TruCluster Server

Cluster Technical Overview

Part Number: AA-RHGVE-TE

September 2002

Product Version: TruCluster Server Version 5.1B

Operating System and Version: Tru64 UNIX Version 5.1B

This document is intended for users and administrators of TruCluster Server who want a summary of its features. The manual describes the components and features of TruCluster Server Version 5.1B, including new and changed features since Version 5.1A.

Hewlett-Packard Company
Palo Alto, California

None of Compaq, HP, or any of their subsidiaries shall be liable for technical or editorial errors or omissions contained herein. The information is provided "as is" without warranty of any kind and is subject to change without notice. The warranties for HP or Compaq products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

Contents

About This Manual

1 Introduction to TruCluster Server

2 Clusterwide File Systems, Storage, and Device Names

3 Connection Manager

4 Highly Available Applications

5 Cluster Application Availability

6 Cluster Alias

7 Cluster Interconnect

8 Distributed Lock Manager

9 Cluster Installation and Administration

Glossary

Index

Figures

Tables

About This Manual

This manual provides a summary of the features that are available in the HP TruCluster Server Version 5.1B product.

The information in this manual does not supersede that found in the TruCluster Server *QuickSpecs*, which contains the authoritative description of this product. See the *QuickSpecs* for the latest product information.

Audience

This manual is for users and administrators who are interested in a descriptive overview of the TruCluster Server features and functions. Users and administrators should be familiar with the HP Tru64 UNIX operating system and TruCluster Server products.

New and Changed Features

The following changes have been made to this manual since the Version 5.1A release:

- Applicable information from the Version 5.1A *Cluster LAN Interconnect* manual has been integrated into this manual and the *Cluster Hardware Configuration*, *Cluster Installation*, and *Cluster Administration* manuals. The *Cluster LAN Interconnect* manual is no longer part of the TruCluster Server documentation set.
- *Section 1.1* has been updated to include the new features that are available in the TruCluster Server Version 5.1B product.
- *Chapter 5* has been updated to provide an overview of the cluster application availability (CAA) features that are new with this release, such as how CAA balances application resources and how application profiles can be extended with user-defined attributes.
- *Chapter 6* has been updated with a Note to reflect the relaxed `gated` restriction that allows a cluster member to run `gated`, run `routed` or use static routing.
- *Chapter 7* has been substantially rewritten to include additional information about selecting a cluster interconnect, previously published in the Version 5.1A *Cluster LAN Interconnect* manual.

- *Chapter 9* has been updated to include information about rolling multiple cluster members in parallel during a rolling upgrade from TruCluster Server Version 5.1A.

Organization

This manual is organized as follows:

Chapter 1 Provides an introduction to TruCluster Server features.

Chapter 2 Describes supported file systems, the cluster file system (CFS), context-dependent symbolic links (CDSLs), storage, and device-naming conventions.

Chapter 3 Introduces the connection manager and its role in forming and maintaining a cluster through quorum and votes.

Chapter 4 Defines the three basic types of highly available applications in a cluster: single-instance, multi-instance, and distributed.

Chapter 5 Provides an overview of the cluster application availability (CAA) subsystem, which provides clusterwide management for single-instance applications.

Chapter 6 Provides an overview of the cluster alias subsystem, which makes the cluster look like a single system to Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) applications.

Chapter 7 Provides an overview of LAN and Memory Channel cluster interconnects.

Chapter 8 Describes the distributed lock manager (DLM), which provides specialized functions that allow cooperating processes in a cluster to synchronize access to a shared resource.

Chapter 9 Provides an overview of cluster installation and administration, including rolling upgrades.

Glossary Defines common terms that are used throughout the TruCluster Server documentation.

Related Documents

The following documents and manuals provide detailed information about the TruCluster Server product:

- TruCluster Server *QuickSpecs* — The authoritative description of the TruCluster Server Version 5.1B product. You can find the latest version of the *QuickSpecs* at the following URL:

  `http://www.tru64unix.compaq.com/docs/pub_page/spds.html`

- *Cluster Release Notes* — Provides a brief introduction to new features in TruCluster Server and describes known problems and workarounds.

- *Cluster Hardware Configuration* — Describes how to set up the systems that will become cluster members, and how to configure cluster shared storage.
- *Cluster Installation* — Describes how to install the TruCluster Server software.
- *Cluster Highly Available Applications* — Describes how to deploy existing applications in a TruCluster Server cluster and how to write cluster-aware applications.
- *Cluster Administration* — Describes cluster-specific administration tasks.

You can find the latest versions of the TruCluster Server documentation at the following URL:

`http://www.tru64unix.compaq.com/docs/pub_page/cluster_list.html`

Icons on Tru64 UNIX Printed Manuals

The printed version of the Tru64 UNIX documentation uses letter icons on the spines of the manuals to help specific audiences quickly find the manuals that meet their needs. (You can order the printed documentation from HP.) The following list describes this convention:

G	Manuals for general users
S	Manuals for system and network administrators
P	Manuals for programmers
R	Manuals for reference page users

Some manuals in the documentation help meet the needs of several audiences. For example, the information in some system manuals is also used by programmers. Keep this in mind when searching for information on specific topics.

The *Documentation Overview* provides information on all of the manuals in the Tru64 UNIX documentation set.

Reader’s Comments

HP welcomes any comments and suggestions you have on this and other Tru64 UNIX manuals.

You can send your comments in the following ways:

- Fax: 603-884-0120 Attn: UBPG Publications, ZKO3-3/Y32
- Internet electronic mail: `readers_comment@zk3.dec.com`

A Reader’s Comment form is located on your system in the following location:

```
/usr/doc/readers_comment.txt
```

Please include the following information along with your comments:

- The full title of the manual and the order number. (The order number appears on the title page of printed and PDF versions of a manual.)
- The section numbers and page numbers of the information on which you are commenting.
- The version of Tru64 UNIX that you are using.
- If known, the type of processor that is running the Tru64 UNIX software.

The Tru64 UNIX Publications group cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate HP technical support office. Information provided with the software media explains how to send problem reports to HP.

Conventions

This manual uses the following conventions:

`#`
A number sign represents the superuser prompt.

`%` **`cat`**
Boldface type in interactive examples indicates typed user input.

`file`
Italic (slanted) type indicates variable values, placeholders, and function argument names.

`cat`(1)
A cross-reference to a reference page includes the appropriate section number in parentheses. For example, `cat`(1) indicates that you can find information on the `cat` command in Section 1 of the reference pages.

1

Introduction to TruCluster Server

TruCluster Server Version 5.1B is a highly integrated synthesis of the HP Tru64 UNIX operating system software, AlphaServer systems, and storage devices that operate as a single virtual system. Members of the cluster can share resources, data storage, and clusterwide file systems under a single security and management domain, yet they can boot or shut down independently without disrupting the cluster's services to clients.

A TruCluster Server environment can be as simple or as feature-rich as you require. You configure a cluster that fits your needs, from a two-node cluster up to an eight-node cluster running high availability applications such as transaction processing systems, servers for network client/server applications, data-sharing applications that require maximum uptime, and distributed parallel processing applications that take full advantage of the TruCluster Server application programming interfaces (APIs).

TruCluster Server includes a cluster alias for the Internet protocol suite (TCP/IP) so that a cluster appears as a single system to its network clients and peers.

This chapter provides the following information:

- A list of the new or changed features for this release (Section 1.1)
- A comprehensive table of TruCluster Server features (Section 1.2)

1.1 New or Changed Features for Version 5.1B

The following lists new or changed features for TruCluster Server Version 5.1B. Each item is accompanied by a pointer to a TruCluster Server manual, a section in this manual, or a reference page that provides information about that feature.

- Cluster file system (CFS) load balancing — The CFS load balancing daemon (`cfsd`) monitors and analyzes file system usage. You can configure `cfsd` to automatically relocate file systems when members join or leave the cluster, when storage connectivity changes, or as a result of CFS memory usage. See the *Cluster Administration* manual, `cfsd.conf`(4), and `cfsd`(8) for more information on CFS load balancing.
- Distributing file systems via the `mount -o` command — The `mount` command `server=`*`name`* option allows you to specify which cluster

member serves a given file system at startup. See the *Cluster Administration* manual, `mount`(2), and `mount`(8) for more information.

- Forced unmounting of an Advanced File System (AdvFS) file system or domain — TruCluster Server Version 5.1B includes the `cfsmgr -U` command that enables you to forcibly unmount an entire AdvFS domain that is being served by any cluster member. See the *Cluster Administration* manual and `cfsmgr`(8) for more information.
- CFS quiesce — Tru64 UNIX Version 5.1B includes the `freezefs` command, which causes an AdvFS domain to enter into a metadata-consistent frozen state. See the *Cluster Administration* manual, `freezefs`(8), and `thawfs`(8) for more information.
- Parallel rolling upgrade — This feature allows you to upgrade more than one cluster member at a time, thus shortening the time required to upgrade a cluster. See Section 9.1, the *Cluster Installation* manual, and `clu_upgrade`(8) for more information on parallel rolling upgrade.
- Relaxed `gated` requirement — Prior releases of TruCluster Server required that cluster members run the `gated` routing daemon. Version 5.1B relaxes that restriction to allow a cluster member to run `gated`, run `routed`, or use static routing. See the *Cluster Administration* manual for more information about the relaxed `gated` requirement.
- Balancing CAA application resources — CAA can reevaluate the placement of application resources at a regularly scheduled time or at any time that you decide to balance applications manually. See the *Cluster Highly Available Applications* manual for more information about this feature and the new `caa_balance` command.
- Creating CAA user-defined attributes — The format of application resource profiles can be extended with user-defined attributes. See the *Cluster Highly Available Applications* manual for more information.
- Directing output from CAA action scripts — You can redirect output from an action script so that it is displayed when CAA commands are executed. See the *Cluster Highly Available Applications* manual for more information.
- Availability measurement — Any user can specify and display availability measurements for application resources. See the *Cluster Administration* manual for more information about this feature and the new `caa_report` command.
- CAA monitored by `esmd` — The Essential Services Monitor daemon (`esmd`) automatically restarts the `caad` daemon if necessary. See the *Cluster Administration* manual for more information.

