be used by themselves or in conjunction with another programming style such as Message Passing Interface.

single-streaming processor (SSP)

A basic programmable computational unit of a Cray X1 system. See also *node*; *MSP*; *MSP mode*; *SSP mode*.

socket

1. A feature of the operating system that allows a process to access the Internet. A process opens a socket, specifies the service it needs, binds the socket to a specific destination, and then sends or receives data. 2. An interprocess communication endpoint; a software connection that links a port on a computer system to a network.

SSP mode (single-streaming mode)

Processing where an SSP executes complete programs. See also *MSP mode*.

superuser

The user, usually the system administrator, who is logged in as root. Most system administration is performed while the system administrator is logged in as root. This account is different from an ordinary user account because root has access to all system files and is not constrained by the usual system of permissions that control access to files, directories, and programs.

support node

The node that is used to run serial commands, such as shells, editors, and other user commands (`ls`, for example). See also *application node*; *OS node*; *node*.

switch

In networks, a device that filters and forwards packets between LAN (local area network) segments.

syntax

A formal description of how to enter a command or a programming language statement.

System Activity Reporter

The function that is used to track the use of processor time, number and type of processors, memory, and I/O to secondary storage and over network connections. It is invoked with the `sar` command.

system node

A Cray X1 system requires a minimum of two nodes. Each system typically uses at least one node that is designated as both an OS node and a support node; this node is often called a system node; however, there is no node flavor of "system." See also *application node*; *OS node*; *support node*; and *node flavor*.

System Port Channel (SPC)

The proprietary I/O channel that provides I/O access to the Cray X1 system by connecting the I-chip to an I/O Channel Adapter on an I/O board.

telnet

A TCP/IP utility that provides remote login between hosts on a network. It is a virtual terminal protocol in the Internet suite of protocols that allows users of one host to log in to a remote host and interact as terminal users of that host. The `telnet`(1) command invokes no operating system requirement. Also refer to the `rlogin`(1) man page.

thrashing

A phenomenon that occurs when a fixed quantity of a reusable resource is allocated on a least-recently-used basis but the cycle length of the reuse is larger than the quantity of the resource, thereby ensuring that reuse never occurs. When applied to cache, it can slow the execution of a program.

thread

The active entity of execution. A sequence of instructions together with machine context (processor registers) and a stack. On a parallel system, multiple threads can be executing parts of a program at the same time.

TLB miss

A memory delay that occurs when a memory location is referenced and the page that contains that memory location does not have an entry in the appropriate translation lookaside buffer.

UNICOS

The operating system for Cray SV1 series systems.

UNICOS/mk

The operating system for Cray T3E systems.

UNICOS/mp

The operating system for Cray X1 systems.

uniform memory access (UMA)

A system in which any memory element can be read from or written to in the same, constant time.

vector

An array, or a subset of an array, on which a computer operates. When arithmetic, logical, or memory operations are applied to vectors, it is referred to as vector processing.

vector instruction

An instruction that operates on a series of elements by repeating the same function and producing a series of results; eliminates instruction startup time for all of the operands but the first.

vector processing

A technique whereby operations are performed simultaneously on the elements of an array instead of iteratively.

vectorization

A form of processing that uses one instruction for the simultaneous performance of iterative operations on elements in sets of ordered data.

XFS file system

The standard UNICOS/mp file system on the Cray X1 mainframe; a journaling, mature file system with support for very large individual files and file systems.

XLV logical volume

An object that behaves like a disk partition, but whose storage can span several physical disk devices. Logical volumes behave like regular disk partitions; they appear as block and character devices in the `/dev` directory and can be used as arguments anywhere a disk device can be specified. XLV logical volumes contain volumes, subvolumes, and volume elements.

Index