

# Cray SV1™ Series SuperCluster® Administrator's Guide

S-2253-10008

---

Copyright © 1999, 2000 Cray Inc. All Rights Reserved. This document or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Cray Inc.

---

#### LIMITED AND RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the Rights in Data clause at FAR 52.227-14 and/or in similar or successor clauses in the FAR, or in the DOD, DOE or NASA FAR Supplements. Unpublished rights reserved under the Copyright Laws of the United States.

---

Autotasking, CF77, Cray, Cray Ada, Cray Channels, Cray Chips, CraySoft, Cray Y-MP, Cray-1, CRInform, CRI/*TurboKiva*, HSX, LibSci, MPP Apprentice, SSD, SUPERCLUSTER, UNICOS, UNICOS/mk, and X-MP EA, are federally registered trademarks and Because no workstation is an island, CCI, CCMT, CF90, CFT, CFT2, CFT77, ConCurrent Maintenance Tools, COS, Cray Animation Theater, Cray APP, Cray C90, Cray C90D, Cray CF90, Cray C++ Compiling System, CrayDoc, Cray EL, Cray J90, Cray J90se, Cray J916, Cray J932, CrayLink, Cray NQS, Cray/REELlibrarian, Cray S-MP, Cray SSD-T90, Cray SV1, Cray T90, Cray T94, Cray T916, Cray T932, Cray T3D, Cray T3D MC, Cray T3D MCA, Cray T3D SC, Cray T3E, CrayTutor, Cray X-MP, Cray XMS, Cray-2, CSIM, CVT, Delivering the power . . ., DGauss, Docview, EMDS, GigaRing, HEXAR, IOS, ND Series Network Disk Array, Network Queuing Environment, Network Queuing Tools, OLNET, RQS, SEGLDR, SMARTE, SUPERLINK, System Maintenance and Remote Testing Environment, Trusted UNICOS, and UNICOS MAX are trademarks of Cray Inc.

---

DynaText and DynaWeb are registered trademarks of Enigma. ESCON is a trademark of International Business Machines Corporation. Kerberos is a trademark of Massachusetts Institute of Technology. LSF is a trademark of Platform Computing Corporation. NFS, ONC+, Solaris, and Sun are trademarks of Sun Microsystems, Inc. OSF and Open Software Foundation are trademarks of Open Software Foundation, Inc. PerfAcct and PerfStat are trademarks of Instrumental, Inc. PostScript is a trademark of Adobe Systems, Inc. TeamQuest Baseline is a trademark of TeamQuest Corporation. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited. X/Open is a trademark of X/Open Company Ltd. The X device is a trademark of the Open Group.

---

The UNICOS operating system is derived from UNIX® System V. The UNICOS operating system is also based in part on the Fourth Berkeley Software Distribution (BSD) under license from The Regents of the University of California.

---

## **New Features**

*Cray SV1™ Series SuperCluster® Administrator's Guide*

S-2253-10008

This guide contains the following new or changed information for the UNICOS 10.0.0.8 release:

- UDB commands issue the common UDB commands with the SuperCluster options.
- Updated the configuration file text in Section A.2.4 of Appendix A.
- Miscellaneous minor corrections were made throughout the book.



# Record of Revision

---

<i><b>Version</b></i>	<i><b>Description</b></i>
001	April 1999 Original Printing. This printing supports the UNICOS 10.0.0.5 release.
002	July 1999 This printing supports the UNICOS 10.0.0.6 release.
003	February 2000 This printing supports the UNICOS 10.0.0.7 release.
10008	November 2000 This printing supports the UNICOS 10.0.0.8 release.



# Contents

---

	<i>Page</i>
<b>Preface</b>	<b>ix</b>
Related Publications . . . . .	ix
Obtaining Publications . . . . .	xi
Conventions . . . . .	xi
Reader Comments . . . . .	xiii
<b>Introduction [1]</b>	<b>1</b>
Cray SV1 Series SuperCluster System Overview . . . . .	1
Hardware Components . . . . .	2
System Workstation (SWS) . . . . .	3
Scalable I/O (SIO): GigaRing Channel . . . . .	4
System Configuration . . . . .	5
Software Components . . . . .	8
UNICOS Operating System . . . . .	8
Cray SV1 Series Cluster Bundle . . . . .	9
<b>SWS Configuration [2]</b>	<b>11</b>
Configuring GigaRing Topology Files on the SWS . . . . .	11
Allocating GigaRing and Node Numbers . . . . .	13
Topology File Rules . . . . .	13
GigaRing Numbering Scheme . . . . .	14
GigaRing/Node Allocation Example . . . . .	15
Domains . . . . .	17
Configuring Options Files on the SWS . . . . .	17
Validating Configuration Changes to the Options and Topology Files . . . . .	18
<b>S-2253-10008</b>	<b>iii</b>

	<i>Page</i>
<b>SWS Operations [3]</b>	<b>19</b>
Opening Mainframe Consoles . . . . .	19
Using Automatic Recovery . . . . .	20
Booting System Components . . . . .	20
Booting Everything . . . . .	21
Booting Domains . . . . .	21
Booting Individual Mainframes Within the Cluster . . . . .	22
Bringing Up Multiuser Mode . . . . .	23
Dumping Mainframes and I/O Nodes . . . . .	24
Dumping All Mainframes and I/O Nodes . . . . .	25
Dumping Domains . . . . .	25
Shutting Down Mainframes . . . . .	26
Shutting Down the SuperCluster . . . . .	27
Shutting Down Individual Mainframes within the Cluster . . . . .	27
Halting Mainframes . . . . .	28
<b>Mainframe Configuration [4]</b>	<b>29</b>
File System Layout . . . . .	29
File System Strategies . . . . .	30
Sharing File Systems . . . . .	31
File Synchronization . . . . .	31
Overview . . . . .	31
Setup Procedure . . . . .	33
Execution Procedure . . . . .	34
Configuring Mainframe Operating System Parameter Files . . . . .	35
Defining the GigaRing Host-to-Host Connection . . . . .	35
Determining and Setting the Number of mbufs . . . . .	36
Configuring the Cluster UDB Feature . . . . .	36



	<i>Page</i>
Installation Process . . . . .	37
Configuration Procedure . . . . .	37
Initialization Procedure . . . . .	39
Setting Up the TCP/IP Network . . . . .	39
IP Addressing Considerations . . . . .	42
IP Address-to-Hardware Address Mapping . . . . .	45
Routing Within the Cluster . . . . .	46
Controlling Startup With /etc/config/rcoptions and rc.* Files . . . . .	48
Controlling Shutdown With /etc/shutdown.* Files . . . . .	50
/etc/shutdown.pre Script . . . . .	51
Configuring Cray SV1 Series Cluster Bundle Software Products . . . . .	51
NQE . . . . .	51
NFS/BDS . . . . .	51
MPI . . . . .	52
DMF . . . . .	52
<b>Mainframe Operations [5]</b>	<b>53</b>
Maintaining the User Database . . . . .	53
Node Level UDB Tools . . . . .	54
Cluster Level UDB Tools . . . . .	54
UDB Source Editing Tool . . . . .	55
UDB Source Collection Tool . . . . .	56
UDB Source Merging Tool . . . . .	57
UDB Cluster Updating Tool . . . . .	57
UDB Server . . . . .	59
UDB Helper . . . . .	59
Batch Scheduling and Load Balancing . . . . .	59
Centralized Accounting . . . . .	60
Backing Up and Restoring File Systems . . . . .	60

	<i>Page</i>
<b>Monitoring and Online Diagnostics From the SWS [6]</b>	<b>61</b>
Monitoring the State of Your Cluster . . . . .	61
Viewing the System State . . . . .	62
Folding and Masking the GigaRing . . . . .	64
Message Logs . . . . .	64
Preparing the System to Run Offline Diagnostics . . . . .	65
<b>Appendix A System Configuration File Examples</b>	<b>67</b>
/etc/config/param File . . . . .	67
/etc/config/udb/clusterUDB.conf File . . . . .	73
General Directives . . . . .	73
Record Merging Directives . . . . .	76
New Record Template Directive . . . . .	77
Default Cluster UDB Configuration File . . . . .	77
<b>Appendix B Glossary</b>	<b>83</b>
<b>Index</b>	<b>89</b>
<b>Figures</b>	
Figure 1. Cray SV1–4 SuperCluster System . . . . .	2
Figure 2. GigaRing Channel . . . . .	4
Figure 3. Direct Connection Configuration (Cray SV1–4 System) . . . . .	6
Figure 4. Matrix Connection Configuration (Cray SV1–8 System) . . . . .	7
Figure 5. Example Topology File . . . . .	12
Figure 6. Cray SV1 series–12 GigaRing/Node Allocation Example . . . . .	16
Figure 7. Smallest SuperCluster GigaRing Interfaces . . . . .	40
Figure 8. Large SuperCluster GigaRing Interfaces . . . . .	41

	<i>Page</i>
Figure 9. Routing Within the Cluster . . . . .	47
<b>Tables</b>	
Table 1. Recommended File Systems Local to Each Cluster Node . . . . .	30
Table 2. Recommended File Systems Shared Within a Cluster . . . . .	31
Table 3. Internet Addressing . . . . .	44
Table 4. Default State and Color Representation . . . . .	63



# Preface

---

This guide is written for system administrators of Cray SV1 series SuperCluster systems. It includes information specific to the system administration of Cray SV1 series SuperCluster systems. This guide does not cover basic system administration. For information on basic mainframe system administration topics, refer to the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*. For information on basic SWS system administration topics, refer to the *SWS-ION Administration and Operations Guide*.



**Warning:** Starting with the UNICOS 10.0 release, the term *Cray ML-Safe* replaces the term *Trusted UNICOS*, which referred to the system configuration used to achieve the UNICOS 8.0.2 release evaluation. Because of changes to available software, hardware, and system configurations since the UNICOS 8.0.2 system release, the term *Cray ML-Safe* does not imply an evaluated product, but refers to the currently available system configuration that closely resembles that of the evaluated Trusted UNICOS 8.0.2 system.

For the UNICOS 10.0 release, the functionality of the Trusted UNICOS system has been retained, but the `CONFIG_TRUSTED` option, which enforces conformance to the strict B1 configuration, is no longer available.

## Related Publications

Information on the structure and operation of a Cray computer system running the UNICOS operating system, as well as information on administering various products that run under the UNICOS system, is contained in the following documents:

- *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems* contains information on basic system administration.
- *UNICOS Installation Guide for Cray SV1 SuperCluster Systems* describes how to install the UNICOS operating system on Cray SV1 series SuperCluster systems.
- *UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems* describes how to install the UNICOS operating system on Cray J90se and Cray SV1 series GigaRing based systems.
- *SWS-ION Administration and Operations Guide* contains information on the administrative and operational procedures that pertain to SWS-ION software.

- *NQE Administration* describes how to configure, monitor, and control the Network Queuing Environment (NQE) running on a UNIX system.
- *General UNICOS System Administration* contains information on performing basic administration tasks as well as information about system and security administration using the UNICOS multilevel (MLS) feature. This publication contains chapters documenting file system planning, UNICOS startup and shutdown procedures, file system maintenance, basic administration tools, crash and dump analysis, the UNICOS multilevel security (MLS) feature, and administration of online features.
- *UNICOS Resource Administration* contains information on the administration of various UNICOS features available to all UNICOS systems. This publication contains chapters documenting accounting, automatic incident reporting (AIR), the fair-share scheduler, file system quotas, file system monitoring, system activity and performance monitoring, and the Unified Resource Manager (URM).
- *UNICOS Configuration Administrator's Guide* provides information about the UNICOS kernel configuration files and the run-time configuration files and scripts.
- *UNICOS Configuration Administrator's Guide* contains information on administration of networking facilities supported by the UNICOS operating system. This publication contains chapters documenting TCP/IP for the UNICOS operating system, the UNICOS network file system (NFS) feature, and the network information system (NIS) feature.
- *Kerberos Administrator's Guide* contains information on administration of the Kerberos feature, a set of programs and libraries that provide distributed authentication over an open network. This publication contains chapters documenting Kerberos implementation, configuration, and troubleshooting.
- *Tape Subsystem Administration* contains information on administration of UNICOS and UNICOS/mk tape subsystems. This publication contains chapters documenting tape subsystem administration commands, tape configuration, administration issues, and tape troubleshooting.

The following man page manuals contain additional information that may be helpful.

**Note:** For the UNICOS 10.0 release, man page reference manuals are not orderable in printed book form. Instead, they are available as printable PostScript files provided on the same DynaWeb CD as the rest of the supporting documents for this release. Individual man pages are still available online and can be accessed by using the `man(1)` command.

- *UNICOS User Commands Reference Manual*
- *UNICOS System Calls Reference Manual*
- *UNICOS File Formats and Special Files Reference Manual*
- *UNICOS Administrator Commands Reference Manual*
- *UNICOS System Libraries Reference Manual*
- *SWS-ION Reference Manual*

## Obtaining Publications

The *User Publications Catalog* describes the availability and content of all Cray hardware and software documents that are available to customers. Customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, call +1-651-605-9100. Cray employees may send e-mail to `orderdsk@cray.com`

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

## Conventions

The following conventions are used throughout this document:

<u>Convention</u>	<u>Meaning</u>
<code>command</code>	This fixed-space font denotes literal items (such as commands, files, routines, pathnames, signals, messages, programming language structures, and e-mail addresses) and items that appear on the screen.

manpage(*x*)

Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers:

1	User commands
1B	User commands ported from BSD
2	System calls
3	Library routines, macros, and opdefs
4	Devices (special files)
4P	Protocols
5	File formats
7	Miscellaneous topics
7D	DWB-related information
8	Administrator commands

Some internal routines (for example, the `_assign_asgcmd_info()` routine) do not have man pages associated with them.

*variable*

Italic typeface denotes variable entries and words or concepts being defined.

**user input**

This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.

[ ]

Brackets enclose optional portions of a command or directive line.

...

Ellipses indicate that a preceding element can be repeated.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.



Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

## Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and number of the document with your comments.

You can contact us in any of the following ways:

- Send e-mail to the following address:

`pubs@cray.com`

- Send a fax to the attention of “Software Publications” at: +1-651-605-9001.
- File an SPR; use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.
- Call the Software Publications Group, through the Cray Support Center, using one of the following numbers: 1-800-950-2729 (toll free from the United States and Canada) or +1-651-605-8805.
- Send mail to the following address:

Software Publications  
Cray Inc.  
1340 Mendota Heights Road  
Mendota Heights, MN 55120

We value your comments and will respond to them promptly.



# Introduction [1]

---

This chapter discusses the following Cray SV1 series SuperCluster system topics:

- System overview
- Hardware components
- Software components

Before you perform any of the procedures in this guide, you must first boot your scalable I/O (SIO) and UNICOS software in multiuser mode. See the *UNICOS Installation Guide for Cray SV1 SuperCluster Systems* for more information.

**Note:** This guide does not cover basic system administration in detail. It includes only information necessary for the system administration of Cray SV1 series SuperCluster systems. For information on basic mainframe administration topics, see the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*. For information on basic SWS administration topics, see the *SWS-ION Administration and Operations Guide*.

## 1.1 Cray SV1 Series SuperCluster System Overview

The Cray SV1 series SuperCluster system is a high-performance supercomputer designed as a *cluster*, a collection of multiple mainframes coupled to each other by interconnects. It is software and hardware compatible with other parallel vector Cray supercomputers. The Cray SV1 series SuperCluster system runs the UNICOS operating system that was developed specifically for Cray supercomputers.

The significant capabilities of the Cray SV1 series SuperCluster system include:

- High-speed scalar, dual-pipe vector, and parallel processing
- A large, fast central memory
- High-performance input/output

This processing power is delivered to the user by autovectorizing ANSI Standard Fortran, C, and C++ compiling environments.

The Cray SV1 series SuperCluster system is a *capacity cluster* in which resources are managed and allocated across the cluster, but each application runs on one node (this includes redundant resources for restarting applications.) A *node* is

defined as a single UNIX image (an individual computer). A single node may be a member of one or more clusters.

Figure 1 illustrates a Cray SV1 series-4 SuperCluster system, which contains four Cray SV1 series mainframes.

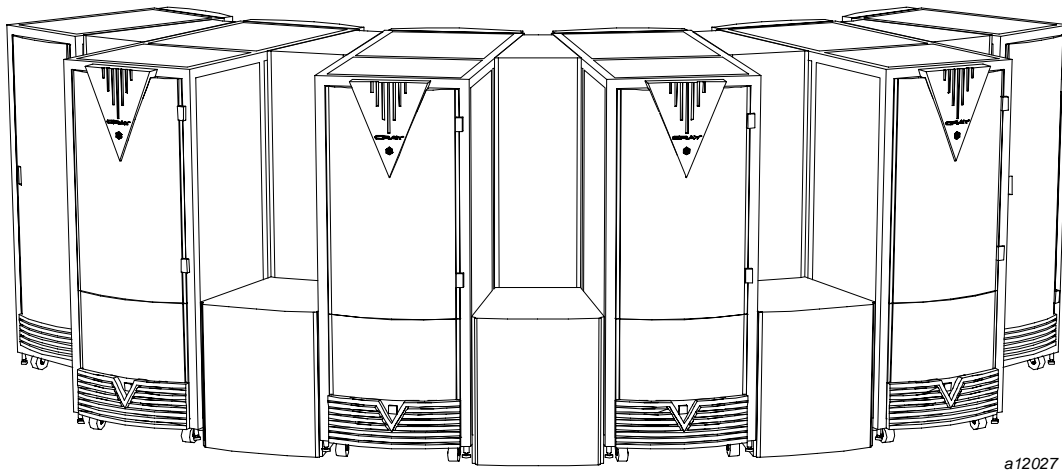


Figure 1. Cray SV1-4 SuperCluster System

## 1.2 Hardware Components

The minimum hardware configuration for each Cray SV1 series SuperCluster system is a Cray SV1 series-4 system, which is referred to as a *SuperCluster building block*. Each SuperCluster building block includes the following components:

- A system workstation (SWS)
- Four Cray SV1 series mainframes (four processing cabinets) with two peripheral cabinets
- Channel connection hardware (module connectors and cables) that comprise up to five scalable I/O (SIO) GigaRing channels:
  - One private GigaRing channel per mainframe, each with an I/O node (ION), for a total of four channels.

- One GigaRing channel per SuperCluster building block for intra-node communication, connected to all four mainframes with an ION.

One GigaRing connection is optionally available per CPU module. A Cray SV1 series system supports up to eight CPU modules, so up to eight GigaRing connections are available per mainframe. Two GigaRing connections are used per mainframe in a Cray SV1 series-4 configuration, and three GigaRing connections per mainframe are used in the minimum Cray SV1 series-8 through Cray SV1 series-32 configurations.