- Use of Internet Protocol Version 6 (IPv6) addresses — Cluster members can use and advertise IPv6 addresses. However, the following restrictions apply to using IPv6 addresses in a cluster:
  - An IPv6 address cannot be associated with a cluster alias.
  - An IPv6 address cannot be used for a cluster interconnect address; `clu_create` and `clu_add_member` do not accept IPv6 addresses for use as cluster interconnect addresses.

1.2 TruCluster Server Features

TruCluster Server Version 5.1B provides the features listed in Table 1–1.

Table 1–1: Features in the TruCluster Server Version 5.1B Product

Feature	Description
Clusterwide namespace	The cluster file system (CFS) supports a single clusterwide namespace and uniform coherent access to all file systems in a cluster. Context-dependent symbolic links (CDSLs) are used to maintain per-system configuration and data files within the shared CFS root (`/`), `/usr`, and `/var` file systems. See Section 2.2 for more information on CFS. See Section 2.5 for more information on CDSLs.
Clusterwide access to disk and tape storage	The device request dispatcher facility provides highly available clusterwide access to both character and block disk devices, as well as tape devices. All cluster disk and tape I/O passes through the device request dispatcher. See Section 2.3 for more information on the device request dispatcher.
Clusterwide Logical Storage Manager (LSM)	The semantics of LSM have been extended to a cluster environment. See Section 2.8 for more information on LSM in a cluster environment.
Connection manager	The connection manager ensures that all cluster members communicate with each other in order to control the formation and continued operation of a cluster. The connection manager calculates the votes required for quorum and decides when members are added to and removed from the cluster. See Chapter 3 for more information on the connection manager.

Table 1–1: Features in the TruCluster Server Version 5.1B Product (cont.)

Feature	Description
Cluster application availability (CAA)	The CAA facility provides resource monitoring and application restart capabilities. It provides the same type of application availability provided by user-defined services in the TruCluster Available Server Software and TruCluster Production Server Software products. See Chapter 4 for a definition of the types of applications that can run in a cluster. See Chapter 5 for more information on CAA's role in making single-instance applications highly available.
Cluster alias	The cluster alias subsystem lets TCP and UDP applications address the cluster as though it were a single system. When the cluster is created, a default cluster alias is defined that addresses all cluster members. A site can define additional aliases that address some or all cluster members. See Chapter 6 for more information on cluster aliases.
Highly available network file system (NFS) server using cluster alias	As shipped, the cluster is a highly available NFS server. CFS ensures that file systems exported from a TruCluster Server cluster are highly available to clients. By default, clients use the default cluster alias as the name of the NFS server when mounting file systems exported by the cluster. However, clients can also use an alias listed in the `/etc/exports.aliases` file. See Section 6.11 for more information.
Highly available Internet services using cluster alias	As shipped, the cluster supports many Internet services, such as `telnet` and `ftp` as highly available services. These services are designated as `in_multi` services in the `/etc/clua_services` file, and the cluster alias subsystem routes packets and requests addressed to these services to available cluster members. Nothing special was done to the services; the design of the cluster alias subsystem makes this possible.
multiple cluster interconnects	TruCluster Server Version 5.1B supports either Memory Channel or local area network (LAN) hardware as the cluster interconnect. The Memory Channel interconnect is a high-speed interconnect designed specifically for the needs of clusters. TruCluster Server provides a Memory Channel application programming interface (API) library, which is the same as that provided in the TruCluster Production Server Software product. Support for a LAN cluster interconnect provides an alternative to Memory Channel when configuring a new cluster or upgrading an existing TruCluster ASE cluster. See Chapter 7 for more information on LAN and Memory Channel interconnects. See the *Cluster Highly Available Applications* manual for a description of the Memory Channel API.

Table 1–1: Features in the TruCluster Server Version 5.1B Product (cont.)

Feature	Description
Distributed lock manager (DLM)	TruCluster Server supports the DLM and its API, which is the same as that provided in the TruCluster Production Server Software product. See Chapter 8 for a description of the DLM. See the *Cluster Highly Available Applications* manual for a description of the DLM API.
Single-system management	Because a cluster uses CFS, all member systems' configuration files are available for management. The SysMan suite of graphical management utilities provides an integrated view of the cluster environment, letting you manage a single member or the entire cluster. See Chapter 9 for an overview of cluster installation and administration.
Rolling upgrade	TruCluster Server Version 5.1B supports rolling upgrade. You can roll a cluster from TruCluster Server Version 5.1A to Version 5.1B. See Chapter 9 for more information.
File system partitioning	CFS makes it possible to mount an AdvFS file system so that it is accessible to only a single cluster member. This is referred to as **file system partitioning**. File system partitioning is provided in TruCluster Server to ease migration from TruCluster Production Server Software or TruCluster Available Server Software Version 1.5 or Version 1.6. File system partitioning is not intended as a general purpose method for restricting file system access to a single member. See the *Cluster Administration* manual and `mount`(8) for more information.
Single Security Domain	Because a cluster uses CFS, there is a single copy of security administration files such as `/etc/passwd` and `/etc/group`. A user who is authenticated on one member has access to all members. A user with access to a file on one member has access to that file from any member. Access control lists (ACLs) are uniformly available to all members. See the Tru64 UNIX *Security Administration* manual for more information.
Expanded process IDs (PIDs)	PIDs are expanded to a full 32-bit value. PIDs are unique across a cluster. Each cluster member has a block of numbers that it assigns as PIDs.

2

Clusterwide File Systems, Storage, and Device Names

From a configuration and administration point of view, perhaps the most important feature of TruCluster Server is the creation of a single, clusterwide namespace for files and directories. This namespace provides each cluster member with the same view of all file systems. In addition, there is a single copy of most configuration files. With few exceptions, the directory structure of a cluster is identical to that of a standalone system.

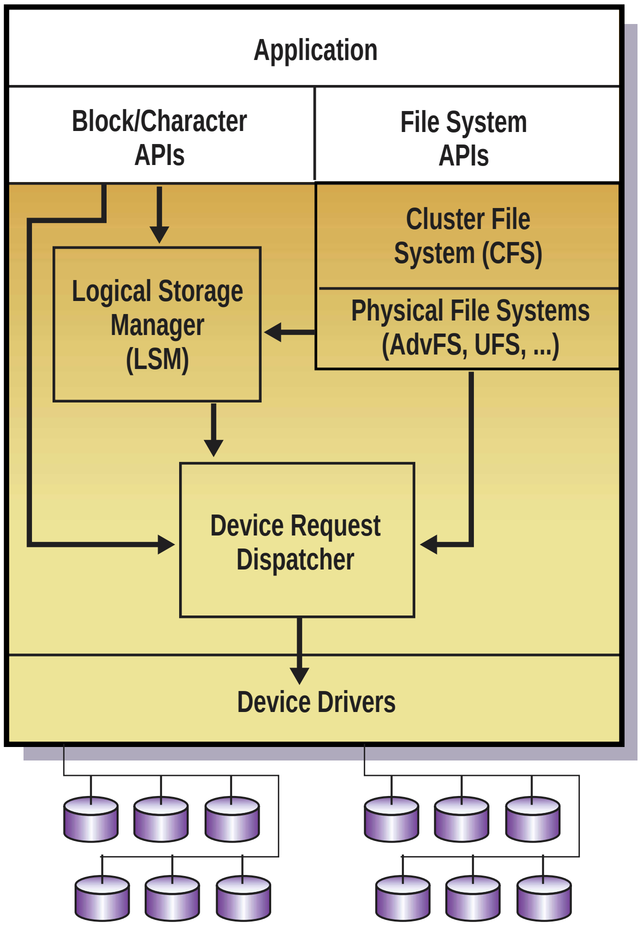
The clusterwide namespace is implemented by several TruCluster Server technologies, including the cluster file system (CFS) and the device request dispatcher, both of which are described in this chapter.

This chapter discusses the following topics:

- Supported file systems (Section 2.1)
- Cluster file system (CFS) (Section 2.2)
- Device request dispatcher (Section 2.3)
- CFS and device request dispatcher FAQ (Section 2.4)
- Context-dependent symbolic links (CDSLs) (Section 2.5)
- Device names (Section 2.6)
- Worldwide ID (Section 2.7)
- Clusters and the Logical Storage Manager (LSM) (Section 2.8)

To begin to understand how storage software works in a cluster, examine Figure 2–1. This figure shows a high-level view of storage software layering in a cluster. The device request dispatcher controls all I/O to physical devices; all cluster I/O passes through this subsystem. Also, CFS layers on top of existing file systems such as the Advanced File System (AdvFS).

Figure 2–1: Storage Software Layering in a Cluster


Application
Block/Character APIs
File System APIs
Cluster File System (CFS)
Logical Storage Manager (LSM)
Physical File Systems (AdvFS, UFS, ...)
Device Request Dispatcher
Device Drivers

ZK-1547U-AI

2.1 Supported File Systems

Table 2–1 summarizes supported file systems.

Table 2–1: File Systems Supported in a Cluster

Type	How Supported	Failure Characteristics
Advanced File System (AdvFS)	Read/write	A file domain is served by a member selected on the basis of its connectivity to the storage containing the file system. Upon member failure, CFS selects a new server for the domain. Upon path failure, CFS uses an alternate device request dispatcher path to the storage.
CD-ROM file system (CDFS)	Read-only	A CD-ROM device is served for read-only access by the member that is directly connected to the device. Because TruCluster Server does not support CD-ROM devices on a shared bus, a CD-ROM device becomes inaccessible to the cluster when the member to which it is locally connected fails, even if it is being served by another member. The device becomes accessible again when the member that failed rejoins the cluster.
DVD-ROM file system (DVDFS)	Read-only	A DVD-ROM device is served for read-only access by the member that is directly connected to the device. Because TruCluster Server does not support DVD-ROM devices on a shared bus, a DVD-ROM device becomes inaccessible to the cluster when the member to which it is locally connected fails, even if it is being served by another member. The device becomes accessible again when the member that failed rejoins the cluster.
File-on-File Mounting (FFM) file system	Read/write (local use)	Can be mounted and accessed only on the local member.
Memory file system (MFS)	Read/write (local use)	A cluster member can mount an MFS file system read-only or read/write. The file system is accessible only by that member. There is no remote access; there is no failover.
Named pipes	Read/write (local use)	Reader and writer must be on the same member.

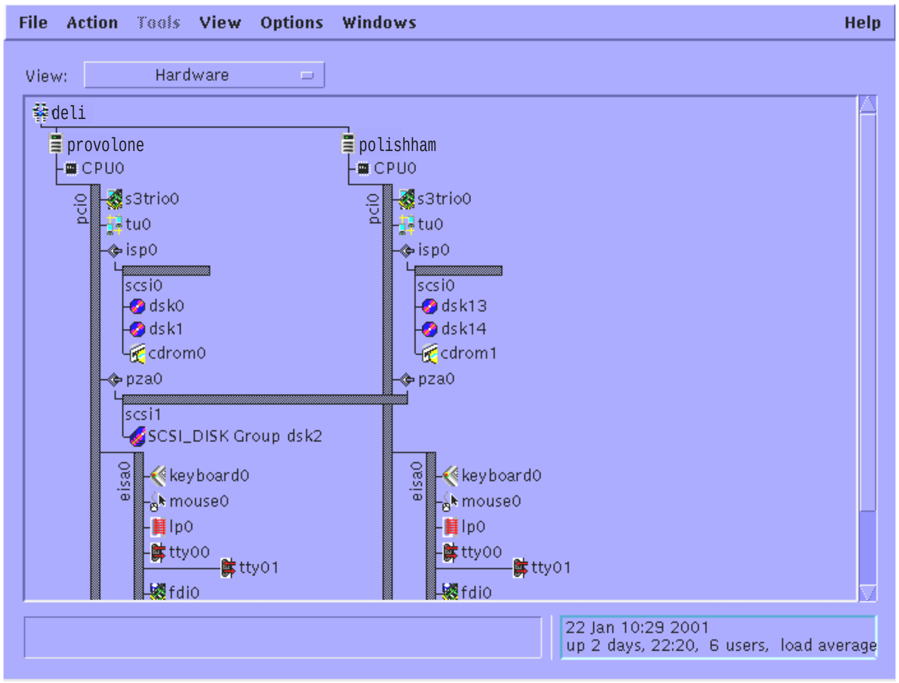
Table 2–1: File Systems Supported in a Cluster (cont.)

Type	How Supported	Failure Characteristics
Network file system (NFS) server	Read/write	External clients use the default cluster alias, or an alias listed in `/etc/exports.aliases`, as the host name when mounting file systems NFS-exported by the cluster. File system failover and recovery is transparent to external NFS clients.
NFS client	Read/write	A cluster member can mount an NFS file system whose server is outside the cluster. If the cluster member fails, the file system is automatically unmounted. If the cluster uses `automount` or `autofs`, the file system is remounted automatically; otherwise, the file system must be remounted manually.
PC-NFS server	Read/write	PC clients use the default cluster alias, or an alias listed in `/etc/exports.aliases`, as the host name when mounting file systems NFS-exported by the cluster. File system failover and recovery is transparent to external NFS clients.
`/proc` file system	Read/write (local use)	Each cluster member has its own `/proc` file system, which is accessible only by that member.
UNIX file system (UFS)	Read-only (clusterwide) Read/write (local use)	A UFS file system explicitly mounted read-only is served for clusterwide read-only access by a member selected for its connectivity to the storage containing the file system. Upon member failure, CFS selects a new server for the file system. Upon path failure, CFS uses an alternate device request dispatcher path to the storage. Read/write support is identical to MFS read/write support. A cluster member can mount a UFS file system read/write. The file system is accessible only by that member (only that member can read it, only that member can write it). There is no remote access; there is no failover.

If you know how to manage a Tru64 UNIX system, you already know how to manage a TruCluster Server cluster because TruCluster Server extends single-system management capabilities to clusters. It provides a clusterwide namespace for files and directories, including a single root (`/`) file system that all cluster members share. In a like manner, it provides a clusterwide namespace for storage devices; each storage device has the same unique device name throughout the cluster.

The SysMan suite of graphical management utilities provides an integrated view of the cluster environment, letting you manage a single member or the entire cluster. Figure 2–2 shows the SysMan Station hardware view for a cluster named `deli` with two members: `provolone` and `polishham`.

Figure 2–2: Cluster View of Hardware



View: Hardware

22 Jan 10:29 2001
up 2 days, 22:20, 6 users, load average

ZK-1700U-AI

TruCluster Server preserves the following availability and performance features of the TruCluster products provided for the Tru64 UNIX Version 4.0 series operating system:

- Like the TruCluster Available Server Software and TruCluster Production Server products, TruCluster Server lets you deploy highly available services that can access their disk data from any member in the cluster.

  Any application that can run on Tru64 UNIX can run as a highly available single-instance application in a cluster. The application is automatically relocated (failed over) to another cluster member in the event that a required resource, or the current member itself, becomes unavailable.

- Like the TruCluster Production Server Software product, TruCluster Server lets you run components of distributed applications in parallel, providing high availability while taking advantage of cluster-specific synchronization mechanisms and performance optimizations.

2.2 Cluster File System

The cluster file system (CFS) makes all files visible to and accessible by all cluster members. Each cluster member has the same view; it does not matter whether a file is stored on a device that is connected to all cluster members or on one that is private to a single member. By maintaining cache coherency across cluster members, CFS guarantees that all members at all times have the same view of file systems mounted in the cluster.

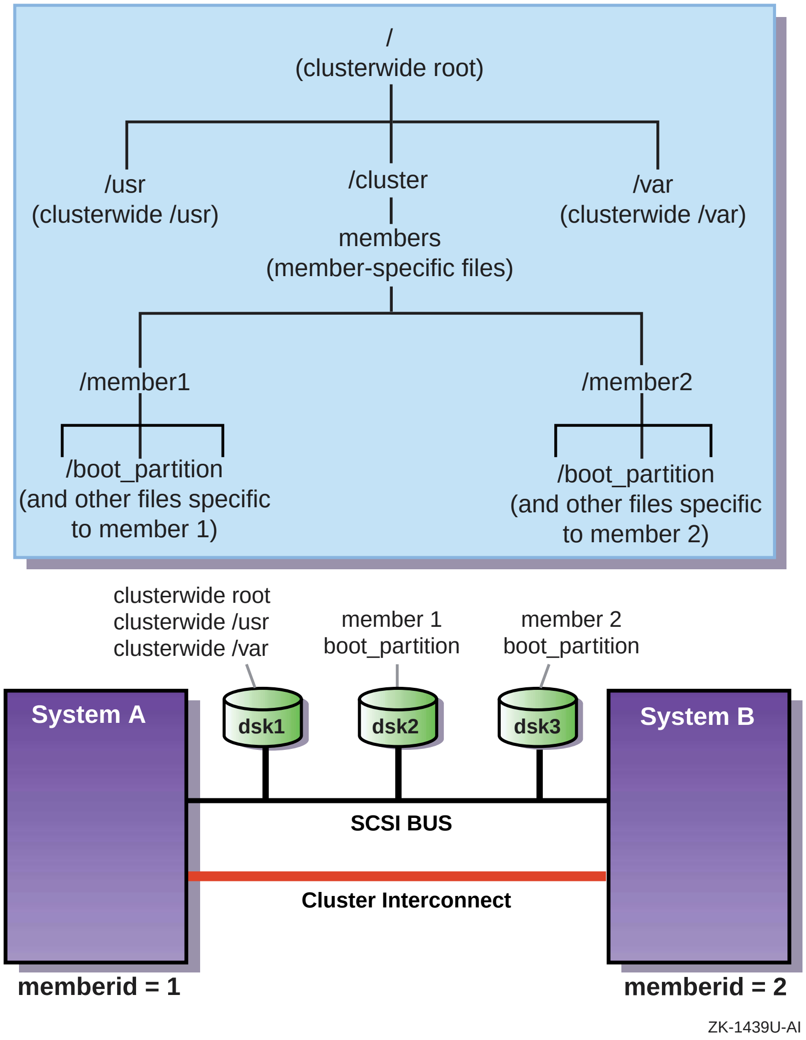
From the perspective of the CFS, each file system or AdvFS domain is served to the entire cluster by a single cluster member. Any cluster member can serve file systems on devices anywhere in the cluster. File systems mounted at cluster boot time are served by the first cluster member to have access to them. This means that file systems on devices on a bus private to one cluster member are served by that member.

This client/server model means that a cluster member can be a client for some domains and a server for others. In addition, you can transition a member between the client/server roles. For example, if you enter the `/usr/sbin/cfsmgr` command without options, it returns the names of domains and file systems, where each is mounted, the name of the server of each, and the server status. You can use this information to relocate file systems to other CFS servers, which balances the load across the cluster.

Because CFS preserves full X/Open and POSIX semantics for file-system access, file management interfaces and utilities work in the same way they do on a standalone system.