The number of additional GigaRing channels accessible from each fully populated system depends on how many CPUs are configured in the system, up to a maximum of six additional GigaRing channel connections per Cray SV1 series system (with 32 CPUs in the system).

Cray SV1 series-8 or larger systems also include one GigaRing channel per mainframe for inter-node communication (for a total of three channels). (Note that each mainframe is limited to eight GigaRing channels.)

Each Cray SV1 series processing cabinet within a SuperCluster building block is also called a Cray SV1 series *node*. Each SuperCluster building block can contain up to four nodes. A Cray SV1 series node has 2 to 32 Gbytes of main memory and 8 to 32 CPUs. Each Cray SV1 series CPU includes a 256-Kbyte, four-way associative cache for all scalar and vector data. Each node contains one 8 x 8 backplane with eight memory modules and from two to eight processor modules.

A Cray SV1 series SuperCluster system is formed by combining 4 to 32 Cray SV1 series nodes into a system with up to 8 SuperCluster building blocks or 32 processor cabinets with an appropriate number of peripheral cabinets. For example, to create a Cray SV1 series-8 SuperCluster system, two SuperCluster building blocks are combined, with inter-node GigaRing connections between specific systems in the SuperCluster building block.

### 1.2.1 System Workstation (SWS)

The system workstation (SWS) is a Sun workstation running the Solaris operating system.

The workstation serves as the system console for each SIO device and the Cray SV1 series SuperCluster building block. The workstation also runs management and maintenance software for each Cray SV1 series mainframe and SIO devices. This console is connected via a private Ethernet connection to each SIO. It is shipped with a DAT tape device for backups.

### 1.2.2 Scalable I/O (SIO): GigaRing Channel

The Cray scalable I/O (SIO) architecture consists of a system of I/O nodes (IONs) connected by a high-speed system channel called the *GigaRing channel*. This channel connects Cray SV1 series systems to the following GigaRing clients:

- Other Cray SV1 series nodes.
- I/O nodes (IONs) for networks and peripherals such as disks and tapes:
  - Single-purpose nodes (SPNs), which support specific network interfaces and/or devices (HIPPI, ESCON tape, block mux tape, fibre disks, and IPI disks).
  - Multipurpose nodes (MPNs), which provide an interface based on the SBus standard to support industry-standard I/O channels (Ethernet, FDDI, ATM OC-3 and SCSI disks and tapes). There is one MPN per mainframe in the Cray SV1 series SuperCluster.

Figure 2 illustrates the GigaRing channel concept.

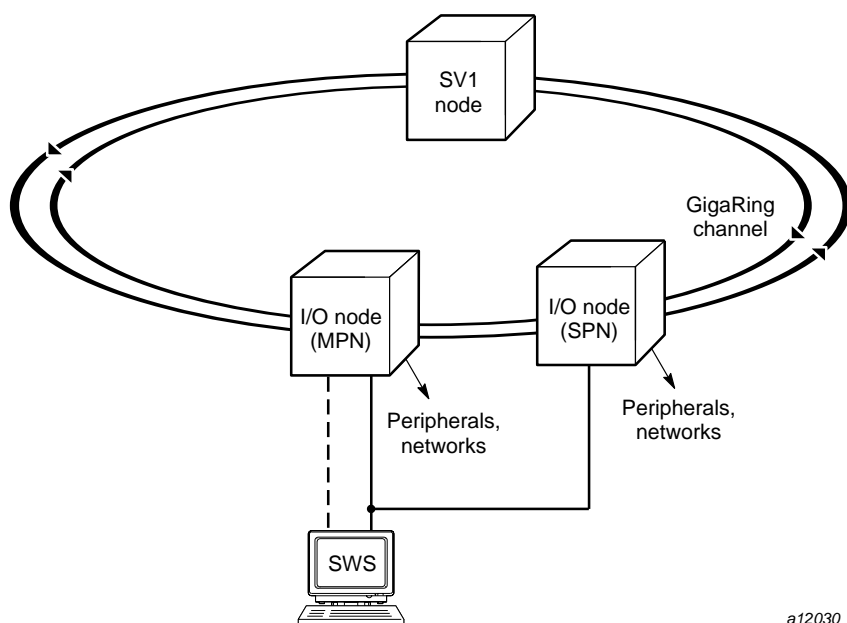


Figure 2. GigaRing Channel

### 1.2.3 System Configuration

The Cray SV1 series SuperCluster system is available in the following configurations (SV1 meaning SV1 series):

Cray SV1-4

Cray SV1-8

Cray SV1-12

Cray SV1-16

Cray SV1-20

Cray SV1-24

Cray SV1-28

Cray SV1-32

Memory size covers a range of from 16-128 Gbytes for a Cray SV1 series-4 SuperCluster system to 128-1024 Gbytes for a Cray SV1 series-32 SuperCluster system.

A four-node SuperCluster system must have a minimum of 8 CPUs per node. Systems larger than four nodes must have a minimum of 12 CPUs per node.

The Cray SV1 series-4 SuperCluster system has a *direct connection* configuration. In this configuration, each mainframe node in the Cray SV1 series SuperCluster system is connected to every other mainframe in the SuperCluster system through shared GigaRing channels, with multiple mainframes on each shared GigaRing channel. Figure 3 illustrates a direct connection configuration.

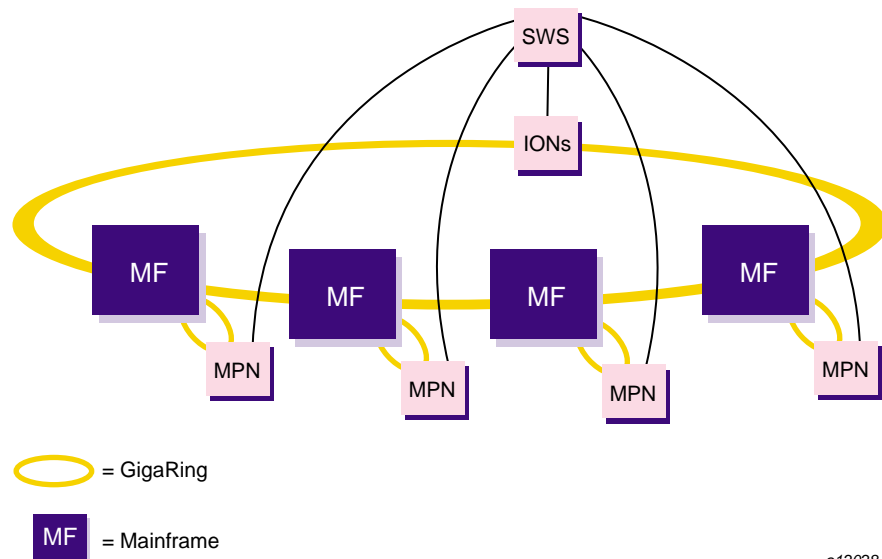


Figure 3. Direct Connection Configuration (Cray SV1-4 System)

All SuperCluster systems larger than the Cray SV1 series-4 system have a *matrix connection* configuration. To access mainframes that are not on the same shared GigaRing channels, one mainframe from the shared GigaRing Internet Protocol (IP) forwards the communication to another GigaRing channel where the target mainframe resides. There is at most one IP forwarding of communication between any two mainframes in the SuperCluster system.

Figure 4, page 7, shows a matrix connection configuration and illustrates the various terms used to describe the GigaRing interconnection between the two Cray SV1 series-4 SuperCluster building blocks in a Cray SV1 series-8 SuperCluster system.



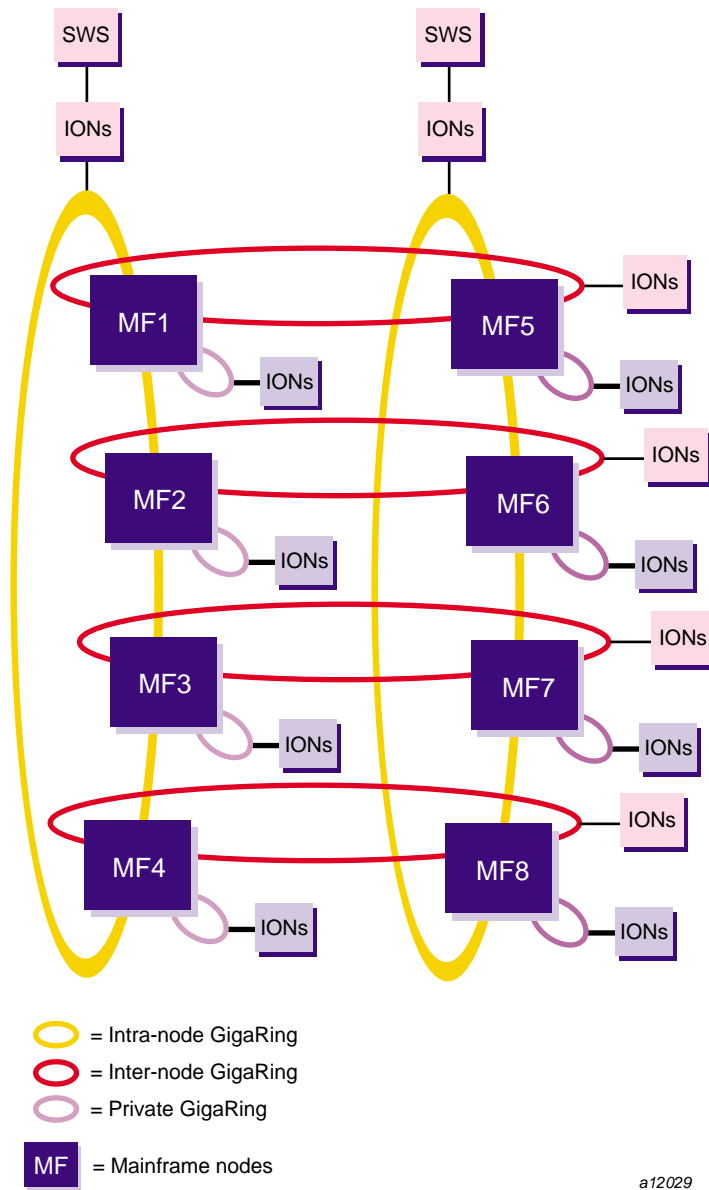


Figure 4. Matrix Connection Configuration (Cray SV1-8 System)

Figure 4, page 7, shows how *intra-node* GigaRing connections are made between the four Cray SV1 series mainframes that make up a SuperCluster building block, while *inter-node* GigaRing connections are made between two or more SuperCluster building blocks. The intra-node and inter-node GigaRing connections are used to provide common disk and network I/O capabilities to the SuperCluster system.

Each mainframe in the SuperCluster system has a private GigaRing connection that contains at least one ION containing the system's primary and secondary `root`, `usr`, and `src` file systems. The private GigaRing connection may contain additional IONs to provide unique disk and network I/O capabilities to the single Cray SV1 series system.

Additional GigaRing channels can be added for private I/O by a node or for shared I/O between two or more nodes. The number of additional GigaRing channels accessible from each fully populated system depends on how many processor modules with channel adapters are configured in the system. There is a maximum of eight processor modules with channel adapters in the node.

The Cray SV1 series-4 system has a minimum of two GigaRing channels, while a Cray SV1 series-8 or larger system has a minimum of three GigaRing channels.

## 1.3 Software Components

The following sections discuss system software components included with the Cray SV1 series SuperCluster system.

**Note:** All software required to run your system is initially installed by Cray. Your consoles, I/O subsystems, and Cray SV1 series mainframes are fully operational when you receive them.

### 1.3.1 UNICOS Operating System

Each Cray SV1 series SuperCluster system node runs the UNICOS operating system, which was developed specifically for Cray supercomputers. The UNICOS operating system is based on the UNIX System V operating system with Berkeley extensions. It is an interactive and batch operating system that offers many features for performance, functionality, application portability, and I/O connectivity.

The UNICOS operating system combines all of the strengths inherent in the UNIX operating system, such as its familiar user interface and production-oriented features. These features include high-performance I/O,

multiprocessing support, ANSI/IBM tape support, resource allocation and control, and enhanced process scheduling.

Concurrent batch and interactive processing is supported via the Network Queuing Environment (NQE) and TCP/IP.

### 1.3.2 Cray SV1 Series Cluster Bundle

The Cray SV1 series Cluster Bundle is a set of software products that provide increased throughput, data availability, and automatic workload distribution between two or more UNICOS systems connected by a network.

Cluster capabilities are provided by the following components:

- NQE for UNICOS

Network Queuing Environment (NQE) for UNICOS is a workload management system that automatically balances tasks across mixed servers attached to a network.

- BDS

Bulk Data Services (BDS) is an enhancement to the network file system (NFS) that handles large file transactions over high-speed networks.

- ONC+

ONC+ is a set of technologies licensed by Sun Microsystems to enable development of distributed applications in a multivendor environment. Its components include the following:

- NFS V3
- NIS+
- Lock Manager V3
- RPC authentication: AUTH KERB

- MPT

The Cray Message Passing Toolkit (MPT) consolidates support for message passing into a single product. MPT consists of optimized versions of libraries and associated software to support the leading message-passing standards, PVM and MPI, as well as one-way data-passing communication via the Cray Shared Memory Library.

Your system may have additional SuperCluster software capabilities, which require separate licenses. These include the following:

- The Cray Data Migration Facility (DMF) provides client/server data migration and hierarchical storage management.
- The Load Sharing Facility (LSF) provides workload distribution and job scheduling.

Several third-party software packages are also available that you may find useful.

# SWS Configuration [2]

---

This chapter discusses the following topics:

- Configuring GigaRing topology files on the SWS
  - Allocating GigaRing and node numbers
  - Domains
- Configuring options files on the SWS
- Validating configuration changes to the options and topology files

**Note:** This chapter does not cover basic SWS configuration in detail. It includes only information necessary for the configuration of Cray SV1 series SuperCluster systems. For information on basic SWS configuration topics, refer to the *SWS-ION Administration and Operations Guide*.

## 2.1 Configuring GigaRing Topology Files on the SWS

The *GigaRing topology file* is an ASCII file consisting of statements that describe the physical layout of mainframe and I/O nodes (IONs) on one or more GigaRing channels. In this context, the term *node* is used to designate both of the following:

- The *GigaRing node chip*, which manages communications on the ring and with a client
- The *client hardware node*, which is the ION or mainframe connected to the GigaRing node chip

(Where appropriate, a distinction between these types of nodes will be made.)

System-level operational commands such as `bootsys(8)`, `dumpsys(8)`, and `haltsys(8)` use the topology file to determine the components of the system that are to be booted, dumped, or halted. A topology file is therefore required when you use these commands.

Your system is preinstalled with a default GigaRing topology file. You should determine whether these files satisfy your hardware configuration and site needs, and modify them if needed. If you perform a reinstallation, you may want to recreate the topology file to your specifications.

The default GigaRing topology file used by system-level commands is specified by the `TOPOLOGY` environment variable. If `TOPOLOGY` is not defined, the default location is `/opt/config/topology`.

In most cases, you will have a single topology file. Only one topology file is used at any one time by operational software and diagnostics.

Figure 5 shows an example GigaRing topology file for a single SuperCluster building block:

```
#####
# Shared Rings
#####

# Intra GigaRing Connections
#####

RING    ring-040          040
SV1     sn3001-1          01    CONNECTION=1
SV1     sn3002-1          02    CONNECTION=1
SV1     sn3003-1          03    CONNECTION=1
SV1     sn3004-1          04    CONNECTION=1
FCN-1   040-fcn0          041    MAINTENANCE

# Inter GigaRing Connections
#####

#####
# Non-Shared GigaRing Connections
#####

# sn3001
#####

RING    ring-sn3001-0      0
SV1     sn3001-0          01    CONNECTION=0 BOOTNODE
MPN-1   sn3001-mpn0       041    MAINTENANCE

# sn3002
#####
```

```

RING    ring-sn3002-0    0
SV1     sn3002-0        02    CONNECTION=0 BOOTNODE
MPN-1   sn3002-mpn0     041    MAINTENANCE

# sn3003
#####

RING    ring-sn3003-0    0
SV1     sn3003-0        03    CONNECTION=0 BOOTNODE
MPN-1   sn3003-mpn0     041    MAINTENANCE

# sn3004
#####

RING    ring-sn3004-0    0
SV1     sn3004-0        04    CONNECTION=0 BOOTNODE
MPN-1   sn3004-mpn0     041    MAINTENANCE

```

Figure 5. Example Topology File

### 2.1.1 Allocating GigaRing and Node Numbers

This section describes the GigaRing and I/O node numbering scheme for Cray SV1 series SuperCluster systems. The scheme is a numbering recommendation to aid in identifying where in the system a given GigaRing address or component name resides.

**Note:** *I/O node* refers to the GigaRing node, and *cluster node* refers to one of the mainframes in the cluster.

#### 2.1.1.1 Topology File Rules

The following items describe ring numbering:

- Ring numbers must be in the range from 0 through 127 (0 - 0177).
- All GigaRing channels connected to a specific mainframe must have different ring numbers. Ring numbers can be reused as long as two rings on the same mainframe are not assigned the same number.

The following items describe I/O node numbering:

- Node numbers must be in the range from 1 through 63 (0 - 077).

- All I/O nodes on a GigaRing channel must have unique numbers. I/O node numbers can be reused on other GigaRing channels.

The following items describe names:

- All names in the topology file must be unique.
- Cluster node names must start with the name of the mainframe followed by a nonalphanumeric character and some other unique character string (*mfname-#*).

For additional information on topology files, see the `topology(5)` man page on the SWS.

#### 2.1.1.2 GigaRing Numbering Scheme

The following items describe GigaRing numbering:

- Private GigaRing ring numbers are set to the mainframe connection to which they are connected (0 - 7).

Private GigaRing channels are rings that are connected to only one mainframe.

- I/O nodes on inter-node GigaRing connections are assigned 020, 021, 022, or 023.
- I/O node addresses 024 through 037 can be used for sites that add additional intra-node GigaRing connections.
- I/O nodes on intra-node GigaRing connections are assigned 040, 041, 042, 043, 044, 045, 046, or 047. The last digit represents the number of the cluster node.

I/O node addresses 050 through 057 can be used for sites that add additional intra-node GigaRing connections.

- A numbering scheme for ring addresses 010 through 019 and 060 through 077 is not defined.

The following items describe I/O node numbering:

- All mainframes are assigned a number based on their location in the cluster. The first cluster mainframes are assigned 1 through 4; the second cluster mainframes are assigned 5 through 8 (010), and so on.



All mainframe GigaRing I/O nodes are assigned the number of the mainframe, 1 through 32 (01 - 040). All GigaRing connections to mainframe one are assigned node address 1.