Figure 2–3 shows the relationship between file systems contained by disks on a shared bus and the resulting cluster directory structure. Each member boots from its own boot partition, but then mounts that file system at its `boot_partition` mount point in the clusterwide name space. This figure is only an example to show how each cluster member has the same view of file systems in a cluster. Many physical configurations are possible, and a real cluster provides additional storage to mirror the critical root (`/`), `/usr`, and `/var` file systems.

Figure 2–3: CFS Makes File Systems Available to All Cluster Members



CFS provides several performance enhancements:

- Direct I/O: When direct I/O is enabled for a file by opening the file with the O_DIRECTIO flag, read and write requests on it are executed to and from disk storage through direct memory access, bypassing AdvFS and CFS caching. This may improve I/O performance for database applications that do their own caching and file region synchronization. Remote CFS clients, as well as applications that are local to the CFS server, can read and write directly to file systems that are opened for direct I/O. Regardless of which member originates the I/O request, direct

I/O to a file does not go through the cluster interconnect to the CFS server.

- Direct-access cached reads: A performance enhancement for AdvFS file systems when reading files 64 KB or larger in size. Direct-access cached reads allow CFS to read directly from storage simultaneously from multiple cluster members. If the cluster member that issues the read request is directly connected to the storage containing the file system, direct-access cached reads access the storage directly and do not go through the cluster interconnect to the CFS server. This enhancement maintains the served file system model by having the server perform metadata and log updates, but offloads the cluster interconnect and the CFS server by performing file I/O directly to storage from CFS clients.

  Any application that performs read and writes to a file of 64 KB or larger in size uses direct-access cached reads when reading from that file. For example the following types of applications benefit from direct-access cached reads:

  – Multi-instance read mostly applications such as Web servers and proxy servers because they can perform simultaneous direct access reads from multiple cluster nodes.

  – Backup applications because, regardless of which node the application runs on, the file system contents do not pass through the cluster interconnect.

- Mounting file systems: When a mount command is issued, if the member on which the mount command is issued does not have connectivity, a member with connectivity to the underlying storage is chosen as the CFS server for the file system.

For more information, see the *Cluster Administration* manual.

2.3 Device Request Dispatcher

In a TruCluster Server cluster, the **device request dispatcher** subsystem controls all I/O to physical devices. All cluster I/O passes through this subsystem, which enforces single-system open semantics so only one program can open a device at any one time. The device request dispatcher makes physical disk and tape storage available to all cluster members, regardless of where the storage is physically located in the cluster. It uses the new device-naming model to make device names consistent throughout the cluster. This provides great flexibility when configuring hardware. A member does not need to be directly attached to the bus on which a disk resides to access storage on that disk.

When necessary, the device request dispatcher uses a client/server model. While CFS serves file systems and AdvFS serves domains, the device request dispatcher serves devices, such as disks, tapes, and CD-ROM drives. However, unlike the client/server model of CFS in which each file system or AdvFS domain is served to the entire cluster by a single cluster member, the device request dispatcher supports the use of many simultaneous servers.

In the device request dispatcher model, devices in a cluster are either single-served or direct-access I/O devices. A single-served device, such as a tape device, supports access from only a single member: the server of that device. A direct-access I/O device supports simultaneous access from multiple cluster members. Direct-access I/O devices on a shared bus are served by all cluster members on that bus.

You can use the `drdmgr` command to look at the device request dispatcher's view of a device. In the following example, device `dsk6` is on a shared bus, and is served by three cluster members.

```
# drdmgr dsk6

   View of Data from member polishham as of 2000-07-26:10:52:40

                   Device Name: dsk6
                   Device Type: Direct Access IO Disk
                 Device Status: OK
             Number of Servers: 3
                   Server Name: polishham
                  Server State: Server
                   Server Name: pepicelli
                  Server State: Server
                   Server Name: provolone
                  Server State: Server
            Access Member Name: polishham
           Open Partition Mask: 0x4 < c >
  Statistics for Client Member: polishham
     Number of Read Operations: 737
    Number of Write Operations: 643
          Number of Bytes Read: 21176320
       Number of Bytes Written: 6184960
```

The device request dispatcher supports clusterwide access to both character and block disk devices. You access a raw disk device partition in a TruCluster Server configuration in the same way you do on a Tru64 UNIX standalone system; that is, by using the device's special file name in the `/dev/rdisk` directory.

Note

Before TruCluster Server Version 5.0, cluster administrators had to define special Distributed Raw Disk (DRD) services to provide this level of physical access to storage. Starting with TruCluster Server Version 5.0, this access is built into the cluster architecture and is automatically available to all cluster members.

2.4 CFS and Device Request Dispatcher FAQ

This section answers frequently asked questions about CFS and the device request dispatcher in the following areas:

- CFS, I/O, and the cluster interconnect (Section 2.4.1)
- AdvFS requested block caching (Section 2.4.2)
- The device request dispatcher and file opens (Section 2.4.3)
- Relocating the CFS server (Section 2.4.4)

2.4.1 CFS, I/O, and the Cluster Interconnect

Question: On a shared bus with direct-access I/O disks, does I/O have to pass through the cluster interconnect?

Answer: For raw I/O, any node that is directly connected to a device has direct access via the device request dispatcher to a raw partition on that device. (The `drdmgr` command lists nodes that are servers for a device.)

Block I/O to directly connected storage, not a file system, goes through the CFS server for the device special file.

For generic file I/O writes, and reads of files less than 64 KB in size, the I/O passes through the CFS server for the file system. If the CFS client node is not the CFS server for the file system, the request is passed across the cluster interconnect to the node that is the CFS server for the file system, and then to the device request dispatcher on the CFS server node. The request never has to go to one node for the CFS server and then to another node for the device request dispatcher. (Asynchronous writes are written into memory and flushed to the server via write-behinds.)

For reads of files 64 KB or larger in size, CFS clients can read the files directly from storage using direct-access cached reads.

In addition, when a program opens a file with `O_DIRECTIO`, read and write requests are executed to and from disk storage through direct memory access, bypassing both AdvFS and CFS caching. Regardless of which member originates the I/O request, direct I/O to a file does not go across the cluster interconnect.

Section 2.2 has more detail on direct access cached reads and direct I/O. Also see `open`(2).

To summarize, I/O goes directly to storage in the following cases:

- Raw I/O to directly connected storage
- The CFS client is also the CFS server
- File I/O to a file opened with `O_DIRECTIO`

- Reads of files 64 KB or larger in size

2.4.2 AdvFS Requested Block Caching

Question: Are requested blocks of an AdvFS file system cached on the CFS client node?

Answer: Yes. CFS clients cache data and do write-behinds.

2.4.3 Device Request Dispatcher and File Opens

Question: When a program opens a file, at what point does the device request dispatcher become involved?

Answer: `open()` is CFS only; `read()` and `write()` involve CFS and the device request dispatcher. The device request dispatcher becomes involved on a `read()` when the cache CFS is reading needs filling, and on a `write()` when the cache that CFS is writing needs emptying.

2.4.4 Relocating the CFS Server

Question: When does it make sense to relocate the CFS server?

Answer: Look at output from the `cfsmgr` command to determine which members handle the most I/O. In general, the goal is to avoid having one node serving all file systems. (CFS uses a lot of memory; you can see a slowdown when all file systems are served by the same member.) The simplest approach is to monitor I/O for a while, decide the members that should be CFS servers for which file systems, and then write some simple boot scripts (for example, in `/sbin/init.d/`) that automatically relocate systems to the correct host.

For example, consider a two-member cluster (M1 and M2) and six file systems (A, B, C, D, E, F). After watching I/O, you decide that you want M1 to serve A, D, and E; and M2 to serve B, C, and F. You write a boot-time script that has M1 relocate A, D, and E to itself, and has M2 relocate B, C, and F to itself.

When balancing I/O among cluster members, balance at the CFS level rather than at the device request dispatcher level. In other words, use `cfsmgr` rather than `drdmgr` to balance I/O among cluster members.

2.5 Context-Dependent Symbolic Links

Although the single namespace greatly simplifies system management, you do not want some configuration files and directories to be shared by all cluster members. For example, a member's `/etc/sysconfigtab` file contains information about that system's kernel component configuration, and only that system should use that configuration. Consequently, the cluster must employ a mechanism that lets each member read and write the file named `/etc/sysconfigtab`, while actually reading and writing its own member-specific `sysconfigtab` file.

Tru64 UNIX Version 5.0 introduced a special form of symbolic link called a **context-dependent symbolic link (CDSL)**, which TruCluster Server uses to create a namespace with the following characteristics. CDSLs allow a file or directory to be accessed by a single name, regardless of whether the name represents a clusterwide file or directory, or a member-specific file or directory. CDSLs keep traditional naming conventions while providing a behind-the-scenes mechanism that makes sure each member reads and writes its own copy of member-specific system configuration files.