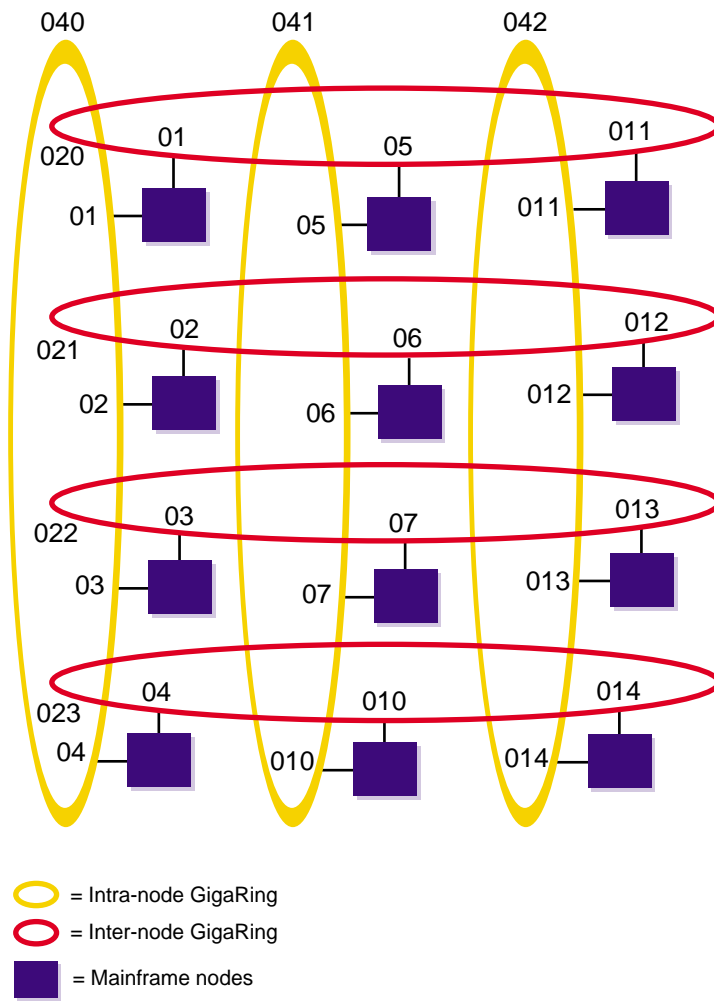
- All I/O nodes are assigned values of 33 through 63 (041 - 077).

The following items describe names:

- Mainframe names are of the form *mfname*-# where *mfname* is the serial number of the mainframe and # is the mainframe connection number.
- I/O node names on the private rings are of the form *mfname-iontype*# where *mfname* is the serial number of the mainframe that attached to the I/O node. *iontype* is the type of I/O node: *mpn*, *fcn*, *ipn*, *bmn*, *esn*, or *hpn*. # is a number that uniquely identifies the I/O node. This number starts at 0 and increments for each I/O node of the same type.
- I/O node names on intra- and inter-node GigaRing connections are of the form *ringnumber-iontype*# where *ringnumber* is the number of the ring on which the I/O node is attached. *iontype* is the type of I/O node: *mpn*, *fcn*, *ipn*, *bmn*, *esn*, or *hpn*. # is a number that uniquely identifies the I/O node. This number starts at 0 and increments for each I/O node of the same type.
- Ring names for shared GigaRing connections are *ring-ringnumber* where *ringnumber* is the number of the ring on which the I/O node is attached.
- Ring names for private GigaRing connections are *ring-mfname-ringnumber* where *mfname* is the serial number of the mainframe. *ringnumber* is the number of the ring on which the ION is attached.

#### 2.1.1.3 GigaRing/Node Allocation Example

Figure 6, page 16, illustrates possible GigaRing and node allocation for a Cray SV1 series-12 system.



a12042

Figure 6. Cray SV1 series-12 GigaRing/Node Allocation Example

The maximum number of mainframe and I/O nodes you can physically configure on a ring is fairly large, but the practical limit is approximately nine connections (for mainframes and I/O nodes).

### 2.1.2 Domains

System-level operational commands such as `bootsys(8)` can execute on the *domain* of specified mainframes. A domain consists of the specified mainframes themselves plus the GigaRing channels and I/O nodes that are connected to only the specified mainframes.

For example, to boot the `sn3001` domain (`sn3001`, `ring-sn3001--0` and `sn3001--mpn0`) (as configured in the sample topology file in Figure 5, page 12), enter the following command:

```
bootsys -d sn3001
```

Note that the shared GigaRing channel (`ring--040`) and the I/O node on the shared ring (`040--fcn0`) are not initialized or booted because other mainframes are on the same ring. In addition, none of the other private rings are processed because `sn3001` is not connected to those rings.

For more detailed information on domains, see the *SWS-ION Administration and Operations Guide* and the `bootsys(8)`, `dumpsys(8)`, and `scantopo(8)` man pages.

## 2.2 Configuring Options Files on the SWS

An *options* file is an ASCII file consisting of statements that provide site-specific defaults to the low-level SWS commands (such as the `bootsvl(8)` command) that are invoked by the system-level SWS commands (such as the `bootsys(8)` command). An options file is required if you use the system-level commands; if you wish to use all of the defaults provided by the low-level commands, the options file may be empty.

Your system is preinstalled with a default options file. You should determine whether this file satisfies your hardware configuration and site needs, and modify it if needed. If you perform an initial installation or reinstallation, you must create the options file.

The default options file is specified by the `OPTIONS` environment variable. If `OPTIONS` is not defined, the default location is `/opt/config/options`.

You can also specify an options file in another location by using the `-o` option to the system commands, such as `bootsys(8)`. For example, you could set up the `/opt/config/options` file to contain the options you want to use in a production environment, and have a second file called `/opt/config/devoptions` that contains options appropriate to a

development environment. You can then boot with the appropriate type of options file.

The following is an example options file:

```
sn3001 boot -p /opt/CYRIos/sn3001/param.prod -u /opt/CYRIos/sn3001/unicos.prod
sn3002 boot -p /opt/CYRIos/sn3002/param.prod -u /opt/CYRIos/sn3002/unicos.prod
sn3003 boot -p /opt/CYRIos/sn3003/param.prod -u /opt/CYRIos/sn3003/unicos.prod
sn3004 boot -p /opt/CYRIos/sn3004/param.prod -u /opt/CYRIos/sn3004/unicos.prod
```

For more detailed information on options files, see the *SWS-ION Administration and Operations Guide* and the `options(5)` man page.

## 2.3 Validating Configuration Changes to the Options and Topology Files

After changing or creating the options and topology files, you should verify that the `bootsys(8)` and `dumpsys(8)` commands will execute properly using them. To do this, use the `-V` option to each of these commands. This option validates that the system-level command and all of the low-level commands executed by it (except for halt commands and the `dring(8)` command) can be executed by the user, and that arguments used by the command are valid and accessible. Command arguments are derived from the command line, the topology file, the options file, and command defaults.

For example, if you have created the files `/tmp/myoptions` and `/tmp/mytopology`, you could execute the following sequence of commands:

```
export OPTIONS=/tmp/myoptions
export TOPOLOGY=/tmp/mytopology
```

If you encounter an error, fix it and repeat the validation process to ensure that there are no other errors.

# SWS Operations [3]

---

This chapter discusses the following topics:

- Opening mainframe consoles
- Using automatic recovery
- Booting system components
- Bringing up multiuser mode
- Dumping mainframes and I/O nodes
- Shutting down mainframes (bringing the UNICOS operating system back to single-user mode)
- Halting mainframes

**Note:** This chapter does not cover basic SWS operations in detail. It includes only information necessary for the operations of Cray SV1 series SuperCluster systems. For information on basic SWS operations topics, refer to the *SWS-ION Administration and Operations Guide*.

The operations described in this chapter require that system components are configured in the GigaRing topology file. Although not required, you may find it useful to set up system defaults in one or more options files. For more information about configuring the topology and options files, see the *SWS-ION Administration and Operations Guide*.

## 3.1 Opening Mainframe Consoles

A console is required to boot a mainframe. To open a persistent console for each mainframe, enter the following `mfcon` command for each mainframe in the SuperCluster system:

```
mfcon -p mfname
```

The *mfname* variable specifies the name of the mainframe.

Another method is to use the `-c` option on the `bootsys(8)` command. For more information on this command, see Section 3.3.3

The `consys` command uses the `mfcon` command to open a console for each mainframe specified. If no mainframes are specified, `consys` opens a console

for each mainframe in the topology file. See the `consys(8)` man page for more information.

**Note:** Note that `consys` does not support the passing of command options to `mfcon`, therefore it is recommended that you use the `mfcon` command.

Cray SV1 series SuperCluster systems also support the `clsh(8)` command, which is a shell script that runs on the SWS. `clsh(8)` allows you to type a command in one SWS window and have it run in several windows. When you type a command in the original window, the command runs in all subwindows, each of which is associated with a node. Results are returned to each node's associated window on the SWS. If you type in one of the subwindows, the command runs only in the subwindow for the associated node. For additional information, see the `clsh(8)` man page.

## 3.2 Using Automatic Recovery

Cray SV1 series SuperCluster systems also support automatic recovery. Automatic recovery enables Cray systems to run in an unattended environment. In the event of a mainframe hang or panic, the automatic recovery capability maximizes system availability for user jobs and keeps the downtime of the mainframe to a minimum. Automatic recovery also enables a Cray system to run during attended operations and have minimal interference with manual operations. For example, if an operator performs a shutdown, automatic recovery will not detect this operation and try and reboot the system. The automatic recovery capability consists of monitoring, error reporting, and notification. For detailed information on using automatic recovery, refer to the *SWS-ION Administration and Operations Guide*.

## 3.3 Booting System Components

You should boot a system component in the following situations:

- After taking a dump image
- After preventive maintenance has been performed
- After changing the configuration

To boot a system component, you should use the `bootsys(8)` command, which boots one or more system components. A system component can be a ring, I/O node (ION), or mainframe.

`bootsys` performs the following actions for all system components specified in the GigaRing topology file:

1. Halts the Cray SV1 series mainframes.
2. Boots the IONs.
3. Initializes the rings.
4. Starts consoles.
5. Boots the mainframes.
6. Initializes Cray SV1 series mainframes.

For a complete list of available options, see the `bootsys(8)` man page. For information about the low-level `bootsv1` command invoked by `bootsys`, see the `bootsv1(8)` man page.



**Caution:** The `bootsys` and `bootsv1` commands do not manage issues related to system shutdown. If you execute these commands on a running mainframe, there may be undesirable consequences, such as loss of data or corruption of file systems. Before executing these commands, you should shut down the UNICOS operating system according to the methods described in the UNICOS administrator documentation.

### 3.3.1 Booting Everything

To initialize all of the GigaRing channels and boot all IONs and mainframes in the GigaRing topology file, enter the following command on each SWS in the SuperCluster system:

```
bootsys
```

After booting your system by executing the `bootsys` command, the UNICOS operating system is in single-user mode (unless the run level is specified in the SWS options file). See Section 2.2, page 17 for more information on configuring the options file.

### 3.3.2 Booting Domains

To boot the domain of specified mainframes, enter the following command on the SWS for that mainframe domain (see Section 2.1.2, page 17 for more information on domains):

```
bootsys -d mfnames
```

The *mfnames* variable specifies the mainframes whose domains should be booted. The *mfnames* value is a comma-separated list of mainframe names. The domain consists of the specified mainframes themselves plus the GigaRing channels and IONs that are connected to only the specified mainframes. If a ring is shared with an unspecified mainframe, that ring and the associated ION will not be booted.

For example, suppose your GigaRing topology file is set up as follows:

- The `sn3001` domain includes `sn3001`, `ring-sn3001--0`, and `sn3001--mpn0`.
- The `sn3002` domain includes `sn3002`, `ring-sn3002--0`, and `sn3002--mpn0`.
- The `sn3001,sn3002` domain includes the super set of `sn3001` and `sn3002` domains.

You could then enter the following command to boot the `sn3001` domain (`sn3001`, `ring-sn3001--0`, and `sn3001--mpn0`):

```
bootsys -d sn3001
```

Enter the following command to boot the `sn3001,sn3002` domain (`sn3001`, `sn3002`, `ring-sn3001--0`, `ring-sn3002--0`, `sn3001--mpn0`, and `sn3002--mpn0`):

```
bootsys -d sn3001,sn3002
```

Enter the following command to boot the `sn3001` domain (`sn3001`, `ring-sn3001--0`, and `sn3001--mpn0`) plus the `sn3002` mainframe (note the white space rather than a comma separating the mainframe names):

```
bootsys -d sn3001 sn3002
```

For more detailed information on domains, see the *SWS-ION Administration and Operations Guide* and the `bootsys(8)`, `dumpsys(8)`, and `scantopo(8)` man pages.

### 3.3.3 Booting Individual Mainframes Within the Cluster

To boot one mainframe configured in the GigaRing topology file, enter the following command on the SWS for that mainframe:

```
bootsys mfname
```



The *mfname* variable specifies the name of the mainframe to be booted.

**Note:** If you have not already used the `mfcon -p mf_name` command and do not have a persistent console running, you may wish to use the `bootsys -c mfname` command. However, this command **does not** boot the IONs or clear the GigaRing channels.

### 3.4 Bringing Up Multiuser Mode

There are two possible methods to bring up the UNICOS operating system to multiuser run mode:

- When booting from a non-running system, by using the `bootsys(8)` command in conjunction with the specification of the run level in the SWS options file. (See Section 3.3.1, page 21, for more information on `bootsys`).
- By using the `levelsys(8)` command (systems must be running or booted to use `levelsys`). This command sets or displays the mainframe operating system run level for multiple mainframes. To set the run level for all mainframes listed in the topology file, enter the following command on each SWS in the SuperCluster system:

```
levelsys -r runlevel
```

For example, to bring up all of the mainframes listed in the topology file to multiuser mode from single-user mode, enter the following command:

```
levelsys -r 2
```

To change the run level to multiuser mode for one mainframe only, enter the following command:

```
levelsys -r 2 mfname
```

See the `levelsys(8)` man page for detailed information on the `levelsys` command.

Multiuser mode is typically run level 2, although you can configure a system to run in multiuser mode at any level between 0 and 6. See the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems* for a detailed discussion of run levels.

### 3.5 Dumping Mainframes and I/O Nodes

If your Cray SV1 series SuperCluster system experiences an operating system panic or hang, you should take a dump image of the common memory of the mainframe or I/O node (ION). These dumps are important and useful for determining probable causes for software or hardware problems. This section explains how to capture memory dumps from the SWS.

When either an ION assertion panic or processor fault panic occurs, a dump of the ION is automatically captured and saved to the SWS in the `/opt/CYRIDump` file. In cases of an apparent hang or unusual system behavior, you must manually perform a multipurpose node (MPN) dump.

To create a dump image of the operating system, you should use the `dumpsys(8)` command, which dumps one or more IONs or mainframe system components.

If your system has been configured with an ION dump device (the recommended method), mainframe memory will be copied to the specified dump device on the ION. (Because the request to dump to the ION comes from the SWS rather than the ION, this method is known as *third-party I/O*. For an overview of the tasks required to enable third-party I/O, see the section “Procedures to Enable Third-Party I/O” in the *SWS-ION Administration and Operations Guide*.)

If your system has not been configured with a dump device, or if you override the configuration, mainframe memory will be copied to the SWS disk. Dumping to the SWS disk is significantly slower.

If you do not specify any options, `dumpsys` performs the following steps for all IONs and mainframes specified in the GigaRing topology file:

1. Halts the mainframes.
2. Dumps the IONs.
3. Boots the MPNs (the SPN is self rebooting).
4. Initializes the GigaRings.
5. Dumps the Cray SV1 series mainframes.

For a complete list of `dumpsys` options, see the `dumpsys(8)` man page. For information about the low-level commands invoked by `dumpsys`, see the `dumpsv1(8)` man page.



**Caution:** Do not execute the `dumpsys` command on a running system. This will cause the system to crash and possibly cause a loss of data or corrupted file systems. Before executing this command, shut down the UNICOS operating system according to the methods described in the UNICOS administration documentation.

### 3.5.1 Dumping All Mainframes and I/O Nodes



**Caution:** Do not execute the `dumpsys` command on a running system. This will cause the system to crash and possibly cause a loss of data or corrupted file systems. Before executing this command, shut down the UNICOS operating system according to the methods described in the UNICOS administration documentation.

To dump all of the mainframes and IONs configured in the GigaRing topology file, enter the following command:

```
dumpsys
```

To indicate a reason for the dump, you can enter the `-r reason` option to the `dumpsys` command. The reason will be inserted into the dump. For example, to dump a mainframe named `production` because of a hung CPU, enter the following command:

```
dumpsys -r "CPU hung" production
```

### 3.5.2 Dumping Domains



**Caution:** Do not execute the `dumpsys` command on a running system. This will cause the system to crash and possibly cause a loss of data or corrupted file systems. Before executing this command, shut down the UNICOS operating system according to the methods described in the UNICOS administration documentation.

To dump the domain of specified mainframes, enter the following command on the SWS for that mainframe domain:

```
dumpsys -d mfnames
```

The *mfnames* variable specifies the mainframe whose domain should be dumped. The *mfnames* value is a comma-separated list of mainframe names. In the context of `dumpsys`, the domain consists of the specified mainframes themselves plus the GigaRing channels and IONs that are connected to only the

specified mainframes. If a ring is shared with an unspecified mainframe, that ring and its associated ION will not be dumped.

Domains are set up in the GigaRing topology file (see Section 2.1, page 11). For example, suppose your topology file is set up as follows:

- The `sn3001` domain includes `sn3001`, `ring-sn3001--0`, and `sn3001--mpn0`.
- The `sn3002` domain includes `sn3002`, `ring-sn3002--0`, and `sn3002--mpn0`.
- The `sn3001,sn3002` domain includes the super set of `sn3001` and `sn3002` domains.

You could then enter the following command to dump the `sn3001` domain, which includes `sn3001`, , and :

```
dumpsys -d sn3001
```

To dump the `sn3001,sn3002` domain (`sn3001`, `sn3002`, `ring-sn3001--0`, `ring sn3002--0`, `sn3001--mpn0`, and `sn3002--mpn0`), enter the following command:

```
dumpsys -d sn3001,sn3002
```

For more detailed information on domains, see the *SWS-ION Administration and Operations Guide* and the `bootsys(8)`, `dumpsys(8)`, and `scantopo(8)` man pages.

## 3.6 Shutting Down Mainframes

To shut down mainframes in the SuperCluster system and bring the UNICOS operating system back to single-user mode, you can use the `levelsys -r shutdown` command. See the `levelsys(8)` man page for detailed information on this command.

The `levelsys -r shutdown` command, which is the same as run-level 6, executes the `/etc/shutdown` script, which unmounts file systems, kills outstanding processes, and returns to single-user mode. However, the specific action taken with run-level 6 is configured in the mainframe operating system `inittab` file.

**Note:** The `levelsys` command requires an active console for the mainframe for which the run level is being changed.

### 3.6.1 Shutting Down the SuperCluster

The following steps shut down all mainframes in the SuperCluster system at once and bring the UNICOS operating system back to single-user mode:

1. Make sure that you are logged in to the mainframe as `root` and that you are in the `root (/), /etc, or /ce` directory.
2. You may want to send active users a message about when the system will be shut down. When the `/etc/shutdown` script returns the UNICOS operating system to single-user run state, it prompts you for a message that will be sent to all users. Before executing `/etc/shutdown`, you can use the `ps -eaf` command on the mainframe to see processes that are running, and the `who -u` command to see whether people are actively using the system.
3. Execute the following command on each SWS in the SuperCluster system to shut down all of the mainframes listed in the topology file:

```
levelsys -r shutdown
```

The time it takes for the shutdown process to complete depends on the number of processes that must terminate and file systems that must be unmounted. (Note that the completion of the `levelsys` command does not mean that the level change is complete.) When the shutdown process is complete, you are in single-user mode, but the UNICOS operating system is still running. You can perform any system administration work as necessary.

### 3.6.2 Shutting Down Individual Mainframes within the Cluster

To shut down individual mainframes and bring the UNICOS operating system back to single-user mode for those mainframes, enter the following command on the SWS for the specified mainframes:

```
levelsys -r shutdown mfnames
```

The *mfnames* variable specifies the name of the mainframes to be shut down. Separate multiple mainframe names with white space.

**Note:** The `levelsys` command completes before the operating system has changed run levels.

### 3.7 Halting Mainframes

Mainframes are automatically halted as part of the `bootsys` and `dumpsys` process, so you will not typically need to perform a manual halt. However, if you choose to do so, you should use the `haltsys(8)` command. See the `haltsys(8)` man page for more information.



**Caution:** The `haltsys` command does not manage issues related to system shutdown. If you execute this command on a running mainframe, there may be undesirable consequences, such as loss of data or corruption of file systems. Before executing this command, you should shut down the UNICOS operating system according to the methods described in the UNICOS administrator documentation.

# Mainframe Configuration [4]

---

This chapter discusses the following topics:

- File system layout
- File synchronization
- Configuring mainframe operating system parameter files
- Configuring the cluster UDB feature
- Setting up the TCP/IP network
- Controlling startup with `/etc/config/rcoptions` and `rc.*` files
- Controlling shutdown with `/etc/shutdown.*` files
- Configuring Cray SV1 series Cluster Bundle software products
  - DCE/DFS
  - NQE
  - NFS/BDS
  - MPI
  - DMF

**Note:** This chapter does not cover basic mainframe configuration in detail. It includes only information necessary for the configuration of Cray SV1 series SuperCluster systems. For information on basic mainframe configuration topics, refer to the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*.

## 4.1 File System Layout

All of the files that are accessible from within the UNICOS operating system are organized into file systems. The root file system contains the base or root of the file system tree. Other file systems are logically attached to (*mounted*) and detached from (*unmounted*) the root file system by the super user.

A file system is typically made up of slices of one or more disk devices. However, a file system can also reside totally or in part in memory.

### 4.1.1 File System Strategies

No one configuration of available disk drives into file systems will prove best for all purposes. In addition, as the needs of users change, the file system layout will most likely need to be reconfigured occasionally. In the absence of a set of absolute rules, the following guidelines should prove useful when you choose the file system layout for your system.

Table 1 shows the recommended file systems that should be local to each cluster node.

Table 1. Recommended File Systems Local to Each Cluster Node

File System	System Name	Notes
root	/	Two separate root partitions; one for backup. Required to be local by the UNICOS installation process.
usr	/usr	Two separate root partitions; one for backup. Required to be local by the UNICOS installation process.
swap	No specific mount point	
tmp	/tmp	
src	/usr/src	Required to be local by the UNICOS installation process.
dump	No specific mount point	Required to be local.
core	/dumps or /core	Required to be local.
dfs	/dfs_cache	
spool	/usr/spool	
adm	/usr/adm	

Table 2 shows the recommended file systems that should be shared within a cluster (for example, via NFS).



Table 2. Recommended File Systems Shared Within a Cluster

File System	System Name	Notes
opt	/opt	Some underlying products can be shared; /opt itself may not be a good candidate for a shared file system.
home	/home	User home directories
dfs_cache	/dfs_cache	

Most file systems listed in the preceding tables can exist on private MPN/SCSI disks. It is desirable to have local `swap`, `tmp`, and `dfs_cache` file systems located on faster disks, for example, on fiber disks or arrays via FCN(s).

#### 4.1.2 Sharing File Systems

Although it is recommended to share user file systems with NFS or another file-sharing mechanism, one limitation is that NFS-shared files may not contain checkpoint files or restartable core files. NFS does not support the restart mode bit that the UNICOS operating system requires to guarantee that the restart file has not been tampered with. Checkpoint and core files should be directed to local file systems. A core file written to an NFS file will not be restartable.

## 4.2 File Synchronization

Frequently, identical files exist on multiple mainframes within a group of mainframes. Administrators can use the `cfsync(8)` and `cfupdate(8)` commands to synchronize files among multiple mainframes. These utilities preserve file names, permissions, and privileges.

#### 4.2.1 Overview

After the file synchronization environment and necessary user accounts have been set up, the `cfsync` and `cfupdate` commands allow administrators to easily propagate files from one mainframe to another. An administrator first selects a mainframe to be the master host from which files are copied. The files on this host are considered to be the *master files*. After modifying the master files, the administrator executes `cfsync`, which archives the files and executes `cfupdate` on the remote mainframes. `cfupdate` copies the archive from the

master host, unpacks the archive, and restores the files while preserving the file permissions and privileges.

Files may be propagated relative to a file system other than /. For example, `/etc/hosts` on the master host could be propagated to `/new_root/etc/hosts` on the remote hosts. To do this, specify the `cfsync -R` option.

The following are examples of files that an administrator may want to be identical across mainframes:

```
/.rhosts
/etc/auto/*
/etc/craylm/*
/etc/daemons
/etc/hosts
/etc/hosts.equiv
/etc/inetd.conf
/etc/issues
/etc/motd
/etc/mrouted.conf
/etc/named.boot
/etc/networks
/etc/printcap
/etc/project
/etc/protocols
/etc/rc.pre
/etc/rc.mid
/etc/rc.pst
/etc/resolv.conf
/etc/services
/etc/shells
/etc/shutdown.pre
```

```
/etc/shutdown.mid  
/etc/shutdown.pst  
/etc/smttd.conf  
/etc/snmpd.conf  
/etc/uidmaps/Get.domains  
/etc/uidmaps/Map.domains  
/etc/uidmaps/Set.domains  
/etc/uidmaps/Update.domains  
/etc/utcd.conf  
/usr/lib/array/arrayd.auth  
/usr/lib/array/arrayd.conf
```

Network configuration files (which may be arbitrarily named)

Other site-specific scripts

#### 4.2.2 Setup Procedure

The following steps describes the setup for file synchronization:

1. Create an identical user account on each mainframe. The user name is specified in the configuration file (`/etc/config/cf/cf_config`) by the `REMOTE_USER` variable.
  - a. On the master host, set `secadm` as a valid category. You may optionally set `secadm` as an initial category.
  - b. On the remote hosts, set `secadm` as both a valid and initial category. For added security, you may wish to disable interactive logins.

The following example disables interactive logins for user `update_user`:

```
# udbgen -c update:update_user:passwd:x:
```

- c. On both the master and remote hosts, add `REMOTE_USER` to the `adm` group.
2. Set up the `.rhosts` file on each of the hosts.

- a. On the master host, an entry for REMOTE\_USER must exist on each of the remote hosts. The REMOTE\_USER variable is defined in the configuration file, /etc/config/cf/cf\_config. The remote hosts are defined in /etc/config/cf/machine\_list.

In the following example, REMOTE\_USER is set to update\_user, and machine\_list contains host1, host2, and host3:

```
host1 update_user
host2 update_user
host3 update_user
```

- b. On the remote hosts, add an entry for either REMOTE\_USER or root on the master host. The REMOTE\_USER entry is needed when cfsync is executed by REMOTE\_USER. Otherwise, when root executes cfsync, the root entry is required.

In the following example, an .rhosts entry exists for both root and REMOTE\_USER. REMOTE\_USER is set to update\_user, and the master host is named master\_host:

```
master_host update_user
master_host root
```

3. Set the umask correctly if root executes cfsync. When cfupdate is executed from cfsync, the REMOTE\_USER must be able to access the archive file on the master host. If root runs cfsync instead of REMOTE\_USER, file access may be denied, because root's umask is set incorrectly. The umask(1) command can be used to set the file creation mask.
4. Edit the configuration file. Modify /etc/config/cf\_config as needed on each host.
5. Check that cfsync is not on a cross-mounted file system. cfsync must be installed with privilege. When cfsync is located on a cross-mounted file system, the privileges will not be available on all hosts. cfsync is located in /etc.

#### 4.2.3 Execution Procedure

The following steps describe how to propagate files to multiple hosts and assume that the default files are used. The steps are executed on the master host.

1. Edit the files to be propagated.

2. Modify `/etc/config/cf/file_list`. Edit the `file_list` file so it contains only the names of the files that are to be propagated.
3. Modify `/etc/config/cf/machine_list`. Edit the `machine_list` file so it contains the names of the mainframes to which the files are propagated. This list must contain full absolute path names. All file names must begin with `/`.
4. As either `root` (on a `PRIV_SU` system) or a user with active `secadm` privilege, execute `cfsync`. If you use a non-root login, you may first need to activate the `secadm` category with `setucat(1)`. This must be done when the initial categories do not include `secadm` (see Section 4.2.2).

### 4.3 Configuring Mainframe Operating System Parameter Files

**Note:** It is strongly recommended that you use the install tool to maintain your system configuration, rather than manually editing the configuration specification language (CSL) parameter file. For more information on the UNICOS installation and configuration menu system (ICMS), see *UNICOS System Configuration Using ICMS* and the online install tool help files.

The configuration specification language (CSL) parameter file contains statements that specify hardware and software characteristics for your system.

When the configuration is activated, a copy of the CSL parameter file is stored in `/etc/config/param`. For all Cray systems, the administrator must manually transfer the parameter file to the workstation.

Each Cray SV1 series system in the SuperCluster system has unique `param` files to handle the private file systems that the Cray SV1 series systems will use. A copy of the default `param` file is located in Appendix A.

#### 4.3.1 Defining the GigaRing Host-to-Host Connection

The network section of the `/etc/config/param` file is used to define the GigaRing interfaces so that TCP/IP can be used for mainframe-to-mainframe communication.

The following lines should be added to the `network` section of the parameter file for each GigaRing channel over which TCP/IP will be used to communicate with another mainframe.

```
gr number {  
    iopath {ring ring_number; node node_number;}  
}
```

In the `gr` device format, *number* is the interface number (for example, 0 would be the first, 1 would indicate a second interface, and so on). *ring\_number* identifies the GigaRing channel that TCP/IP will use to communicate between mainframes. *node\_number* identifies this mainframe's assigned node ID on this ring.

The following example shows the lines used to define sn3001's interface on the intra-node GigaRing topology example (see Figure 5, page 12):

```
gr 0 {  
    iopath {ring 040; node 1;}  
}
```

See “network section” in the “CSL Parameter File” chapter of the *UNICOS Configuration Administrator's Guide* for detailed information on the `network` section of the CSL parameter file. See Appendix A for an example of the parameter file.

#### 4.3.2 Determining and Setting the Number of mbufs

You need to determine and set segments of a special memory pool (mbufs). See the “TCP/IP” chapter in the *UNICOS Networking Facilities Administrator's Guide* for detailed information on determining and setting mbufs. See Appendix A for an example of the parameter file.

## 4.4 Configuring the Cluster UDB Feature

Each node in a SuperCluster system has a local user database (UDB) as if it were a standalone system. Cray has developed tools to synchronize these UDB files throughout the cluster so that the user information contained within the database is consistent among all nodes in the cluster. A cluster user will have only one identity and collection of limits, quotas, and permission options. A single source file makes administering a cluster UDB similar to administering a standalone system, provided there is not a great deal of node-specific tailoring of user information. Cluster UDB administration may be done via the command line from any node in the cluster.

#### 4.4.1 Installation Process

Installation of the cluster UDB places all of the cluster UDB commands in `/etc` when the installation process runs on the node. Along with installation of the commands, symbolic links to `udbgen(7X)`, `udbsee(1)`, and `udb_helper(8)` are created in `/usr/adm/udb/bin`. Installation will add, but not replace, names in this directory.

The cluster UDB configuration file contains locally specified defaults and preferences and the cluster UDB source and update distribution files. A copy of this configuration file, called `clusterUDB.conf`, is installed in `/etc/config/udb` on each node unless a file of that name is present. If the file already exists, the newly installed configuration file will be named `clusterUDB.conf.new`.

#### 4.4.2 Configuration Procedure

After the cluster UDB feature is installed a configuration to match the local environment must be done. The following steps outline what the administrator must do to ensure the cluster UDB works correctly. This procedure assumes that `/etc` and `/usr/adm/udb` are not on file systems that are shared among the nodes of the cluster. A correct configuration file will make using any of the special path and file name options on the cluster UDB commands unnecessary for normal operations.

1. **Decide configuration file location:** Decide where the configuration file, `clusterUDB.config`, will reside. After installation a copy of this file exists on every node, but it will be difficult to maintain a consistent configuration in this way. You are encouraged to copy the configuration file to a file system that is shared among the nodes and, on each node, replace the configuration file in `/etc/config/udb` with a symbolic link to the shared file.
2. **Configure path names:** A number of path names must be set up in the configuration file. The released file uses the path `/usr/udbAdmin` as a place holder that must be changed to a path all nodes can access. The various directories (`Collect`, `Dist`, `Edit`, and `Log`) need to be created in that path.
3. **Specify the list of nodes:** The configuration directive `NodeList` must be changed to the actual list of nodes comprising the cluster. The names will be used to address the nodes when collection and distribution are done.

4. `udb_server(8)` restrictions: Only commands that exist in `/usr/adm/udb/bin` can be run from `udb_server`. The installation process should automatically populate this directory with all needed commands. Because the installation does not touch existing entries in the directory, you may need to check that the content of the directory is correct on each node.
5. Decide how the cluster UDB is to be defined: This involves specifying exempt records, uniform fields, and dropped fields. These definitions control what is placed in the cluster UDB source file and how `udb_merge(8)` checks for cluster uniformity.
  - Exempt records are not checked for uniformity, but all records with the same user name and UID are collapsed to a minimum representation of field value differences over the cluster. The released configuration file makes all records with a UID less than 100 exempt. You must determine what is locally correct.
  - Uniform fields are expected to have identical values per record as specified by user name and UID on each node of the cluster. Exempt records are not checked. Fields not specified as uniform may have different values on each node without triggering warning messages during the merge process. The different values will be maintained in the cluster UDB source file unless they are to be dropped.
  - Dropped fields will not appear in the cluster UDB source file. Typically, fields which are dynamic and reflect activity on a particular node should be dropped.
6. The final configuration task is to define a new user template. This template is used by the `udb_edit -n` option to create a default user record. See the comments in the default UDB configuration file shown in Appendix A for detailed information.
7. Perform network configuration. Configure the `/etc/services` file for service name `udb_server`:

```
udb_server pppp/tcp # Cluster UDB server
```

The port number is placed where *pppp* appears.

8. Configure the `/etc/inetd.conf` file for the cluster UDB server:

```
udb_server stream tcp nowait root /etc/udb_server udb_server
```



#### 4.4.3 Initialization Procedure

After UDB installation and configuration, the cluster UDB needs to be initialized. To initialize the cluster UDB:

1. Gather all UDB entries for all nodes on one of the individual nodes using the `udb_collect` command.
2. Merge all cluster UDB information into a single cluster UDB source file using the `udb_merge` command.

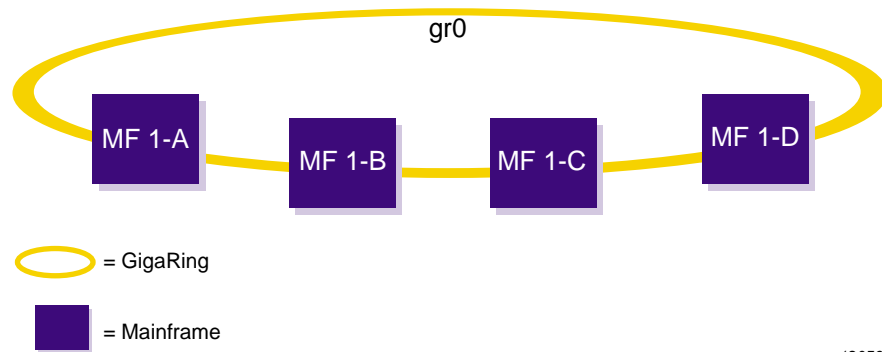
#### 4.5 Setting Up the TCP/IP Network

This section discusses special networking considerations for the Cray SV1 series SuperCluster.

All systems in the SuperCluster system are connected via a host-to-host GigaRing interface which provides the physical connection for TCP/IP communication. The interface type, as specified in the `network` section of the `/etc/config/param` file, is `gr`. Each system will have either one or two of these types of interfaces defined (that is, `gr0` and optionally `gr1`), depending on the size of the SuperCluster system. Each GigaRing connection defines a separate IP network.

As far as TCP/IP is concerned, a SuperCluster system exists as a set of independent systems connected via shared GigaRing networks. These GigaRing networks can be viewed as a matrix, with each system having direct access to any of its neighbors which shares either the inter-node or intra-node GigaRing network, and indirect access with any other system in the SuperCluster system. Indirect access is provided by routing data through a system that is directly connected to both systems in the communication path.