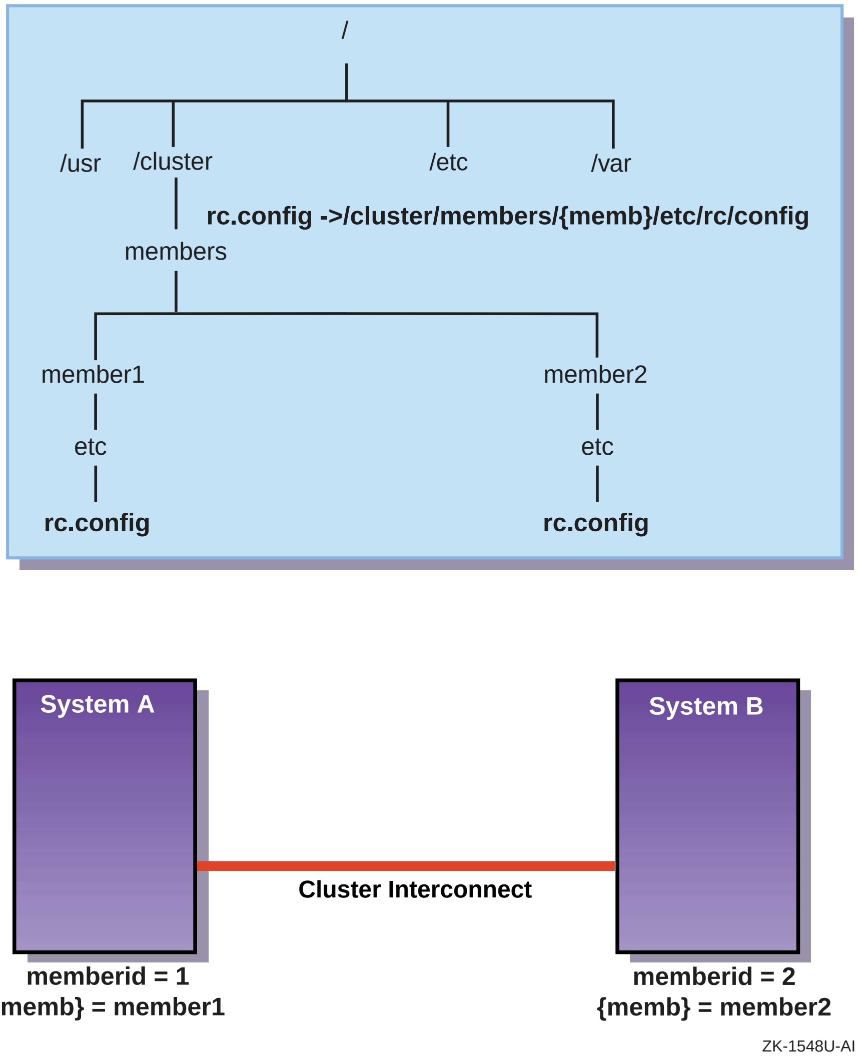
CDSLs contain a variable whose value is determined only during pathname resolution. The `{memb}` variable is used to access member-specific files in a cluster. The following example shows the CDSL for `/etc/rc.config`:

```
/etc/rc.config -> ../cluster/members/{memb}/etc/rc.config
```

When resolving a CDSL pathname, the kernel replaces the `{memb}` variable with the string `member`*`n`*, where *`n`* is the member ID of the current member. Therefore, on a cluster member whose member ID is 2, the pathname `/cluster/members/{memb}/etc/rc.config` resolves to `/cluster/members/member2/etc/rc.config`. Figure 2–4 shows the relationship between `{memb}` and CDSL pathname resolution.

CDSLs are useful when running multiple instances of an application on different cluster members when each member operates on a different set of data. The *Cluster Highly Available Applications* manual describes how applications can use CDSLs to maintain member-specific data sets and log files.

Figure 2–4: CDSL Pathname Resolution



As a general rule, before you move a file or directory, make sure that the destination is not a CDSL. Moving files to CDSLs requires special care on your part to ensure that the member-specific files are maintained. For example, consider the file `/etc/rc.config` as shown in the following example:

```
/etc/rc.config -> ../cluster/members/{memb}/etc/rc.config
```

If you move a file to `/etc/rc.config`, you replace the symbolic link with the actual file; `/etc/rc.config` will no longer be a symbolic link to `/cluster/members/{memb}/etc/rc.config`.

The mkcdsl command lets system administrators create CDSLs and update a CDSL inventory file. The cdslinvchk command verifies the current CDSL inventory. For more information on these commands, see mkcdsl(8) and cdslinvchk(8).

For more information about CDSLs, see the Tru64 UNIX *System Administration* manual, hier(5), ln(1), and symlink(2).

2.6 Device Names

This section provides an introduction to the device-naming model introduced in Tru64 UNIX Version 5.0. For a detailed discussion of this device-naming model, see the Tru64 UNIX *Hardware Management* manual.

Device names are consistent clusterwide:

- They are persistent beyond boot.
- A device name stays with the device even when you move a disk or tape to a new location in the cluster.

Prior to the release of Tru64 UNIX Version 5.0, disk device names encoded the I/O path for the disk. This path incorporated many pieces of data, and minimally included the following pieces of information: the device driver used to access the controller to which the disk is connected, the instance of the controller within the system that the driver manages, and a per-controller device unit ID.

For example, the rz device driver was used to access both SCSI and ATAPI/IDE device controllers. Disks connected to these controllers had names of the form rz*n*, where *n* identified both the controller to which the disk was connected and the unit ID. For example, a disk with SCSI ID=3 on the second SCSI/ATAPI/IDE controller was known as rz11. If that disk was moved to the third controller, it was accessed as rz19.

Tru64 UNIX Version 5.0 introduced a new device naming model in which the device name simply consists of a descriptive name for the device and an instance number. These two elements form the base name of the device, such as dsk0. The instance number in a device's new name does not correlate to the unit number in its old name: the operating system assigns the instance numbers in sequential order, beginning with 0 (zero), as it discovers devices. Additionally, most modern disks have IDs that can be used to uniquely identify the disk. For disks that support this feature, Tru64 UNIX Version 5.0 keeps track of this ID and uses it to build and maintain a table that maps disks to device names. As a result, moving one of these disks from one physical connection to another does not change the device name for the disk. This gives the system administrator greater flexibility when configuring disks in the system.

In a TruCluster environment, the flexibility provided by the new device naming model is particularly useful because each disk within the cluster has a unique name.

Note

> Although Tru64 UNIX supports old-style device names as a compatibility option, TruCluster Server supports only new-style device names. Applications that depend on old-style device names (or the structure of `/dev`) must be modified to use the new device-naming model.

Table 2–2 lists some examples of new device names.

Table 2–2: Examples of New Device Names

Old Name	New Name	Description
`/dev/rz4c`	`/dev/disk/dsk4c`	The `c` partition of the fifth disk recognized by the operating system.
`/dev/rz19c`	`/dev/disk/dsk5c`	The `c` partition of the sixth disk recognized by the operating system.

The suffix assigned to the device name special files differs depending on the type of device, as follows:

- Disks — In general, disk device file names consist of the base name and a one-letter suffix from `a` through `z`; for example, `/dev/disk/dsk0a`. Disks use `a` through `h` to identify partitions. By default, floppy disk and CD-ROM devices use only the letters `a` and `c`; for example, `floppy0a` and `cdrom1c`.

  For raw device names, the same device names are in the directory `/dev/rdisk`.

- Tapes — These device file names have the base name and a suffix composed of the characters `_d` followed by a single digit; for example, `tape0_d0`. This suffix indicates the density of the tape device, according to the entry for the device in the `/etc/ddr.dbase` file; for example:

Device	Density
`tape0`	default density
`tape0c`	default density with compression
`tape0_d0`	density associated with entry 0 in `/etc/ddr.dbase`
`tape0_d1`	density associated with entry 1 in `/etc/ddr.dbase`

With the new device special file naming for tapes, the old name suffix directly mapping to the new name suffix, as follows:

Old Suffix	New Suffix
l (low)	_d0
m (medium)	_d2
h (high)	_d1
a (alternative)	_d3

Tapes have two sets of device names; both conform to the new naming convention — the /dev/tape directory for rewind devices and the /dev/ntape directory for no-rewind devices. To determine which device special file to use, look in the /etc/ddr.dbase file.

Tru64 UNIX provides utilities to identify device names. For example, the following hwmgr commands display device and device hierarchy information in a cluster:

```
# hwmgr -view devices -cluster
# hwmgr -view hierarchy -cluster
```

You can use hwmgr to list a member's hardware configuration and correlate bus-target-LUN names with /dev/disk/dsk*n* names. For more information on the hwmgr command, see hwmgr(8).

Note

The Logical Storage Manager (LSM) naming conventions have not changed.

2.7 Worldwide ID

Tru64 UNIX associates the new device name with the **worldwide ID (WWID)** of a disk. A disk's WWID is unique; it is set by the manufacturers for devices that support WWID. No two disks can have the same WWID. Using the WWID to identify a disk has two implications. After a disk is recognized by the operating system, the disk's /dev/disk/dsk name stays the same even if its SCSI address changes.

This ability to recognize a disk lets Tru64 UNIX support multipathing to a disk where the disk is accessible through different SCSI adapters. If disks are moved within a TruCluster Server environment, their device names and how users access them remain the same.

Note

The names of disks behind RAID array controllers are associated with both the WWID of their controller module and their own bus, target, and LUN position. In this case, moving a disk changes its device name. However, you can use the `hwmgr` utility to reassociate such a disk with its previous device name.

The following `hwmgr` command displays the WWIDs for a cluster:

```
# hwmgr -get attr -a name -cluster
```

2.8 Clusters and the Logical Storage Manager

The Logical Storage Manager (LSM) provides shared access to all LSM volumes from any cluster member. LSM consists of physical disk devices, logical entities, and the mappings that connect both. LSM builds virtual disks, called volumes, on top of UNIX physical disks. LSM transparently places a volume between a physical disk and an application, which then operates on the volume rather than on the physical disk. For example, you can create a file system on an LSM volume rather than on a physical disk.

As previously shown in Figure 2–1, LSM is layered on top of the device request dispatcher. Using LSM in a cluster is like using LSM in a single system. The same LSM software subsets are used for both clusters and noncluster configurations, and you can make configuration changes from any cluster member. LSM keeps the configuration state consistent clusterwide.

The following list outlines LSM support for basic clusterwide file systems:

- Supported: root (`/`), `/usr`, `/var` file systems; and member swap partitions.
- Not supported: quorum disk and member boot disks.

See the Tru64 UNIX *Logical Storage Manager* manual for configuration and usage issues that are specific to LSM in a TruCluster Server environment. See the *Cluster Administration* manual for the differences between the Tru64 UNIX and TruCluster Server implementations of LSM.