Figure 7 depicts the smallest SuperCluster system (that is, four systems which are all directly connected via the same GigaRing interface):



a12056

Figure 7. Smallest SuperCluster GigaRing Interfaces

As the SuperCluster system grows, additional GigaRing interfaces need to be defined such that the SuperCluster system can be depicted as a matrix. Figure 8, page 41 depicts an arbitrarily large SuperCluster system:

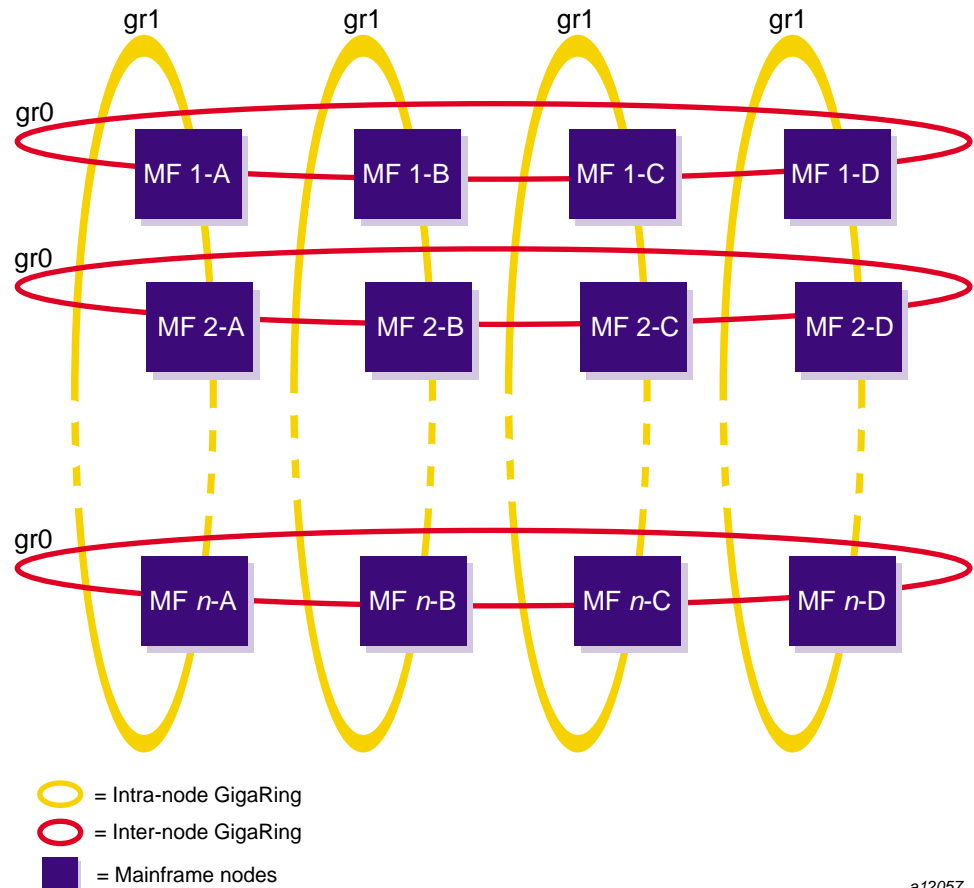


Figure 8. Large SuperCluster GigaRing Interfaces

**Note:** The practical limit on the number of systems configured per GigaRing network is eight.

No requirement exists to define the X axis rings as `gr0` and the Y axis rings as `gr1`, but care should be taken when naming each system's GigaRing interfaces as the configuration can become difficult to maintain. Because a SuperCluster building block consists of four systems, it is easiest to define its interface as `gr0` and the GigaRing network, which connects these SuperCluster building blocks, as each system's `gr1` interface.

The following sections detail the configuration of the network that supports communication between all systems in a Cray SV1 series SuperCluster system.

#### 4.5.1 IP Addressing Considerations

To provide TCP/IP access to all Cray SV1 series systems in the cluster, each Cray SV1 series system must have an IP host address assigned to each of its GigaRing interfaces. Also, each GigaRing channel must exist as its own IP network. As the SuperCluster system grows, the number of IP host and network addresses required also grows. The number of IP host and network addresses required in a SuperCluster building block (the smallest SuperCluster system) is four and one, respectively. In an eight-system SuperCluster system, sixteen IP host addresses and six network addresses are required. As the SuperCluster system grows from this point, eight additional IP host addresses and one additional network address are required for each building block added to the SuperCluster system.

To conserve your site's address space, the software used on the Cray SV1 series systems in the SuperCluster system supports variable subnet masks. This allows for the use of only one class C equivalent network address to define the entire IP network that connects all of the systems in the largest possible SuperCluster system. Because eight systems at most can be directly connected over an individual GigaRing network, only the last four bits of any IP address need to be used to define the individual hosts on that network. As a result, the least restrictive subnet mask that needs to be used to define these GigaRing networks is 0xFFFFFFF0 (or 255.255.255.240).

**Note:** If your SuperCluster system consists of at most sixteen systems (that is, a four-by-four SuperCluster system), you can use the more restrictive subnet mask of 0xFFFFFFF8 (or 255.255.255.248), as only the last three bits of each address are required to define four hosts. As indicated earlier, this configuration can become difficult to maintain, so if there is a possibility of your SuperCluster system growing beyond sixteen systems, it is recommended to immediately start using the least restrictive subnet mask.

Assuming the assignment of a class C equivalent address is used to take care of the host/network addressing requirements, the following defines the available network addresses:

```
class C address . 0
class C address . 16
class C address . 32
class C address . 48
class C address . 64
```

```
class C address.80  
class C address.96  
class C address.112  
class C address.128  
class C address.144  
class C address.160  
class C address.176  
class C address.192  
class C address.208  
class C address.224  
class C address.240
```

*class C address* is the first three octets of the Internet address assigned to the SuperCluster system.

Each host address then becomes the network address, modified such that the last octet is summed with a number from 1 to 8, indicating which host the system is defined as on the given network.

As the configuration can become complex, it is suggested to use the network addresses for the `gr0` interfaces starting from 0 and going up, and the `gr1` interface from 240 on down. So, using the arbitrarily large SuperCluster system shown in Figure 8, page 41, the Internet addressing could be set up as shown in Table 3, page 44, for the indicated systems (leaving out the first three octets of the Internet addresses, because they will all be the same):

Table 3. Internet Addressing

System	Interface	Host Address	Network
1-A	gr0	.1	.0
	gr1	.241	.240
1-B	gr0	.2	.0
	gr1	.225	.224
1-C	gr0	.3	.0
	gr1	.209	.208
1-D	gr0	.4	.0
	gr1	.193	.192
2-A	gr0	.17	.16
	gr1	.242	.240
2-B	gr0	.18	.16
	gr1	.226	.224
2-C	gr0	.19	.16
	gr1	.210	.208
2-D	gr0	.20	.16
	gr1	.194	.192
...			
$n$ -A	gr0	$.(16*(n-1))+1$	$.(16*(n-1))$
	gr1	$.(240+n)$	240
$n$ -B	gr0	$.(16*(n-1))+2$	$(16*(n-1))$
	gr1	$.(224+n)$	.224
$n$ -C	gr0	$.(16*(n-1))+3$	$(16*(n-1))$
	gr1	$.(208+n)$	.208
$n$ -D	gr0	$.(16*(n-1))+4$	$(16*(n-1))$
	gr1	$.(192+n)$	.192

**Note:** No requirement exists to define the IP addresses as shown in Table 3, page 44. This information is provided simply to assist in the configuration of the GigaRing interfaces. The only requirements are that each GigaRing interface be assigned an IP address, and that each GigaRing exists as its own IP network/subnet.

#### 4.5.2 IP Address-to-Hardware Address Mapping

An IP address is a logical address that allows the networking software to operate independently of hardware addressing considerations. As a result, in order for TCP/IP to operate correctly, all systems must be configured such that these IP addresses can be mapped to their hardware equivalent address. This mapping is done statically because the GigaRing hardware does not support broadcast messages, nor does the software support address resolution protocol (ARP) over this interface.

A GigaRing hardware address consists of a pair of numbers: one that identifies the ring number of the GigaRing, and another that identifies the node number of the host. This information can be obtained by looking at the topology file on the SWS. The mapping is performed statically by using the `arp(8)` command on each system and specifying the name of a file that associates each host's IP address, on the given GigaRing, to its equivalent GigaRing address.

Using the default system configuration tools, the files `/etc/gr0.arp` and optionally `/etc/gr1.arp` need to be created such that they contain this mapping for their respective interfaces. These file names are associated with their interfaces in the `/etc/config/interfaces` file. The format of this file is as follows:

```
IP address      0:0:0:0:ring:node
```

where:

*IP address* is the host's IP address or host name, which maps to the host's IP address in either the `/etc/hosts` file or by `named(8)`.

`0:0:0:0:` is a place holder and must be specified as shown.

*ring* is the GigaRing number assigned to this interface.

*node* is the host's node number on *ring*, which has been assigned *IP address*.



**Caution:** Both the *ring* and *node* numbers **must** be specified in hexadecimal (without the leading 0x).

Each host on any given GigaRing network must have the same mapping for all hosts on that network. In other words, all of system 1-x's `/etc/gr0.arp` files must be the same. Also, all of system n-A's `/etc/gr1.arp` files must be the same, all of system n-B's `/etc/gr1.arp` files must be the same, and so on.

### 4.5.3 Routing Within the Cluster

Once the IP-to-hardware address mapping is set up, any two directly connected hosts can communicate using TCP/IP. To allow for any two indirectly connected systems to communicate, the appropriate routes must be created. You can do this by using the `route(8)` command directly, or by specifying these routes in the `static` section of the `/etc/gated.conf` file, which is processed by `staticrts(8)` during system initialization to create the necessary routes.

Going back to the matrix configuration shown in Figure 8, page 41, each system will need a route for each `gr1` network defined on the other hosts connected to its `gr0` interface, and a route to each of the other `gr0` networks. Although this setup is not a requirement, the following is recommended for simplicity:

- To define the routes to the other `gr1` networks to go through the appropriate host connected via the system's `gr0` interface.
- To define the routes to the other `gr0` networks to go through the appropriate host connected via the system's `gr1` interface.

Assuming the network addressing is assigned as recommended above, the matrix would appear as shown in Figure 9, page 47 (substituting the network address for the interface name):



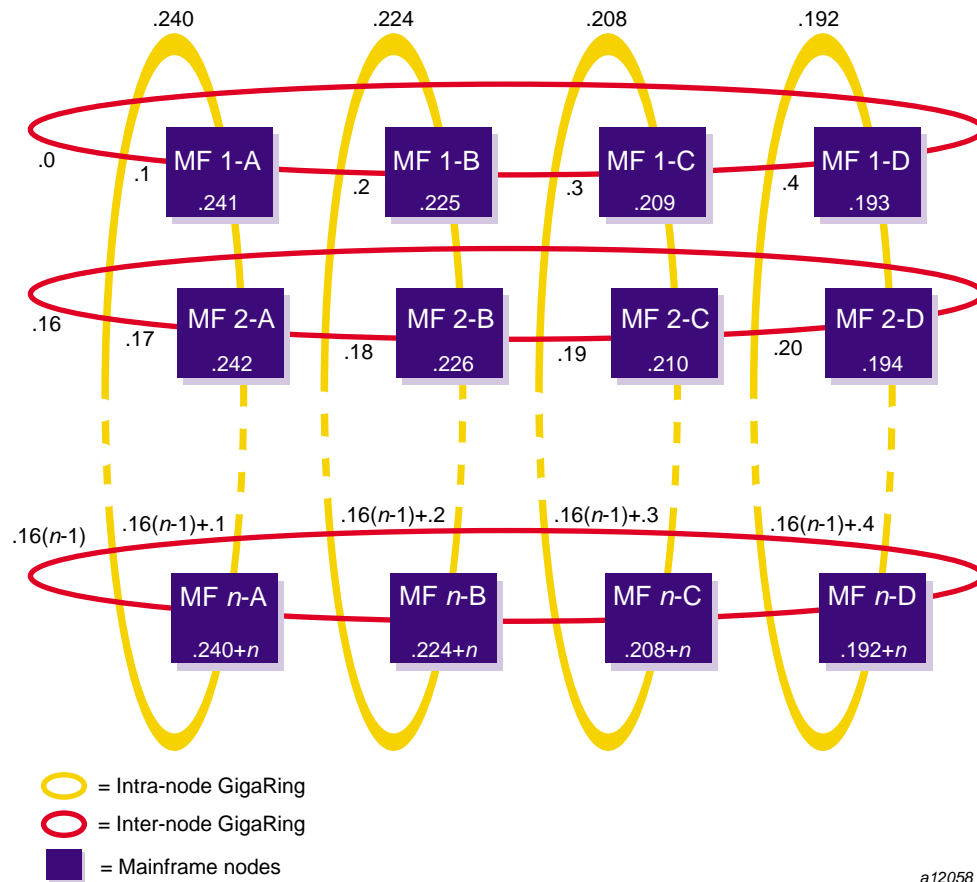


Figure 9. Routing Within the Cluster

The required routes for any system x-A would then be as follows, using route to create them:

```
/etc/route add -net network.0 network.241
/etc/route add -net network.16 network.242
/etc/route add -net network.32 network.243
...
/etc/route add -net network.(16*(n-1)) network.(240+n)

/etc/route add -net network.224 network.((16*(x-1))+2)
/etc/route add -net network.208 network.((16*(x-1))+3)
```

```
/etc/route add -net network.192 network.((16*(x-1))+4)
```

The required routes for any system *x*-B would be as follows, using `route` to create them:

```
/etc/route add -net network.0 network.225
/etc/route add -net network.16 network.226
/etc/route add -net network.32 network.227
...
/etc/route add -net network.(16*(n-1)) network.(224+n)

/etc/route add -net network.240 network.((16*(x-1))+2)
/etc/route add -net network.208 network.((16*(x-1))+3)
/etc/route add -net network.192 network.((16*(x-1))+4)
```

and so on...

where *n* goes from 1 to the number of building blocks in the SuperCluster and **is not** equal to *x* (that is, the route to any system's own `gr0` network does not have to be duplicated as this route is added when the interface is configured up).

**Note:** *network* is the class C equivalent network, or subnet, address assigned to the SuperCluster's GigaRing networks.

## 4.6 Controlling Startup With `/etc/config/rcoptions` and `rc.*` Files

During the startup of each mainframe in the SuperCluster system, various actions can be done automatically, not at all, or only after verification by operator input. The following files are used to control the startup behavior of each mainframe in the SuperCluster system:

- `/etc/config/rcoptions`
- `/etc/config/rc.pre`
- `/etc/config/rc.mid`
- `/etc/config/rc.pst`

The `/etc/config/rcoptions` file is controlled by the UNICOS Installation/Configuration Menu System (ICMS) from various component parts, which are in turn either generated by ICMS, or maintained manually by the

system administrator. See the *General UNICOS System Administration* manual for a detailed explanation of the `/etc/config/rcoptions` file.

The following paragraphs show how to set RC option variables for booting. Each RC option variable has the following form:

```
RC_XXXX= 'runlevel=action[ :runlevel=action] '
```

where:

*runlevel* is an `/etc/init(1M)` level (0123456SsQqabc) or an asterisk (\*), which specifies all unspecified levels.

*action* is YES for unconditionally execute the function, NO for unconditionally skip the execution of the function, or ASK, which specifies to ask the operator whether or not to execute the function.

**Note:** YES and '\*=YES' are synonymous, and blank fields default to '\*=YES'.

The following examples show how the RC option variables can be used to customize the features available by default when booting:

#### Example

```
RC_XXXX= 'YES'
```

```
RC_XXXX= '2=YES:3=NO:*=ASK'
```

#### Definition

Always execute this function.

Always execute this function when going to level 2; never execute this function when going to level 3; and ask for all other levels.

The settings in the `rcoptions` file affect whether or not systems can go from single-user mode to multiuser mode without operator input. In particular, the `RC_CONTErr` parameter controls whether or not systems continue to multiuser mode if an error is encountered. By default, this parameter is set to ASK, which stops the multiuser mode process until the operator responds to the error displayed on the system console.

It is recommended that a site's `rcoptions` file not contain an ASK when each system in the SuperCluster system is being brought up.

Sites should review the basic `rcoptions` file shown above and the parameter definitions, which are documented in section 3.4 of *General UNICOS System Administration*.

The `/etc/config/rc.pre`, `/etc/config/rc.mid`, and `/etc/config/rc.pst` files are entry points into `/etc/rc` and allow sites to customize the actions taken during the booting of a system into multiuser mode. Section 3.4 of *General UNICOS System Administration* describes where these files are executed during the process of getting a system to multiuser mode. The following are common uses for each of these scripts:

- `/etc/rc.pre` script to control what type of NQS startup should occur; for example, whether separate queues should exist for different times of day
- `/etc/rc.mid` script to update the `/etc/motd` file, control whether the system should be restricted, make sure the mail `syslog` file is satisfactory, and make sure the `/etc/hosts` file is as new as possible
- `/etc/rc.pst` script to start the NQS queues, restore the set memory scheduling parameters, `mkfs`, `fsck`, and mount scratch file systems, and enable disk queue sorting

## 4.7 Controlling Shutdown With `/etc/shutdown.*` Files

During the shutdown of each mainframe in the SuperCluster system, additional actions can be done automatically by adding the following site shutdown files:

- `/etc/shutdown.pre`
- `/etc/shutdown.mid`
- `/etc/shutdown.pst`

Site shutdown files are executed at various points during system shutdown. For a detailed description of when the files are executed, see section 3.2 of *General UNICOS System Administration*. The following are common uses for each of these scripts:

- `/etc/shutdown.pre` script to provide a unique shutdown procedure for site-specific levels of operation
- `/etc/shutdown.mid` script to allow NFS file systems to be unmounted prior to stopping networks
- `/etc/shutdown.pst` script to synchronize the UDB and fair-share information after the networks have been stopped

#### 4.7.1 /etc/shutdown.pre Script

You should create an `/etc/shutdown.pre` script that contains a `wall(8)` command indicating that the system is to be shut down in a specified number of seconds (`-d secs`). The default delay is 60 seconds. If you want more of a delay and additional `wall` commands to be issued, these can be added to the `/etc/shutdown.pre` script.

A basic form of `/etc/shutdown.pre` is shown in the following example:

```
/etc/wall<<!  
^GSystem going down in 60 seconds. Please logoff now.^G  
!
```

See the `shutdown(8)` man page for more information.

## 4.8 Configuring Cray SV1 Series Cluster Bundle Software Products

This section describes any special administrative issues for Cray SV1 series Cluster Bundle software products. Where there are no special issues, you are referred to the basic product documentation.

### 4.8.1 NQE

The Network Queuing Environment (NQE) for the UNICOS operating system is a workload management system that automatically balances tasks across mixed servers attached to a network. See *NQE Installation* and *NQE Administration* for basic information on configuring NQE.

### 4.8.2 NFS/BDS

The UNICOS network file system (NFS) allows users to share directories and files across a network of machines. See the *UNICOS Networking Facilities Administrator's Guide* for basic information on configuring NFS.

`bds` is a user-level server, implemented as an enhancement to the network file system (NFS), that provides direct I/O capabilities over the network. `bds` is typically started out of startup scripts at boot time. See the `bds(8)` and `mount(8)` man pages for basic information on BDS.

### 4.8.3 MPI

The Message Passing Interface (MPI) is a component of the Cray Message Passing Toolkit (MPT), which is a software package that supports parallel programming across a network of heterogeneous computer systems through a technique known as message passing.

The following MPI limitation exists when running across a cluster.

There is currently a limit of 55 processes for MPI jobs using the shared memory implementation (`-nt` option on `mpirun` command). This limit is based on the number of open socket descriptors and may be a configuration issue. This limit will be slightly less for MPI jobs using the TCP/IP implementation (`-np` on `mpirun`). See the `mpirun(1)` man page and the *Message Passing Toolkit: MPI Programmer's Manual* for information on MPI.

### 4.8.4 DMF

The optional UNICOS Cray Data Migration Facility (DMF) provides online file system space by moving selected files offline to a designated storage devices. These files remain cataloged in their original directories and behave as if they are still disk resident. Likewise, an online disk can be considered a cached copy of a larger virtual disk space. See *Cray Data Migration Facility (DMF) Release and Installation Guide* for basic information on configuring DMF.

# Mainframe Operations [5]

---

This chapter discusses the following topics:

- Maintaining the user database
- Batch scheduling and load balancing
- Centralized accounting
- Backing up and restoring file systems

**Note:** This chapter does not cover basic mainframe operations in detail. It includes only information necessary for the operations of Cray SV1 series SuperCluster systems. For information on basic mainframe operations topics, refer to the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*.

## 5.1 Maintaining the User Database

Each node in a SuperCluster system has a local user database (UDB) as if it were a stand-alone system. Cray has developed tools to synchronize these UDB files throughout the cluster so that the user information contained within the database is consistent among all nodes in the cluster. A cluster user will have only one identity and collection of limits, quotas, and permission options. A single source file makes administering a cluster UDB similar to administering a stand-alone system, provided there is not a great deal of node-specific tailoring of user information. Cluster UDB administration may be done via the command line from any node in the cluster.

The supercluster UDB commands issue the common UDB commands with the supercluster options.

A cluster UDB configuration file contains locally specified defaults and preferences and the cluster UDB source and update distribution files. By default, the file is located in `/etc/config/udb` and is called `clusterUDB.conf`. See Appendix A for the default cluster UDB configuration file.

### 5.1.1 Node Level UDB Tools

Each node in the cluster contains the set of UDB files common to all UNICOS systems. In addition, each node has access to the cluster UDB configuration file, the distribution files, and the cluster UDB source file.

The `udbgen` command is used to create or maintain information in each node's UDB. This command must be run on the node that contains the target UDB. See the `udbgen(8)` man page for detailed information on this command.

To support many nodes with common UDB source and allow each node to have tailored user fields, target node names must be included with the source. The `udbgen field-name:field-value: pair` called `node:node-name:` provides for the specification of the node name.

The block introduction field name, `change`, allows records to be created or updated from the source. This is necessary because the UDB may not have the same record population on all nodes when a cluster UDB source file is distributed.

The `udbsee` command converts information from the UDB into an ASCII file. This command must be run on the node that contains the target UDB. Its output has been modified to include the `node:node-name:` pair in every output record if the `-n (node)` option is specified. If the source is intended for input to the UDB merge tool, specific options (`-a`, `-g`, `-n`, `-v`, and `-t change`) are required.

For detailed information, see the `udb(5)`, `udbgen(8)`, and `udbsee(1)` man pages.

### 5.1.2 Cluster Level UDB Tools

Information in a UDB record is organized into three categories for cluster UDB administration:

- Non-configurable node invariant user identity information. The fields in this category are `name` and `uid`. Except for any configured exceptions, all records must comply. Exception records are named in the configuration file using the `Exempt` keyword.
- Configurable node invariant fields. These fields are named in the configuration file with the `Uniform` keyword. The default configuration file specifies most of the UDB fields to be in this category.
- Node-specific information having no cluster wide meaning. Such fields as `shextime` and `shusage` belong to this category. The merge process skips



fields in this category. These fields are named in the configuration file with the `Drop` keyword.

When `udbsee` produces UDB source on each node, each record includes the `node-name`. Many fields are identical on each node because of the rules requiring a consistent user identity. When the source is combined with the merge tool, all records have cluster uniform and node-specific field-value pairs.

When working with the cluster level UDB tools the administrator, using the `udb_edit(8)` tool and any desired ASCII editor, edits an extract from the merged UDB source file to change user attributes. The extract is used as input to the `udb_update(8)` tool to update the individual node UDB files.

#### 5.1.2.1 UDB Source Editing Tool

The UDB source editing tool, `udb_edit`, is used to create a template cluster UDB record for the creation of a new user record or to extract an existing cluster UDB record for editing.

When you run `udb_edit`, an editor is started to allow you to manipulate the record being created, changed, or deleted. The `EDITOR` environment variable selects the editor.

To create a record for a new user:

Run `udb_edit` with the `-n` option. For example, to create a record for a new user named `newuser`, use the following command:

```
udb_edit -n newuser
```

New user records are created using a template extracted from the cluster UDB configuration file (default name: `clusterUDB.conf`). The template default field values for new records in the cluster UDB configuration file may be set by the administrator.

A file is created in the directory named in the `EditDirectory` configuration directive in the cluster UDB configuration file. The name of the file is `username.yyyymmddhhmmss`. The date suffix makes the file name unique and orders the files according to time of creation.

To edit an existing record:

Run `udb_edit` and specify the user name to select from the merge file (the cluster UDB source file). For example, the following command retrieves user `rn1`'s record for editing:

```
udb_edit rn1
```

User `rn1`'s record is opened in the editor specified by the `EDITOR` environment variable. To change the UDB record, specify new values for fields; then exit and save the file.

To delete a user's UDB record, run `udb_edit` to get the user's record for editing. Change the block introduction field name from `update` to `delete` and proceed as when updating an existing user record.

To create or regenerate the index file, use the command:

```
udb_edit -X
```

For detailed information on command options, see the `udb_edit(8)` man page.

#### 5.1.2.2 UDB Source Collection Tool

The UDB source collection tool, `udb_collect(8)` forwards UDB source to the collecting node in a cluster upon request. The command can be executed from any directory, but it will deposit the collected files in the directory named by the `CollectionDirectory` configuration directive in the cluster UDB configuration file. Options allow changing the suffix to something other than `.udb` and running in verbose mode so that progress messages are written.

The source file is collected on each node using the `udb_server(8)` command as follows:

```
udb_server udbsee -agmv -t change -o %Cnodename.suffix
```

This command will not replace any existing files of the same name. This set of options produces the only format acceptable to `udb_merge(8)`, the UDB source merging tool. The `udbsee` output files (UDB source) from each node made available by the `udb_collect` command are read by the `udb_merge` tool and converted into a single cluster UDB source file.

For detailed information on command options, see the `udb_collect(8)` man page.

### 5.1.2.3 UDB Source Merging Tool

The UDB source merging tool, `udb_merge(8)`, reads the `udbsee` output files (UDB source) from each node in the cluster and converts them into a single cluster UDB source file while applying the rules in the cluster UDB configuration file. (The `udb_collect(8)` command collects the node UDB source files.)

In each record in the cluster UDB source file, a field having identical values in each node's UDB will be merged into a single `field-name:field-value:expression`. Non-uniform fields will be identified with the appropriate node names. Fields in violation of the uniformity rules will be marked in the cluster UDB source file for correction. Node-specific fields will be dropped from the merged data. Dropped fields are intended to prevent the merged source file from containing any node-specific information that should not be restored to the node's UDB when UDB synchronization is done. Information such as share exit time and share usage fall into this category.

Other information may also be dropped if the administrator so configures the tool. The cluster UDB configuration file (default name: `clusterUDB.conf`) contains merge configuration statements that define exempt records, uniform fields, and fields to drop. A default configuration file is provided that specifies recommended node invariant and node-specific fields. See Appendix A for the default cluster UDB configuration file.

For detailed information on command options, see the `udb_merge(8)` man page.

### 5.1.2.4 UDB Cluster Updating Tool

The UDB cluster updating tool, `udb_update(8)`, distributes UDB changes to all nodes in the cluster.

This command provides four separate operations:

- Distributes the entire cluster UDB source file to all nodes.
- Distributes the collection of user records that are edited or created with `udb_edit` to all nodes.
- Propagates the update of a specific field on one node to all nodes in the cluster.
- Joins a node to the cluster and applies pending changes to its UDB.

The distribution file created by the `udb_update -u` and `-e` command options will be in the `DistributionDirectory` with the name prefix specified by the

DistributionFile configuration directive. The default prefix is `Dist`. The file name also encodes the creation time. The initial file name has the form `Distyyyymmddhhmmss`.

If you run `udb_update` without the `-e`, `-j`, `-s`, or `-u` options, `udb_update` uses all the files in the `EditDirectory` as though each had been specified by the `-u` option. After running in this mode (without the `-e`, `-j`, `-s`, or `-u` options) `udb_update` automatically deletes all the update records in the `EditDirectory` after creating the cluster file. Otherwise, update records are retained in the `EditDirectory` until you manually delete them.

The list of nodes to contact is provided through the `NodeList` configuration directive. The `udb_update` command creates hard links to the distribution file for each node using the name `Distyyyymmddhhmmss`.

If the distribution file is created by the `udb_update -u` and `-e` command options, the file to which the hard links point will be removed before node distribution is started. When all of the links are deleted, the distribution file will disappear, so that no administrative cleanup is necessary. If the `-s` option is used, the links will point to the source file, but that file is not removed and so remains after node distribution is finished.

As each node completes the update, the link for that node will be removed. Any links remaining after the update completes means that those nodes did not perform the update.

The `udb_server` command is called to execute the `udbgen` command on each of the nodes. A typical server request would be `udbgen %Dfilename`.

If `udb_update` is called with the `-e` option, `udbgen` is run with the `-E` option to adjust associated fields correctly on all nodes. At this time, the only associated field action is to update the password change time if password aging is active and the password is changed.

When a distribution file is created, a copy is appended to the change log to record the action. If the cluster source file is distributed (`-s` option), the change log contains the action but not the content of the source file. The change log is in `LogDirectory` and is named by the `ChangeLog` configuration directive.

For detailed information on command options, see the `udb_update(8)` man page.

#### 5.1.2.5 UDB Server

The `udb_server(8)` command executes the cluster UDB requests on individual nodes. It reads the request information (through `inetd`) from the remote `udb_collect` or `udb_update` command, formats it, and calls the `udb_helper` shell script.

The `udb_server` command is started by `inetd` for a `udb_collect` or `udb_update` command running on another node in the cluster. `udb_helper` performs any local site actions and calls `udbsee` or `udbgen` as directed by the remote `udb_collect` or `udb_update` command.

**Note:** The `udb_server` command is to be used only by `inetd`. Do not use it from the command line.

For detailed information on command options, see the `udb_server(8)` man page.

#### 5.1.2.6 UDB Helper

The `udb_helper` shell script can be used to customize UDB source file collection and update actions on each node in the cluster. `udb_helper` is called by `udb_server`.

**Note:** This script is not intended to customize the UDB, but provides a way to initiate any non-standard process that may be required at a site when the UDB is updated.

For detailed information on command options, see the `udb_helper(8)` man page.

## 5.2 Batch Scheduling and Load Balancing

The Load Sharing Facility (LSF) Suite is a new workload management system for the Cray SV1 series system. LSF provides comprehensive load sharing and job scheduling across a Cray SV1 series SuperCluster system. It can also be used across a mix of Cray SV1 series systems and systems of other vendors. LSF is sold by Platform Computing. For information on using LSF, see the documentation available on the Platform Computing web site at the URL: <http://www.platform.com>.

The Network Queueing Environment (NQE) for UNICOS is a workload management system that automatically balances tasks across mixed servers attached to a network. NQE is in maintenance mode on Cray systems, and in

retired mode on other vendors' systems. See *NQE User's Guide* for basic information on using NQE.

### 5.3 Centralized Accounting

UNICOS provides two types of system accounting, standard UNIX System V accounting and Cray system accounting (CSA). You may use one or the other of these accounting packages at your site.

For basic information on using standard UNIX System V accounting, see *UNICOS Resource Administration*.

For introductory information on using CSA, which is the more complete and frequently used of the two accounting types, see the "Accounting" chapter in the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*. For more detailed information on CSA, see *UNICOS Resource Administration*.

Third-party software packages are available to roll up accounting data from all systems in a cluster into a single report. These packages include:

- PerfAcct and PerfStat from Instrumental, Inc.
- TeamQuest Baseline from TeamQuest Corporation

### 5.4 Backing Up and Restoring File Systems

The optional UNICOS Data Migration Facility (DMF) provides online file system space by moving selected files offline to a designated storage device(s). These files remain cataloged in their original directories and behave as if they are still disk resident. Likewise, an on-line disk can be considered a cached copy of a larger virtual disk space.

See *Cray Data Migration Facility (DMF) Release and Installation Guide* for basic information on using DMF.

See the "Backing Up and Restoring File Systems" chapter in the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems* for basic information on backing up and restoring file systems.

# Monitoring and Online Diagnostics From the SWS [6]

---

This chapter discusses the following topics:

- Monitoring the state of your cluster
- Folding and masking the GigaRing
- Message logs
- Preparing the system to run offline diagnostics

**Note:** This chapter does not cover basic system monitoring and online diagnostics in detail, nor does it cover using the automatic recovery feature to monitor a system. It includes only information necessary for the monitoring and diagnosing of Cray SV1 series SuperCluster systems. For information on basic system monitoring and online diagnostics topics, refer to the *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems* and the *SWS-ION Administration and Operations Guide*.

## 6.1 Monitoring the State of Your Cluster

You can use the `ops(8)` command to view the state of mainframes, I/O nodes (IONs), and rings. This command invokes a graphical user interface that allows you to easily view the state. Depending on your information needs, you can run the `ops` command on selected SWSs (and their associated SuperCluster building blocks) in the cluster or on all SWSs (and their associated SuperCluster building blocks) in the cluster.

The `ops(8)` command obtains its information from the `stateds(8)` and `hwmcontrol(8)` commands, which provide state information gathered from the `statesrvc(8)` state server daemon.

**Note:** The `hwmd(8)` command (which manages the execution of the check commands that supply information to the state server) is the only utility associated with `ops` that places traffic on the ring if the mainframe is not in a booted state; when diagnostics are run, the monitoring of all hardware components must be shut off by executing the following command:

```
hwmcontrol -a off
```

To resume monitoring of all hardware components, execute the following command:

```
hwmcontrol -a on
```

See the `hwmcontrol(8)` man page for more information.

The `statesrvc` and `hwmd` daemons must be running before you invoke `ops`. For information on how to enable the automatic starting of these daemons, see the *SWS-ION Administration and Operations Guide*.

After starting the state server and hardware monitor, run the following command to populate the state server:

```
bootsys
```

In subsequent sessions, the information maintained by the state server will be used.

### 6.1.1 Viewing the System State

On the SWS, run the `ops(8)` command in the background:

```
sws$ ops &
```

**Note:** If you need to run `ops` remotely, refer to the *SWS-ION Administration and Operations Guide*.

By default, `ops(8)` displays the status of mainframes, IONs, and rings, plus `hwmcontrol(8)` information. (You can also set the `ops` command to display the `stateds(8)` pipe information.) If the `hwmcontrol` status bar shows the word `DISCONNECTED`, a pipe to either `hwmcontrol(8)` or `stateds(8)` has been lost which usually indicates that the state server is not running.

You can modify which components `ops` will display, as well as colors and font size, from within the tool.



Table 4 describes the state groupings and their default color representation; additional information about each state (such as `RUN LEVEL`) is also supplied for each item within the `ops` display.

Table 4. Default State and Color Representation

State Grouping	Color
RUNNING	Green
TRANSITIONS	Yellow
STOPPED	Red
ERRORS	Red
UNKNOWN	Gray

The groupings listed in Table 4 contain specific states, each of which can have a different color representation. For example, the `RUNNING` group consists of the following states, each of which can have a unique color: `BOOTED`, `CONNECTED`, `INITIALIZED`, and `RUN LEVEL`. You can also specify font size.

You can specify system-wide configuration information by modifying the `ops` options and then saving the current configuration to a file named `/opt/config/opsrc`. Each user can also create an individual `$HOME/.opsrc` resource file, which takes precedence. If the `$HOME/.opsrc` and `/opt/config/opsrc` files are not present, the `ops` utility has a built-in set of defaults, as described in Table 4.

To invoke `ops` using a specific configuration file, enter the following command:

```
ops -c config_file
```

See the online help provided with the `ops` tool for more information about states and configuration. You can also obtain state information in a non-graphical format by using the `stateds(8)` and `hwmcontrol(8)` commands; see the man pages for details.

If you want to minimize the amount of space that the `ops` window takes up on your workspace, you can iconify it. If something changes within the `ops` display while it is iconified, an asterisk will be prepended to the icon name. For example, the name would change from `OPS (wslg6)` to `* OPS (wslg6)`. When you open the window, the asterisk is cleared from the icon name.

## 6.2 Folding and Masking the GigaRing

GigaRing channel resiliency is provided by ring folding and masking:

- *Ring folding* allows concurrent maintenance of GigaRing node I/O controllers to be carried out. By using the `dring(8)` or `diagring(8)` diagnostic programs, the GigaRing channel can be electronically looped back at the two nodes adjacent to a failed node or cable to isolate it from the rest of the ring. Once the loopback is completed, the failed node or cable can be removed or replaced, after which the GigaRing channel can be unfolded. The GigaRing channel runs at half bandwidth while it is folded.
- *Ring masking* allows maintenance of a GigaRing channel on a running system. If one of the two rings fails, the GigaRing channel can be masked, with communications continuing on the other ring. When the channel is repaired, it can be unmasked.

To mask a ring, a special value is written programmatically (by the `dring` or `diagring` program) into a memory-mapped register (MMR) on each node. This action prevents the nodes from transmitting packets to the faulty ring. The GigaRing channel runs at half bandwidth while it is masked.

In addition to performing resiliency operations (such as ring masking or folding), the `dring` and `diagring` programs let you view information about GigaRing channels and their associated nodes and execute diagnostic commands. Diagnostic commands provide you with the ability to run specific node and ring tests, to isolate failures to specific components, and to take either automated or manual corrective actions when possible.

See the `dring(8)` and `diagring(8)` man pages and the chapter on “GigaRing Diagnostic Tests” in the *SWS-ION Administration and Operations Guide* for detailed information on the `dring` and `diagring` programs.

## 6.3 Message Logs

The use and contents of Cray SV1 series SuperCluster message logs are identical to single (non-clustered) GigaRing systems. For a description of the log files that are important for you to monitor, see the “Log Files” chapter in *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Series Systems*. For information about accounting logs and reports, see the “Accounting” chapter in this manual. For information on the SWS-ION message logs, see the *SWS-ION Administration and Operations Guide*.