3

Connection Manager

Clustered systems share various data and system resources, such as access to disks and files. To achieve the coordination that is necessary to maintain resource integrity, the cluster must have clear criteria for membership and must disallow participation in the cluster by systems that do not meet those criteria.

The connection manager is a distributed kernel component that monitors whether cluster members can communicate with each other, and enforces the rules of cluster membership. The connection manager:

- Forms a cluster, adds members to a cluster, and removes members from a cluster
- Tracks which members in a cluster are active
- Maintains a cluster membership list that is consistent on all cluster members
- Provides timely notification of membership changes using Event Manager (EVM) events
- Detects and handles possible cluster partitions

An instance of the connection manager runs on each cluster member. These instances maintain contact with each other, sharing information such as the cluster's membership list. The connection manager uses a three-phase commit protocol to ensure that all members have a consistent view of the cluster.

This chapter provides the following information:

- A discussion of quorum, votes, and cluster membership (Section 3.1)
- A discussion of how the connection manager calculates quorum (Section 3.2)
- When and how to use a quorum disk (Section 3.3)

3.1 Quorum and Votes

The connection manager ensures data integrity in the face of communication failures by using a voting mechanism. It allows processing and I/O to occur in a cluster only when a majority of **votes** are present. When the majority of votes are present, the cluster is said to have **quorum**.

The mechanism by which the connection manager calculates quorum and allows systems to become and remain cluster members depends on a number of factors, including expected votes, current votes, node votes, and quorum disk votes. This section describes these concepts.

3.1.1 What Is a Cluster Member?

The connection manager is the sole arbiter of cluster membership. A node that has been configured to become a cluster member, either through the `clu_create` or `clu_add_member` command does not become a cluster member until it has rebooted with a clusterized kernel and is allowed to form or join a cluster by the connection manager. The difference between a cluster member and a node configured to become a cluster member is important in any discussion of quorum and votes.

After a node has formed or joined a cluster, the connection manager forever considers it to be a cluster member (until someone uses `clu_delete_member` to remove it from the cluster). A disruption of communications in a cluster (such as that caused by broken or disconnected hardware) might cause an existing cluster to divide into two or more clusters. If the cluster divides, known as a cluster partition, nodes may consider themselves to be members of one cluster or another. However, as discussed in Section 3.2, the connection manager at most allows only one of these clusters to function.

3.1.2 Node Votes

Node votes are the fixed number of votes that a given member contributes towards quorum. Cluster members can have either 1 or 0 (zero) node votes. Each member with a vote is considered to be a **voting member** of the cluster. A member with 0 (zero) votes is considered to be a **nonvoting member**.

Voting members can form a cluster. Nonvoting members can only join an existing cluster.

A member's votes are initially determined by the `cluster_node_votes` kernel attribute in the `clubase` subsystem of its member-specific `etc/sysconfigtab` file.

3.1.3 Quorum Disk Votes

In certain cluster configurations, described in Section 3.3, you can enhance cluster availability by configuring a **quorum disk**. **Quorum disk votes** are the fixed number of votes that a quorum disk contributes towards quorum. A quorum disk can have either 1 or 0 (zero) votes.

Quorum disk votes are initialized from the `cluster_qdisk_votes` kernel attribute in the `clubase` subsystem of each member's `etc/sysconfigtab` file.

When configured, a quorum disk's vote plays a unique role in cluster formation because of the following rules, which are enforced by the connection manager:

- A booting node cannot form a cluster unless it has quorum.
- Before the booting node can claim the quorum disk and its vote, it must be a cluster member.

In the situation where the booting node needs the quorum disk vote to achieve quorum, these rules create an impasse: the booting node can never form a cluster.

The connection manager resolves this dilemma by allowing booting members to provisionally apply the quorum disk vote towards quorum. This allows a booting member to achieve quorum and form the cluster. After it has formed the cluster, it claims the quorum disk. At that point, the quorum disk's vote is no longer provisional; it is real.

3.1.4 Expected Votes

Expected votes are the number of votes the connection manager expects when all configured votes are available. In other words, expected votes are the sum of all node votes that are configured in the cluster, plus the vote of the quorum disk, if one is configured. Each member brings its own notion of expected votes to the cluster; all members must agree on the same number of expected votes.

The connection manager refers to the node expected votes settings of booting cluster members to establish its own internal clusterwide notion of expected votes, referred to as **cluster expected votes**. The connection manager uses its cluster expected votes value when determining the number of votes that the cluster requires to maintain quorum, as explained in Section 3.2.

Use the `clu_quorum` command or the `clu_get_info -full` command to display the current value of cluster expected votes.

The `clu_create` and `clu_add_member` scripts automatically adjust each member's expected votes as a new voting member or quorum disk is configured in the cluster. The `clu_delete_member` command automatically lowers expected votes when a member is deleted. Similarly, the `clu_quorum` command adjusts each member's expected votes as a quorum disk is added or deleted, or node votes are assigned to or removed from a member. These commands ensure that the member-specific expected votes value is the same on each cluster member, and that it is the sum of all node votes and the quorum disk vote, if a quorum disk is configured.

A member's expected votes are initialized by the cluster_expected_votes kernel attribute in the clubase subsystem of its member-specific etc/sysconfigtab file. Use the clu_quorum command to display a member's expected votes.

3.1.5 Current Votes

Current votes are the actual number of votes that are visible within the cluster. If expected votes are the number of configured votes in a cluster, current votes are the number of votes contributed by current members and any configured quorum disk that is on line.

3.2 Calculating Cluster Quorum

The quorum algorithm is the method by which the connection manager determines the circumstances under which a given member can participate in a cluster, safely access clusterwide resources, and perform useful work. The algorithm operates dynamically: that is, cluster events trigger its calculations, and the results of its calculations can change over the lifetime of a cluster. This section describes how the connection manager's quorum algorithm works.

The quorum algorithm operates as follows:

1. The connection manager selects a set of cluster members upon which it bases its calculations. This set includes all members with which it can communicate. For example, it does not include configured nodes that have not yet booted, members that are down, or members that it cannot reach due to a hardware failure (for example, a detached cluster interconnect cable or a bad cluster interconnect adapter).
2. When a cluster is formed, and each time a node boots and joins the cluster, the connection manager calculates a value for cluster expected votes using the largest of the following values:
   - The maximum member-specific expected votes value from the set of proposed members that were selected in step 1.
   - The sum of the node votes from the set of proposed members selected in step 1, plus the quorum disk vote if a quorum disk is configured.
   - The previous cluster expected votes value.

   Consider a three-member cluster with no quorum disk. Each member has one vote and has its member-specific expected votes set to 3. The value of cluster expected votes is currently 3.

A fourth member is then added to the cluster. When the new member boots, the connection manager calculates the new cluster expected votes as 4, which is the sum of node votes in the cluster.

Use the `clu_quorum` or `clu_get_info -full` command to display the current value of cluster expected votes.

3. Whenever the connection manager recalculates cluster expected votes (or resets cluster expected votes as the result of a `clu_quorum -e` command), it calculates a value for quorum votes.

   Quorum votes is a dynamically calculated clusterwide value, based on the value of cluster expected votes, that determines whether a given node can form, join, or continue to participate in a cluster. The connection manager computes the clusterwide quorum votes value using the following formula:

   ```
   quorum votes = round_down((cluster_expected_votes+2)/2)
   ```

   For example, consider the three-member cluster described in the previous step. With cluster expected votes set to 3, quorum votes are calculated as round_down((3+2)/2), or 2. In the case where the fourth member was added suc cessfully, quorum votes are calculated as round_down((4+2)/2), or 3.

   Note

   Expected votes (and, hence, quorum votes) are based on cluster configuration, rather than on which nodes are up or down. When a member is shut down, or goes down for any other reason, the connection manager does not decrease the value of quorum votes. Only member deletion and the `clu_quorum -e` command can lower the quorum votes value of a running cluster.

4. Whenever a cluster member determines that the number of votes it can see has changed (when a node joins the cluster, an existing member is deleted from the cluster, or a communications error is reported), it compares current votes to quorum votes.

   The action the member takes is based on the following conditions:

   - If the value of current votes is greater than or equal to quorum votes, the member continues running or resumes (if it had been in a suspended state).
   - If the value of current votes is less than quorum votes, all of its I/O is suspended and all network interfaces except the cluster interconnect interfaces are turned off. No commands that access a

clusterwide resource will work on that member. The member might appear to be hung.

This state is maintained until sufficient votes are added (that is, until enough members join the cluster or the communications problem is mended) to bring current votes to a value greater than or equal to quorum votes.

The comparison of current votes to quorum votes occurs on a member-by-member basis, although events may make it appear that quorum loss is a clusterwide event.

Depending upon how the member lost quorum, you might be able to remedy the situation by booting a member with enough votes for the member in quorum hang to achieve quorum. If all cluster members have lost quorum, your options are limited to booting a new member with enough votes for the members in quorum hang to achieve quorum, rebooting the entire cluster, or using the troubleshooting procedures discussed in the *Cluster Administration* manual.

3.3 Using a Quorum Disk

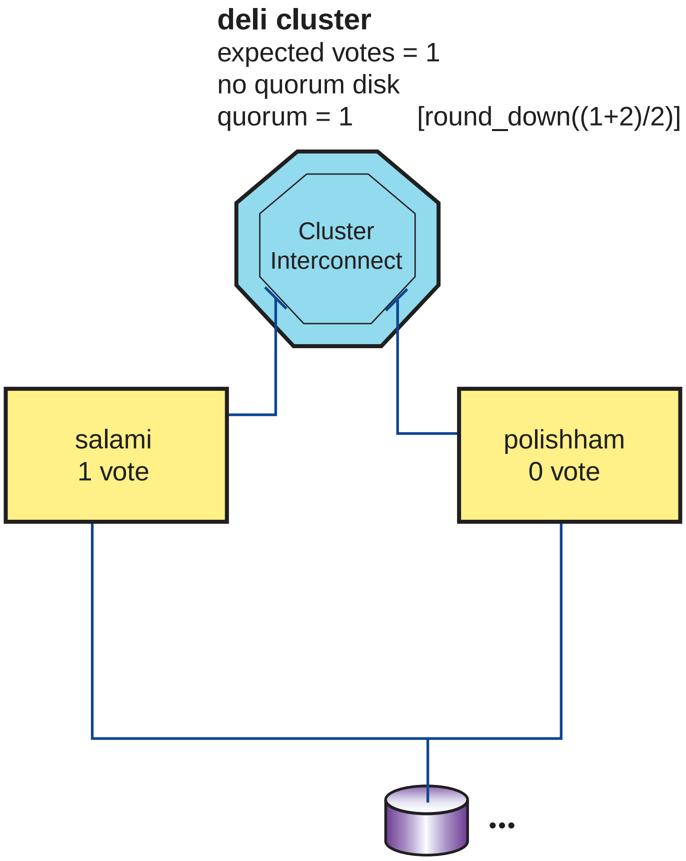
In a two-member cluster configuration, where each member has one member vote and expected votes has the value of 2, the loss of a single member will cause the cluster to lose quorum and all applications to be suspended. This type of configuration is not highly available.

A more realistic (but not substantially better) two-member configuration assigns one member 1 vote and the second member 0 (zero) votes. Expected votes is 1. This cluster can lose its second member (the one with no votes) and remain up. However, it cannot afford to lose the first member (the voting one).

To foster better availability in such a configuration, you can designate a disk on a shared bus as a quorum disk. The quorum disk acts as a virtual cluster member whose purpose is to add one vote to the total number of expected votes. When a quorum disk is configured in a two-member cluster, the cluster can survive the failure of either the quorum disk or one member and continue operating.

For example, consider the two-member `deli` cluster without a quorum disk as shown in Figure 3–1.

Figure 3–1: Two-Member deli Cluster Without a Quorum Disk



ZK-1569U-AI

One member contributes 1 node vote and the other contributes 0 (zero), so cluster expected votes is 1. The connection manager calculates quorum votes as follows:

```
quorum votes =
round_down((cluster_expected_votes+2)/2) =
round_down((1+2)/2) = 1
```

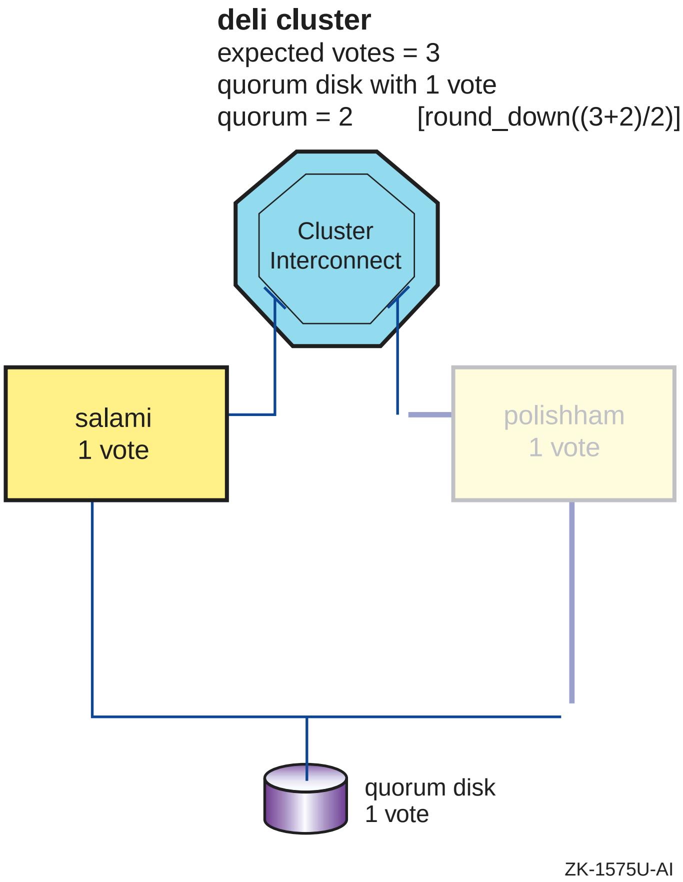
The failure or shutdown of member `salami` causes member `polishham` to lose quorum. Cluster operations are suspended.

However, if the cluster includes a quorum disk (adding 1 vote to the total of cluster expected votes), and member `polishham` is also given a vote, expected votes becomes 3 and quorum votes becomes 2:

```
quorum votes =
round_down((cluster_expected_votes+2)/2) =
round_down((3+2)/2) = 2
```

Now, if either member or the quorum disk leaves the cluster, sufficient current votes remain to keep the cluster from losing quorum. The cluster in Figure 3–2 can continue operation.

Figure 3–2: Two-Member deli Cluster with Quorum Disk Survives Member Loss



The `clu_create` utility allows you to specify a quorum disk at cluster creation and assign it a vote. You can also use the `clu_quorum` utility to add a quorum disk later; for example, when the result of a `clu_delete_member` is a two-member cluster with compromised availability.

To configure a quorum disk, use the `clu_quorum -d add` command. For example, the following command defines `/dev/disk/dsk11` as a quorum disk with one vote:

```
# clu_quorum -d add dsk11 1
Collecting quorum data for Member(s): 1 2
```

```
  Initializing cnx partition on quorum disk : dsk11h

  Successful quorum disk creation
# clu_quorum
Collecting quorum data for Member(s): 1 2

Quorum Data for Cluster: deli as of Thu  Mar  9 09:59:18 EDT 2000

Cluster Common Quorum Data
Quorum disk: dsk11h
        .
        .
        .
```

The following restrictions apply to the use of a quorum disk:

- A cluster can have only one quorum disk.
- Put the quorum disk on a shared bus to which all cluster members are directly connected. If it is not, members that do not have a direct connection to the quorum disk may lose quorum before members that do have a direct connection to it.
- The quorum disk must not contain any data. The `clu_quorum` command overwrites existing data when initializing the quorum disk. The integrity of data (or file system metadata) placed on the quorum disk from a running cluster is not guaranteed across member failures.

  Do not use member boot disks and the disk holding the clusterwide root (`/`) as quorum disks.
- The quorum disk can be quite small. The cluster subsystems use only 1 MB of the disk.
- A quorum disk can have either 1 vote or 0 (zero) votes. In general, always assign a vote to the quorum disk. You might assign an existing quorum disk no votes in certain testing or transitory configurations, such as a one-member cluster (in which a voting quorum disk introduces a second point of failure).
- You cannot use the Logical Storage Manager (LSM) on the quorum disk.

4

Highly Available Applications

Applications on clusters can be divided into three basic types:

single-instance application

A single-instance application runs on only one cluster member at a time. To make this type of application highly available, the cluster must provide a mechanism for starting the application on another cluster member in the event that the current member can no longer run the application. The TruCluster Server high availability mechanism for single-instance applications is the cluster application availability (CAA) subsystem; see Chapter 5 for a description of CAA.

The *Cluster Highly Available Applications* manual provides detailed information about moving applications from the TruCluster Software Version 1.*x* series of products to TruCluster Server Version 5.1B.

multi-instance application

A multi-instance application can run on multiple cluster members at the same time. A multi-instance application by definition is **highly available** because the failure of one cluster member does not affect the instances of the application running on other members. See Chapter 6 for a discussion of how cluster aliases provide transparent client access to applications.

distributed application

A distributed application is specifically designed to run on a cluster, using different members for specific purposes. These applications use the Memory Channel, distributed lock manager (DLM), and cluster alias application programming interfaces (APIs) to integrate applications with cluster resources.

TruCluster Server lets you run components of distributed applications in parallel, providing high availability while taking advantage of cluster-specific synchronization mechanisms and performance optimizations.

See Chapter 6, Section 7.6, Chapter 8, and the *Cluster Highly Available Applications* manual for more information on the subsystems and interfaces that you can use to create distributed applications.

Note

Section 6.7 discusses the `in_single` and the `in_multi` service attributes, which determine whether the cluster alias subsystem arbitrarily selects one member to receive all packets for a service that is reached through an alias or distributes packets among all eligible members of an alias. Although these terms are similar in form to single-instance and multi-instance, the `in_single` and `in_multi` attributes affect how the cluster alias subsystem routes packets and connection requests that are addressed to a service's port. Do not confuse these routing attributes with the generic terms that describe how many instances of an application can run in the cluster. The *Cluster Administration* manual has a section in its Cluster Alias Subsystem chapter that describes the differences between the cluster alias subsystem and the cluster application availability subsystem.

5

Cluster Application Availability

This chapter provides the following information:

- A general overview of the cluster application availability (CAA) subsystem (Section 5.1)
- A discussion of the CAA architecture (Section 5.2)
- An introduction to CAA resources (Section 5.3)
- A description of resource profiles and their use (Section 5.4)
- A description of the action scripts used by CAA commands to manage applications and other resources (Section 5.5)

5.1 Overview of CAA

The cluster application availability (CAA) subsystem provides high availability for single-instance applications and the capability to monitor applications and the state of other types of resources, such as network interfaces, tape devices, and media changer devices. (A single-instance application runs on a single member of a cluster, and cannot be run on more than one member at a time.) A single instance of any application that can run on Tru64 UNIX can be made highly available in a cluster with CAA. For example, in a cluster, the daemons for BIND (`named`), DHCP (`joind`), and network locking (`rpc.lockd` and `rpc.statd`) are managed by CAA.

Each application under CAA control has a resource profile, which describes that application's resource requirements and the circumstances under which it can be relocated to another cluster member. CAA monitors the state of cluster members and resources to ensure that each application runs on a member that meets its resource requirements. Resource profiles can be created and managed through either a command-line interface or a graphical user interface (GUI).