You can use the `logmaint(8)` command to archive and restart SWS `/opt/CYRilog` files. `logmaint` is typically run via the SWS root `crontab(1)` as follows:

```
00 3 * * 0 /opt/CYRicmt/etc/logmaint 9 > /dev/null 2>&1
```

With a cluster of systems, it may be necessary to run `logmaint` more frequently than the "once per week" given in the preceding example. Your site may want to run `logmaint` two to three times per week, depending on how many nodes (mainframes) exist in the cluster.

## 6.4 Preparing the System to Run Offline Diagnostics

To perform offline diagnostics on one of the systems in the cluster, you must first isolate that system; that is, you must fold out that mainframe on every GigaRing channel to which it is attached other than on the private GigaRing channel used to boot that system.

Isolating a system can be done with the following `dring` command:

```
dring -a foldoutmf -m mfname -b
```

This command folds out all nodes of the specified mainframe on all GigaRing channels, except for the node on its boot GigaRing. Because the boot GigaRing must be a private ring, no other mainframe will be affected by further diagnostic operations done via this node.

At this point, the private boot GigaRing and all nodes attached to it can be used for offline diagnostic testing. See the *Cray SV1 Mainframe Troubleshooting* hardware manual for information on available offline diagnostics. After diagnostics and repairs are completed the following two actions are required to bring the system physically back into the cluster configuration:

```
dring -a foldinmf -m mfname  
bootsys mfname
```



# System Configuration File Examples [A]

---

This appendix contains examples of the default `/etc/config/param` and `/etc/config/udb/clusterUDB.conf` files

## A.1 `/etc/config/param` File

This section contains an example of the default `/etc/config/param` file. The configuration specification language (CSL) parameter file contains statements that specify hardware and software characteristics for your system. Each Cray system in the SuperCluster system has unique `param` files to handle the private file systems that the systems will use. When the configuration is activated, a copy of the CSL parameter file is stored in `/etc/config/param` file. For all Cray systems, the administrator must manually transfer the parameter file to the workstation.

**Note:** It is strongly recommended that you use the install tool to maintain your system configuration, rather than manually editing the configuration specification language (CSL) parameter file. For more information on the UNICOS installation and configuration menu system (ICMS), see *UNICOS System Configuration Using ICMS* and the online install tool help files.

```
/* Configuration parameter file
*/

revision "sn3001";

/* GigaRing Information
*/
gigaring {
/*
* BEGIN SECTION: GigaRing Information
*/
gr_route {
ring 00 {
channel 024;
}
ring 040 {
channel 034;
}
}
}
```

```
/*
 * END SECTION: GigaRing Information
 */
}

/* Hardware Information
 */
mainframe {
/*
 * BEGIN SECTION: Hardware Information
 */
    32 cpus;
    1024 Mwords memory;

/* Channel information
 */
    channel 024 is gigaring to ring 00, node 025;
    channel 034 is gigaring to ring 040, node 025;
/*
 * END SECTION: Hardware Information
 */
}

/* UNICOS configuration
 */
unicos {
/*
 * BEGIN SECTION: Kernel Parameters
 */
    4096 NBUF; /* number of system buffer headers */
    2048 NLDCH; /* number of ldcache headers */
    32768 LDCHCORE; /* number of ldcache memory clicks reserved */
    118 LDDMAX; /* maximum number of logical devices */
    20 XDDMAX; /* maximum number of XDD devices */
    256 XDDSLMAX; /* maximum number of XDD slices */
    8 RDDSLMAX; /* maximum number of RAM slices */
    12 TAPE_MAX_CONF_UP;
    65536 TAPE_MAX_PER_DEV;
/*
 * END SECTION: Kernel Parameters
 */
}
```

```

/* Disk Configuration
*
* Special System Devices
*/
filesystem {
/*
* BEGIN SECTION: Disk Configuration
*/
/* Physical device configuration
*/
/* TYPE: MPN
* NODE: sn3001-mpn0
* PRIVATE IP ADDR: 10.1.124.201
* PRIVATE ETHERNET ADDR: 00.03.31.02.46.a9
*/
/* Disk Serial Number: 018935102100 */
xdisk d000425.0 {iounit 1;
    iopath {ring    00; node 042; channel 05; }
    unit 0;
    xdd roota_1    {minor    1;sector        0;length    150000 sectors;}
    xdd usra_2     {minor    2;sector    150000;length    400000 sectors;}
    xdd rootb_3    {minor    3;sector    550000;length    150000 sectors;}
    xdd usrb_4     {minor    4;sector    700000;length    400000 sectors;}
    xdd srca_5     {minor    5;sector   1100000;length    220000 sectors;}
    xdd srcb_6     {minor    6;sector   1320000;length    220000 sectors;}
    xdd opt_7      {minor    7;sector   1540000;length    300000 sectors;}
    xdd tmp_8      {minor    8;sector   1840000;length    250000 sectors;}
    xdd home_9     {minor    9;sector   2090000;length    150000 sectors;}
    xdd swap_10    {minor   10;sector   2240000;length      2634 sectors;}
    xdd dump_11    {minor   11;sector   2242634;length    100000 sectors;}
}
/* Disk Serial Number: 015496122100 */
xdisk d000425.1 {iounit 1;
    iopath {ring    00; node 042; channel 05; }
    unit 1;
    xdd swap_12    {minor   12;sector        0;length    997366 sectors;}
    xdd disk_13    {minor   13;sector    997366;length    104634 sectors;}
}
/* Disk Serial Number: 015640932100 */
xdisk d000425.2 {iounit 1;
    iopath {ring    00; node 042; channel 05; }
    unit 2;
    xdd disk_14    {minor   14;sector        0;length    1102000 sectors;}

```

```
}
/* Disk Serial Number: 024100492100 */
xdisk d000425.3 {iounit 1;
    iopath {ring    00; node 042; channel 05; }
    unit 3;
    xdd disk_15    {minor 15;sector          0;length  1102000 sectors;}
}
/* TYPE: FCN
 * NODE: sn3001-fcn0
 * PRIVATE IP ADDR: unknown
 * PRIVATE ETHERNET ADDR: 00:40:a6:80:05:2b
 */
/* Disk Serial Number: LK030426 */
xdisk d000410.1 {iounit 1;
    iopath {ring    00; node 041; channel 00; }
/* alternate iopath {ring    00; node 041; channel 01; } */
    unit 1;
    xdd disk_16    {minor 16;sector          0;length  4902456 sectors;}
}
/* Disk Serial Number: LK031251 */
xdisk d000410.2 {iounit 1;
    iopath {ring    00; node 041; channel 00; }
/* alternate iopath {ring    00; node 041; channel 01; } */
    unit 2;
    xdd disk_17    {minor 17;sector          0;length  4902456 sectors;}
}
/* Disk Serial Number: LK037116 */
xdisk d000410.3 {iounit 1;
    iopath {ring    00; node 041; channel 00; }
/* alternate iopath {ring    00; node 041; channel 01; } */
    unit 3;
    xdd disk_18    {minor 18;sector          0;length  4902456 sectors;}
}
/* Logical device configuration
 */
ldd dump { minor 1;
    xdd dump_11;
}
ldd roota { minor 2;
    xdd roota_1;
}
ldd usra { minor 3;
    xdd usra_2;
```



```
}  
ldd rootb { minor 4;  
  xdd rootb_3;  
}  
ldd usrb { minor 5;  
  xdd usrb_4;  
}  
ldd srca { minor 6;  
  xdd srca_5;  
}  
ldd srcb { minor 7;  
  xdd srcb_6;  
}  
ldd opt  { minor 8;  
  xdd opt_7;  
}  
ldd tmp  { minor 9;  
  xdd tmp_8;  
}  
ldd home { minor 10;  
  xdd home_9;  
}  
ldd swap { minor 11;  
  xdd swap_10;  
  xdd swap_12;  
}  
ldd disk0 { minor 12;  
  xdd disk_13;  
}  
ldd disk1 { minor 13;  
  xdd disk_14;  
}  
ldd disk2 { minor 14;  
  xdd disk_15;  
}  
ldd disk3 { minor 15;  
  xdd disk_16;  
}  
ldd disk4 { minor 16;  
  xdd disk_17;  
}  
ldd disk5 { minor 17;  
  xdd disk_18;
```

```
}
/*
 * END SECTION: Disk Configuration
 */
/*
 * BEGIN SECTION: Special System Devices
 */
rootdev is ldd roota;
swapdev is ldd swap;
dmpdev is xdd dump_11;
/*
 * END SECTION: Special System Devices
 */
}

/* Network configuration
 */
network {
/*
 * BEGIN SECTION: Network Configuration
 */
    9076 tcp_nmbospace;
    0700 hidirmode;
    0600 hifilemode;

/* If ghippi network devices are present in the
 * param file and you wish to use ICMS to manage
 * the param file, then there are certain rules
 * that you need to follow with regard to the
 * ordering of the network devices in the param
 * file. The network devices must be grouped
 * together by device type (i.e. ghippi*, gether*,
 * gfddi*, gatm*). The ghippi device group must
 * appear first, followed by the other device
 * groups. Failure to do this will cause ICMS to
 * create minor device numbers that may not match
 * those expected by the kernel.
 */

/* Ethernet Network Devices
 */
gether 0 {
    iopath {ring    00; node 042; channel 02; }
```

```

}

/* GigaRing Host-to-Host Interfaces
*/
gr 0 {
  iopath {ring 00; node 025; }
}
gr 1 {
  iopath {ring 040; node 025; }
}
/*
* END SECTION: Network Configuration
*/
}

```

## A.2 /etc/config/udb/clusterUDB.conf File

This section contains an example of the default cluster UDB configuration file. The cluster UDB configuration file contains locally specified defaults and preferences and the cluster UDB source and update distribution files. By default, the file is located in /etc/config/udb and is called clusterUDB.conf.

**Note:** All of the cluster UDB administration tools expect the configuration file to be named clusterUDB.conf so it is best not to change the name.

The configuration file should be shared among the cluster nodes with NFS or some other file sharing mechanism so that there is only one configuration file to maintain. The recommended way to do this is to place a symbolic link in /etc/config/udb named clusterUDB.conf linking to the actual configuration file.

The default configuration file is set up assuming that the home file systems are shared among the cluster nodes and the UDB administrator has a special login named UDBadmin. The home directory contains the directories Collect, Config, Dist, Edit, and Log. The administrator will be required to alter the path names as necessary for the local environment.

### A.2.1 General Directives

The configuration file consists of directives containing keywords and parameters. The directives may be listed in any order.

Syntax rules for the configuration file are as follows:

- The maximum line length is 255 characters. Backslash continuation is not supported.
- Blank lines are ignored.
- Comment text begins with the character #. This character, and the remainder of the line, are discarded.
- Fields are separated by blanks or tabs.
- Keywords are shown in uppercase letters in this description. However, the capitalization of keywords is optional.
- Each cluster UDB administration tool parses the configuration file using a parser tailored to its needs. Only keywords of interest to the specific tool are examined in detail. For this reason each tool may accept or reject a configuration file independently of the other tools.

General configuration file directives are as follows:

CollectionDirectory *directory\_name*

Names the directory where node UDB source files will be placed by `udb_collect` and where they will be found by `udb_merge`.

CommandDirectory *directory\_name*

Names the directory from which `udb_server` executes commands. For security reasons, only commands actually needed by `udb_server` should appear in this directory. Currently, only `udbgen`, `udbsee`, and `udb_helper` need to be installed.

DistributionDirectory *directory\_name*

Names the directory where cluster UDB source files will be placed by `udb_collect` and where they will be found by `udb_update`.

DistributionFile *file\_name*

Names the prefix of the file name where edited cluster UDB source files will be written by `udb_update` and where they will be found by `udb_server`.

LogDirectory *directory\_name*

Names the directory where logging and reporting files will be placed by all the cluster UDB administration tools.

EditDirectory *directory\_name*

Names the directory in which to edit user records.

ReportFile *file\_name*

Many of the tools use a report file to record information that is useful in assessing the success of an operation. Unless overridden by tool options, the released file name is Log in LogDirectory.

ClusterSource *file\_name*

The cluster UDB source file is used by many of the tools. Unless overridden by tool options, the released merge file name is ClusterSource in CollectionDirectory.

NodeList *node<sub>0</sub> node<sub>1</sub> ... node<sub>n</sub>*

Lists the names of all the nodes in the cluster. Any number of space separated names, up to the maximum line length, may appear. Only nodes named in NodeList directives are managed by the cluster UDB administration tools. If multiple directives appear, the list of nodes is the concatenation of all the NodeList directives in the file. The names must be sufficiently qualified to act as network addresses in the local environment.

ChangeLog *file\_name*

Copies of the change files distributed to the nodes are recorded in this file. When the entire cluster UDB source is distributed only a notice that it was done appears in this file. The released name of this file is ChangeLog in LogDirectory.

Timeout *seconds*

Timeout is used by the tools that connect to other nodes, namely `udb_collect` and `udb_update`. If a node fails to report that a request has completed within the configured time,

the request is assumed to have failed. The released timeout value is 90 seconds.

### A.2.2 Record Merging Directives

Record merging configuration file directives are as follows:

`Exempt field operator value`

Records matching this directive are not checked for compliance with the uniformity rule. Only one set of parameters may appear following the `Exempt` keyword, but as many `Exempt` directives as needed may appear. The syntax of the parameters is as follows:

*field* may take the values `name` or `uid`.

*operator* may take the values `<`, `==`, `>`, or `!=`.

*value* is an appropriate value. For example, to exempt all UIDs less than 100, use the following directive: `Exempt uid < 100`. To exempt a record with the user name operator, use the following directive: `Exempt name == operator`.

`Uniform field0 field1 ... fieldn`

The fields named with this directive must be uniform in all but exempt records. Any number of space separated fields, up to the maximum line length, may appear. The field names from all `Uniform` directives are concatenated into a single list. For example, to make fields `acids` and `gids` uniform, use the following directive: `Uniform acids gids`. The directive `Uniform name uid` is required to be present by `udb_merge`.

`Drop field0 field1 ... fieldn`

The fields named with this directive will be removed from the merged cluster UDB source file. Any number of space separated fields, up to the maximum line length, may appear. The field names from all `Drop` directives are concatenated into a single

list. For example, to drop fields `batchhost` and `logtime`, use the following directive: `Drop batchhost logtime`.

### A.2.3 New Record Template Directive

The record creation option of `udb_edit` uses the new record template directive, `Newfield`, to establish default values for all fields in the record. Fields may be removed from this group if their `udbgen` defaults are satisfactory. The order of the field names will be maintained in the new user template record. You may reorder the list to move fields usually modified to a more convenient location. The string `$USERNAME` will be replaced in the new record by the user name supplied to `udb_edit`. The new record template configuration file directive is as follows:

```
Newfield field1:value1: field2:value2: ... fieldn:valuen:
```

The field names and values from all `Newfield` directives are concatenated into a single list. Any number of space separated field name, field value pairs, up to the maximum line length, may appear. The `udbgen` source format rules are used to parse the name/value pairs, making the colon delimiters required. For example, use the following directive to specify the login shell as `/bin/sh`: `Newfield shell :/bin/sh:`.

### A.2.4 Default Cluster UDB Configuration File

The default cluster UDB configuration file is as follows:

```
# USMID @(#)admin/udb/clusterUDB.conf 100.2 03/08/2000 16:41:48
# Cluster UDB configuration file.
# This is the default configuration file. Please do not change this
# file without commenting that the file is no longer as-released.
#
# The standard name for this file is /etc/config/udb/clusterUDB.conf
# The udb_server will not run unless it can open the file under
# that name. Normally the configuration files are on a shared
# filesystem so the entry in /etc/config/udb is a symbolic link
# to the real file.
## Configuration file section for commands that need to know
# the names of the working directories, default file names and the
# names of the nodes.
#
```

```
#      Directory in which to collect node UDB source files. The
#      released name is /usr/udbAdmin/Collect
CollectionDirectory    /usr/udbAdmin/Collect
#
#      Directory from which udb_server executes commands. For security
#      reasons only commands actually needed by udb_server should
#      appear in this directory. At this time only udbgen, udbsee and
#      udb_helper are needed in this directory.
#      The released name is /usr/adm/udb/bin.
CommandDirectory      /usr/adm/udb/bin
#
#      Directory from which to distribute node UDB source files. The
#      released name is /usr/udbAdmin/Dist
DistributionDirectory  /usr/udbAdmin/Dist
#
#      The cluster UDB source file is used by many of the tools. Unless
#      overridden by tool options, the released merge file name is
#      ClusterSource in DistributionDirectory.
ClusterSource    ClusterSource
#
#      File name prefix for edited node UDB source files. The
#      released prefix is Dist to which the tools add yyyymmddhhmmss
DistributionFile    Dist
#
#      Directory in which to edit user records. The released name
#      is /usr/udbAdmin/Edit
EditDirectory      /usr/udbAdmin/Edit
#
#      Directory in which to write log and report files. The
#      released name is /usr/udbAdmin/Log
LogDirectory       /usr/udbAdmin/Log
#
#      Many of the tools use a report file to record information useful
#      to assess the success of the operation. Unless overridden by tool
#      options, the released prefix is Log in LogDirectory.
ReportFile         Log
#
#      Changes applied by udb_update are recorded in the change log.
#      The released name is ChangeLog.
ChangeLog          ChangeLog

##      Configuration file section for network information.
#
```



```

#       The node list is used by the collectors and distributors to know
#       which nodes to contact. The names are also needed for the node
#       directives in the cluster UDB source file. The name must be
#       sufficiently qualified to be a valid network node name.
NodeList      node0.v.com node1.v.com node2.v.com node3.v.com
#
#       Timeout is used by udb_collect and udb_update to detect
#       unresponsive nodes. It must be set high enough to avoid
#       false alarms. The released default is 90 seconds.
Timeout 90

##      Configuration file section for udb_merge
#
#       This file tells udb_merge how to handle its input. Records
#       may be exempted from the rules by naming their uid or name
#       fields in an exemption rule. All records not explicitly made
#       exempt must follow the rules.
#       About field names: Use the udbgen names for all fields. The user
#       name has the field name 'name' in the exemption rule.
#
#       Formats: (Keywords are case insensitive.)
#       * Exemption rule. These records need not follow the rules
#       Exempt uid|create op value
#           op is one of these operators: <, ==, >, !=
#       Example: To exempt all UIDs less than 100-
#           Exempt uid < 100
#           To exempt a record named operator-
#           Exempt name = operator
#
#       * Uniformity rule. These fields must be uniform in all but
#       exempt records.
#       Uniform fn fn fn ...
#           Any number field names (fn) may appear on
#           one or more 'uniform' lines. All field names
#           are concatenated into a single list.
#       Example: Make these fields uniform over the cluster-
#       Uniform name uid passwd pwage gids acids dir shell
#
#       * Drop rule. These fields will be dropped from the merged
#       file. Local fields are usually members of this
#       group.
#       Drop fn fn fn ...
#           Any number field names (fn) may appear on

```

```
#                               one or more drop lines. All field names
#                               are concatenated into a single list.
#                               Example: Drop these fields from the cluster UDB-
#                               Drop lastlogtime logfails loghost logline logtime shusage

Exempt uid < 100
Exempt name == MyName
#     name and uid need to be here to make record skip work correctly
Uniform name uid
Uniform acids adminmax archlim archmed comment comparts cpuquota
Uniform defcomps deflvl dir disabled gids intcat
Uniform jcpulim[b] jcpulim[i] jfilelim[b] jfilelim[i] jmemlim[b] jmemlim[i]
Uniform jproclim[b] jproclim[i] jsdslim[b] jsdslim[i]
Uniform jshmsecs[b] jshmsecs[i] jshmsize[b] jshmsize[i] jsocbflim[b] jsocbflim[i]
Uniform limflags maxlvl mincomps minlvl nice[b] nice[i] parentuid
Uniform passwd pcorelim[b] pcorelim[i] pcpulim[b] pcpulim[i] permbits
Uniform permits pfdlimit[b] pfdlimit[i] pfilelim[b] pfilelim[i]
Uniform pmemlim[b] pmemlim[i] psdslim[b] psdslim[i]
Uniform pwage resgrp root shares shell shflags sitebits trap valcat
Drop batchhost batchtime cpuquotausd lastlogtime logfails loghost logline
Drop logtime shcharge shextime shusage

##      Configuration file section for the new user UDB record.
#      The NewField keyword is needed on each line to parse this section.
#
#      The defaults should be edited to fit the needs of the site.
#
#      Note:
#          'change' is required . Do not use 'create' instead!
#          'cpasswd' is used here instead of 'passwd'
#          'pwage' is used for aging information
#
#      Fields may be removed from this list if their udbgen defaults
#      are satisfactory. The order of this list will be maintained in
#      the new user template file. The administrator may reorder the
#      list to move fields usually modified in a convenient place.
#
#      The string '$USERNAME' will be replaced in the template file by
#      the actual user name.
NewField      change :$USERNAME: uid :next:
NewField      cpasswd :NEW.$USERNAME.PASSWORD:
NewField      dir ::
NewField      gids ::
```

```
NewField      pwage :force:
NewField      shell :/bin/sh:
NewField      acids ::
NewField      adminmax :0:
NewField      archlim :0:
NewField      archmed :0:
NewField      batchhost ::
NewField      batchtime :0:
NewField      comment ::
NewField      comparts :0:
NewField      cpuquota :0:
NewField      cpuquotaused :0:
NewField      defcomps :0:
NewField      deflvl :0:
NewField      disabled :0:
NewField      intcat :0:
NewField      jcpulim[b] :unlimited:
NewField      jcpulim[i] :unlimited:
NewField      jfilelim[b] :unlimited:
NewField      jfilelim[i] :unlimited:
NewField      jmemlim[b] :unlimited:
NewField      jmemlim[i] :unlimited:
NewField      jproclim[b] :100:
NewField      jproclim[i] :100:
NewField      jsdslim[b] :none:
NewField      jsdslim[i] :none:
NewField      jshmsecs[b] :unlimited:
NewField      jshmsecs[i] :unlimited:
NewField      jshmsize[b] :unlimited:
NewField      jshmsize[i] :unlimited:
NewField      jsocbflim[b] :unlimited:
NewField      jsocbflim[i] :unlimited:
NewField      jtapelim[b][0] :none:
NewField      jtapelim[b][1] :none:
NewField      jtapelim[b][2] :none:
NewField      jtapelim[b][3] :none:
NewField      jtapelim[b][4] :none:
NewField      jtapelim[b][5] :none:
NewField      jtapelim[b][6] :none:
NewField      jtapelim[b][7] :none:
NewField      jtapelim[i][0] :none:
NewField      jtapelim[i][1] :none:
NewField      jtapelim[i][2] :none:
```

NewField	jtapelim[i][3] :none:
NewField	jtapelim[i][4] :none:
NewField	jtapelim[i][5] :none:
NewField	jtapelim[i][6] :none:
NewField	jtapelim[i][7] :none:
NewField	lastlogtime :0:
NewField	limflags :0:
NewField	loghost ::
NewField	logfails :0:
NewField	logline ::
NewField	logtime :0:
NewField	maxlvl :0:
NewField	mincomps :0:
NewField	minlvl :0:
NewField	nice[b] :0:
NewField	nice[i] :0:
NewField	parentuid :0:
NewField	pcorelim[b] :unlimited:
NewField	pcorelim[i] :unlimited:
NewField	pcpulim[b] :unlimited:
NewField	pcpulim[i] :unlimited:
NewField	permbits :0:
NewField	permits :0:
NewField	pfddlimit[b] :64:
NewField	pfddlimit[i] :64:
NewField	pfilelim[b] :unlimited:
NewField	pfilelim[i] :unlimited:
NewField	pmemlim[b] :unlimited:
NewField	pmemlim[i] :unlimited:
NewField	psdslim[b] :none:
NewField	psdslim[i] :none:
NewField	resgrp :0:
NewField	root :0:
NewField	shares :0:
NewField	shcharge :0:
NewField	shextime :0:
NewField	shflags :0:
NewField	shusage :0:
NewField	sitebits :0:
NewField	trap :0:
NewField	valcat :0:

# Glossary [B]

---

## Building block

See *SuperCluster building block*.

## Capability cluster

A specific workload is spread across multiple nodes with the use of system functions, usually using a parallel database.

## Capacity cluster

Management and allocation of resources from across the cluster, but where each application still runs on one node. (This includes redundant resources for restarting applications.) The Cray SV1 series SuperCluster is a capacity cluster.

## Cluster

A collection of multiple machines coupled to each other by networks or other similar interconnects.

## Direct connection

Configuration in which each mainframe in the Cray SV1 series SuperCluster is connected to every other mainframe in the cluster via shared GigaRing channels, with multiple mainframes on each shared GigaRing.

## Domain

A domain is specified in the GigaRing topology file. It consists of the specified mainframes themselves plus the GigaRing channels and I/O nodes (IONs) that are connected to only the specified mainframes. Topology domains enable system-level SWS operational commands such as `bootsys(8)` to execute on the domain of specified mainframes.

## /etc/config/param file

See *parameter file*.

`/etc/config/rcoptions`

Files used to control the startup behavior of each system in the SuperCluster. This file is controlled by the UNICOS Installation/Configuration Menu System (ICMS) from various component parts, which are in turn either generated by ICMS, or maintained manually by the system administrator.

`/etc/config/udb/clusterUDB.conf` file

Cluster user database (UDB) configuration file that contains locally specified defaults and preferences and the cluster UDB source and update distribution files.

`/etc/shutdown.*` files

Files executed at various points during system shutdown.

## Folding

Ring folding allows concurrent maintenance of GigaRing node I/O controllers to be carried out. By using the `diagring` or `dring` diagnostic program, the GigaRing channel can be electronically looped back at the two nodes adjacent to a failed node or cable to isolate it from the rest of the ring. Once the loopback is completed, the failed node or cable can be removed or replaced, after which the GigaRing channel can be unfolded. The GigaRing channel runs at half bandwidth while it is folded.

## GigaRing topology file

An ASCII file consisting of statements that describe the physical layout of mainframe and I/O nodes on one or more GigaRing channels.

## Inter-node GigaRing

Inter-node GigaRing connections are made between two or more SuperCluster building blocks. The inter-node GigaRing connection is used to provide common disk and network I/O capabilities to the SuperCluster system.

## Intra-node GigaRing

Intra-node GigaRing connections are made between the four Cray SV1 series systems that make up a SuperCluster building block. The intra-node GigaRing connection is used to provide

common disk and network I/O capabilities to the SuperCluster system.

#### I/O node (ION)

The Cray scalable I/O (SIO) architecture consists of I/O nodes (IONs) connected by the high-speed GigaRing channel. These IONs connect I/O peripherals to the channel to provide various I/O services to system nodes.

#### Masking

Ring masking allows maintenance of a GigaRing channel on a running system. If one of the two rings fails, the GigaRing channel can be masked, with communications continuing on the other ring. When the channel is repaired, it can be unmasked. To mask a ring, a special value is written programmatically (by the `diagring` or `dring` program) into a memory-mapped register (MMR) on each node. This action prevents the nodes from transmitting packets to the faulty ring. The GigaRing channel runs at half bandwidth while it is masked.

#### Matrix connection

Configuration in which each mainframe in the Cray SV1 series SuperCluster is connected to some of the mainframes in the cluster via shared GigaRings. To get to mainframes that are not on the same shared GigaRing channels, one mainframe from the shared GigaRing IP forwards the communication to another GigaRing where the target mainframe resides. This means that there is at most one IP forwarding of communication between any two mainframes in the SuperCluster system.

#### MPN

Multi-purpose nodes (MPNs), which provide an interface based on the SBus standard to support industry-standard I/O channels. There is one MPN per mainframe in the Cray SV1 series SuperCluster.

#### Node

Each processing cabinet within a SuperCluster building block. Each SuperCluster building block can contain up to eight nodes. A node is a Cray SV1 series system with 2 to 32 Gbytes of main memory and 8 to 32 1-Gflops CPUs.

#### Options file

An ASCII file consisting of statements that provide site-specific defaults to the low-level SWS commands (such as the `bootsv1(8)` command) that are invoked by the system-level SWS commands (such as the `bootsys(8)` command).

#### Parameter file

The configuration specification language (CSL) parameter file contains statements that specify hardware and software characteristics for your system. When the configuration is activated, a copy of the CSL parameter file is stored in `/etc/config/param`. Each system in the SuperCluster system has unique `param` files to handle the private file systems that the systems will use

#### Private GigaRing

Each Cray SV1 series system in the SuperCluster system has a private GigaRing connection that contains at least one I/O node containing the system's primary and secondary `root`, `usr`, and `src` file systems. The private GigaRing connection may contain additional I/O nodes to provide unique disk and network I/O capabilities to the single Cray SV1 series system.

#### SIO

The Cray scalable I/O (SIO) architecture consists of a system of I/O nodes (IONs) connected by a high-speed system channel called the *GigaRing channel*.

#### SPN

Single-purpose nodes (SPNs), which support specific network interfaces and/or devices.

#### SuperCluster building block

The minimum hardware configuration for each Cray SV1 series SuperCluster system, which is a Cray SV1 series-4 system. Each SuperCluster building block includes the following components:

- System workstation (SWS)
- Four Cray SV1 series mainframes (four processing cabinets) with two peripheral cabinets



- Channel connection hardware (module connectors and cables) to make up to five SIO (scalable I/O) GigaRing channels:
  - One private GigaRing channel per mainframe, each with an ION (for a total of four).
  - One GigaRing channel per SuperCluster building block for intra node communication, connected to all four mainframes with an ION.

The number of additional GigaRing channels accessible from each fully populated system depends on how many CPUs are configured in the system, up to a maximum of six additional GigaRing channel connections per Cray SV1 series system (with 32 CPUs in the system).

Cray SV1 series-8 or larger systems also include one GigaRing channel per mainframe for inter node communication (for a total of three). (Note that each mainframe is limited to eight GigaRing channels.)

#### System Workstation (SWS)

The system workstation (SWS) for a SuperCluster system is a Sun workstation running the Solaris operating system.

This workstation serves as the system console for each SIO device and the Cray SV1 series SuperCluster building block. The workstation also runs management and maintenance software for the Cray SV1 series system and SIO devices. This console is connected via a private Ethernet connection to each SIO. It is shipped with a DAT tape device for backups.

#### Topology file

See *GigaRing topology file*.



## A

- Accounting, 60
- arp command, 45
- Automatic recovery, 20

## B

- Backups, file system, 60
- Batch scheduling, 59
- BDS, 9
- Booting, 21
  - Domains, 21
  - Everything, 21
  - Individual mainframes, 22
- Booting system, 20
- bootsv1 command, 21
- bootsys command, 20
- Bringing up multiuser mode, 23

## C

- Capacity cluster, 1
- cfsync command, 31
- cfupdate command, 31
- Client hardware node, 11
- clsh command, 20
- Cluster Bundle, 9
- Cluster Bundle software, configuring, 51
- Cluster UDB, 36
- Cluster, definition of, 1
- Configuration changes, validating, 18
- Configuration specification language (CSL)
  - parameter file, 35
- Console, 3, 87
- Console, starting, 19, 21, 22, 27
- Consoles, 19

## Cray SV1 series

### rings

- Inter-node GigaRing, 8
- Intra-node GigaRing, 8
- Private GigaRing, 8

## Cray SV1 series SuperCluster system

- Configuration illustrations, 5
- Configurations, 5
- Hardware components, 2
- Mainframe configuration, 29
- Mainframe operations, 53
- Monitoring and online diagnostics, 61
- Overview, 1
- Software components, 8
- SWS configuration, 11
- SWS operations, 19

- Cray SV1-4 SuperCluster system, illustration, 2

- Cray System Accounting, 60

## D

- Data migration facility, 52, 60
- Data Migration Facility (DMF), 10
- Definitions of terms, 83
- Diagnostics, 65
- diagring program, 64
- Direct connection configuration, 5
- DMF, 52, 60
- Domains, 17
- dring program, 64
- Dumping, 24
  - All mainframes and I/O nodes, 25
  - Domains, 25

## E

- /etc/config/interfaces file, 45
- /etc/config/param file, 35
- /etc/config/param file, example, 67
- /etc/config/rc.mid file, 50
- /etc/config/rc.pre file, 50
- /etc/config/rc.pst file, 50
- /etc/config/rcoptions file, 48
- /etc/config/udb/clusterUDB.conf file, 36
- /etc/config/udb/clusterUDB.conf file, example, 73
- /etc/gated.conf file, 46
- /etc/gr0.arp file, 45
- /etc/gr1.arp file, 45
- /etc/shutdown.\* files, 50
- /etc/shutdown.mid file, 50
- /etc/shutdown.pre script, 51
- /etc/shutdown.pst file, 50
- /etc/shutdown script, 26

## F

- File
  - Synchronization, 31
- File system
  - Layout, 29
  - Sharing, 31
  - Strategies, 30
- File system backups and restoration, 60
- Folding the GigaRing, 64

## G

- GigaRing node chip, 11
- GigaRing numbering scheme, 14
- GigaRing resiliency operations (folding and masking), 64
- GigaRing scalable I/O channel, 4
- GigaRing topology file, 11
- GigaRing/node numbering conventions, 13

- Glossary, 83
- Groups, maintaining, 53

## H

- Halting mainframes, 28
- haltsys command, 28
- Hardware
  - Components, 2
  - Minimum configuration, 2
- Hardware monitor, 61
- \$HOME/.opsrc file, 63
- hwmcontrol command, 61, 62
- hwmd command, 62

## I

- I/O nodes (IONs), 4
- Inter-node GigaRing, definition, 8
- Intra-node GigaRing, definition, 8
- Introduction, 1
- IP address-to-hardware mapping, 45
- IP addressing, 42

## L

- levelsys command, 23, 26
- Load balancing, 59
- Local file systems, 30
- logmaint command, 65
- Logs, 64
- LSF, 10, 59

## M

- Mainframe
  - Configuration, 29
  - Console, starting, 19, 21, 22, 27

- Operations, 53
- Mainframe consoles, 19
- Mainframe operations, 53
- Maintaining the user database, 53
- Maintaining users and groups, 53
- Masking the GigaRing, 64
- Master files, 31
- Matrix connection configuration, 6
- mbufs, section of param file, 36
- Message logs, 64
- Monitoring
  - State of your cluster, 61
- MPI, 52
- MPT, 9, 52
- Multiuser mode, bringing up, 23

## N

- Network setup, 39
- network, section of param file, 35
- NFS/BDS, 51
- Node, definition of, 1, 3
- NQE, 51, 59
- NQE for UNICOS, 9

## O

- Offline diagnostics, 65
- ONC+, 9
- Online diagnostics, 61
- ops command, 61
- ops display, 62
- /opt/config/opsrc file, 63
- /opt/config/options file, 17
- /opt/config/topology file, 12
- OPTIONS environment variable, 17
- Options files, 17

## P

- param file, 35
  - mbufs section, 36
  - network section, 35
- Parameter file, 35
  - mbufs section, 36
  - network section, 35
- Parameter files, 35
- Private GigaRing, definition, 8

## R

- Resiliency operations, 64
- Restoration, file system, 60
- Ring folding, 64
- Ring masking, 64
- route command, 46
- Routing within the cluster, 46
- Run levels, 23

## S

- Scheduling, 59
- Shared file systems, 31
- Shutdown, controlling, 50
- Shutting down, 26
  - Individual mainframes, 27
  - The SuperCluster, 27
- Single-user mode, 21
- Software packages, 60
- Startup, controlling, 48
- State status, 61
- stats command, 61
- statesrvd state server daemon, 61
- Status of the system, 61
- SuperCluster
  - Building block, definition of, 2
  - Connections
    - Inter-node GigaRing, 8

- Intra-node GigaRing, 8
- Private GigaRing, 8
- Direct connection configuration diagram, 6
- Matrix connection configuration diagram, 7
- SV1 series Cluster Bundle, 9
- SV1 series Cluster Bundle software,
  - configuring, 51
- SWS
  - Configuration, 11
  - Executing commands in several SWS
    - windows, 20
  - Hardware description, 3, 87
  - Operations, 19
- SWS operations, 19
- System configuration file examples, 67

## T

- TCP/IP network setup, 39
- Third-party software packages, 60
- TOPOLOGY environment variable, 12
- Topology file, 11
- Topology file rules, 13
- Topology file, example, 12

## U

- UDB, 53

- UDB cluster updating tool, 57
- UDB configuration file, 36
- UDB helper, 59
- UDB records
  - Creating, 55
  - Deleting, 56
  - Editing, 55
  - index file, 56
- UDB server, 59
- UDB source collection tool, 56
- UDB source editing tool, 55
- UDB source merging tool, 57
- UDB tools, cluster level, 54
- UDB tools, node level, 54
- udb\_collect command, 56
- udb\_edit command, 55
- udb\_helper script, 59
- udb\_merge command, 57
- udb\_server command, 59
- udb\_update command, 57
- udbgen command, 54
- udbsee command, 54
- UNICOS operating system, 8
- UNIX System V accounting, 60
- User database, 53
- Users, maintaining, 53