CAA can automatically relocate an application to another cluster member if a required resource, or the current member itself, becomes unavailable. This feature requires no changes to the application itself, and can be used with any single-instance application. CAA also monitors resources so that it can restart applications resources that have gone off line due to a resource failure.

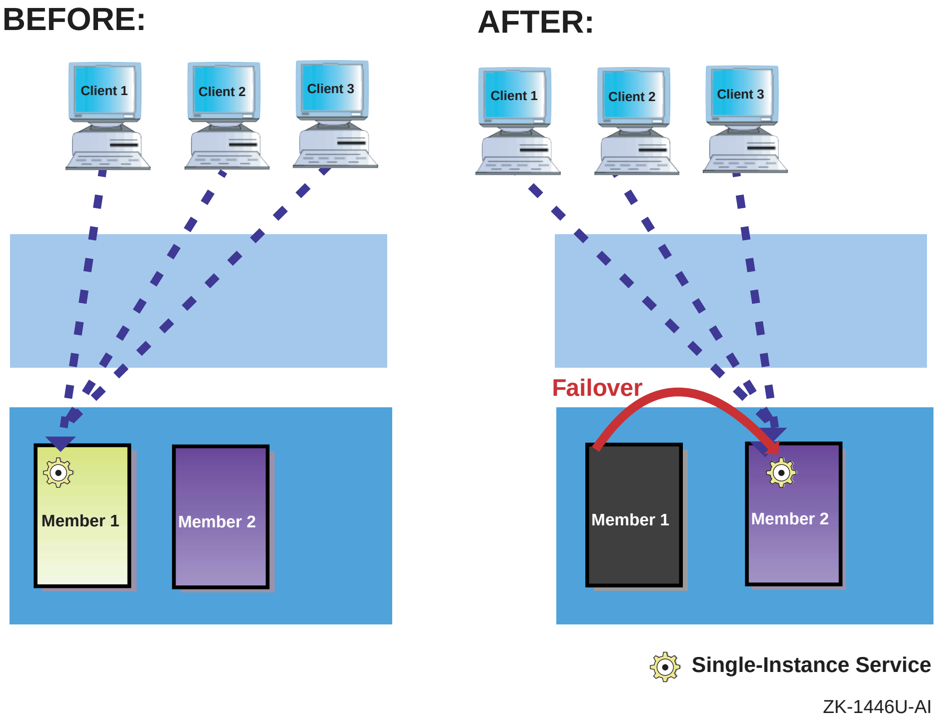
CAA can also reevaluate the placement of any application resource at a regularly scheduled time or at any time that the administrator decides to balance applications manually. Balancing decisions are made using the standard placement decision mechanism of CAA and is not based on any load considerations. Any applications that are not placed optimally are relocated to the most favored cluster member.

Note

CAA's resource monitoring and application restart capabilities are enhancements to the type of application availability provided by available server environment (ASE) for user-defined services in previous TruCluster products.

Figure 5–1 shows how the failure of one member results in the failover of an application to the second member. If clients access the application through a cluster alias, the cluster alias subsystem automatically forwards connection requests to the second member.

Figure 5–1: Application Failover with CAA



5.2 CAA Architecture

The CAA subsystem consists of the following components:

resource

A resource is a cluster software or hardware component that provides a service to end users or to other software components. Resources are the building blocks that CAA uses to make services highly available to clients. CAA supports the following types of resources: applications, network interfaces, tape drives, and media changers.

resource manager

The resource manager communicates with all the components of the CAA subsystem, as well as the connection manager and the Event Manager (EVM).

The resource manager consists of all the CAA daemons running on cluster members. Each CAA daemon (`caad`) starts, stops, relocates, and restarts application resources when a required resource, the application itself, or a cluster member fails. Each cluster member runs a CAA daemon. These daemons are independent but they communicate with each other, sharing information about the status of the resources. If any `caad` daemon fails, the Essential Services Monitor daemon `esmd` restarts the `caad` daemon and management of resources can continue.

The resource manager also uses the resource monitors that monitor the status of a particular type of resource.

resource monitor

A resource monitor is a shared library located in `/var/cluster/caa/monitors`, which is loaded by the resource manager, `caad`, at boot time. There is one resource monitor for each type of resource (application, network, tape, and media changer).

resource profile

Resource profiles contain the information needed by the resource manager and monitors to control application relocation and monitor resources.

A resource profile contains keyword/value pairs that define a resource, its dependencies (for application resources), and how the resource is managed by CAA. After the resource is registered with `caa_register`, the resource manager can use the resource profile.

The `caa_profile` command and SysMan can create resource profiles, or they can be created in any text editor. (Use `caa_profile -validate` to ensure the correct syntax of profiles that are created or modified using a text editor.) Errors other than syntactical errors are detected at the time of registration. This two-stage validation allows for profiles to be created with dependencies on resources that are currently off line or yet to be created.

Resource profiles are located in the `/var/cluster/caa/profile` directory. The file names of resource profiles take the form *`resource_name`*`.cap`.

action script

An action script is a set of commands that are used by CAA to start, stop, and check an application. The name of an application's action script is defined in that application's resource profile .

You can create or update an action script using the command-line interface, SysMan, or a text editor.

Action scripts can make use of variables that are available from CAA. The values of all profile attributes are available to an action script. The reason code variable tells why an action script is being executed. Many locale variables for the environment in which the script is being executed are available to an action script. User-defined attributes are also available to an action script.

You can optionally allow standard output of action scripts to be forwarded to the standard output of the command that invokes the action script. This is not turned on by default.

Action scripts are located in the `/var/cluster/caa/script` directory. The file names of action scripts take the form *`resource_name`*`.scr`.

command-line interface

The CAA subsystem provides the `caa_profile`, `caa_register`, `caa_unregister`, `caa_start`, `caa_stop`, `caa_relocate`, `caa_balance`, `caa_report`, and `caa_stat` commands to manage and monitor resources. See `caa`(4) for a list of all CAA reference pages.

The command-line interface interacts with resource profiles, action scripts, and the resource manager.

graphical user interface

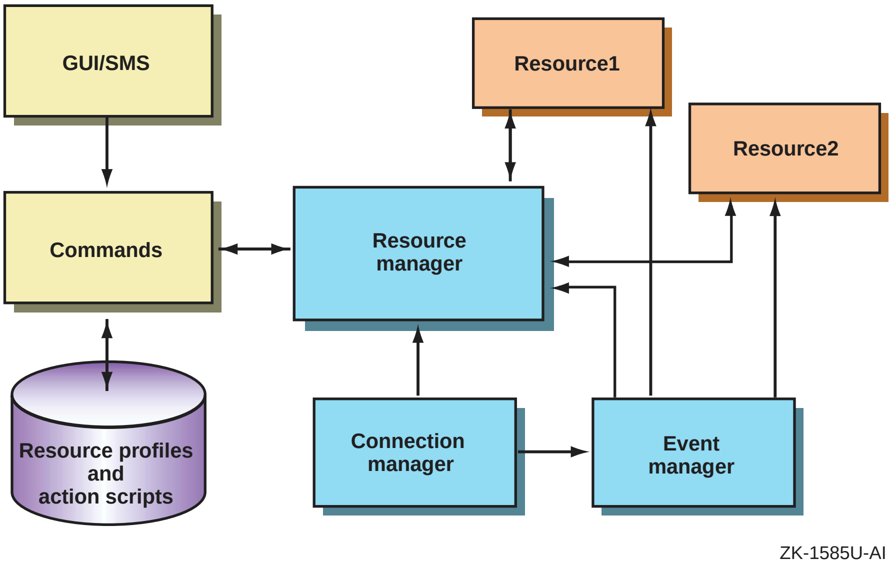
SysMan Menu and SysMan Station provide graphical user interfaces (GUIs) to perform system management tasks for the cluster, cluster members, and CAA applications. For more information on using the GUIs for performing system management tasks for CAA applications, see `sysman`(8) and the online help for the SysMan Menu and SysMan Station.

The CAA GUI calls the command-line interface to interact with resource profiles, action scripts, and the resource manager.

Although the connection manager and Event Manager are not part of the CAA subsystem, the subsystem makes extensive use of these facilities.

Figure 5–2 shows a graphical representation of the CAA architecture.

Figure 5–2: CAA Architecture



5.3 Resources

A **resource** is a cluster software or hardware component that provides a service to end users or to other software components. Resources are the building blocks that CAA uses to make services highly available to clients. CAA supports the following types of resources:

`application`

An executable program. An application resource can have dependencies on other resources, including another application resource. In the resource profile that defines an application resource, these dependencies are defined as either required, `REQUIRED_RESOURCES`, or optional, `OPTIONAL_RESOURCES`.

If you define a resource as a required resource and the required resource becomes unavailable, CAA stops the application. CAA then attempts to restart the application on another member that has the required resource. If CAA cannot restart the application on another member because the other member is down or because the placement policy forbids starting the application on that member, the application is stopped. CAA does not restart the application until all required resources are available.

You can use optional resources in conjunction with required resources and the placement policy to help determine the optimal system on which to start an application. If an optional resource becomes unavailable the application does not fail over.

`network`

A network interface. All cluster members can indirectly access any network attached to any member. An application that makes extensive use of a network connection available on another cluster member can add traffic to the cluster interconnect, and slow down performance of both the application and the cluster. Defining a network resource as a required resource for an application is useful when you want an application to run on a member with direct connectivity to a specific network.

If you define a network resource as a required resource for an application and the network interface adapter fails, CAA relocates or stops the application if it cannot relocate the resource.

If you define a network resource as an optional resource for an application, CAA starts the application on a member that is directly connected to the network. If the subnet adapter fails, the application reverts to accessing the network indirectly.

`tape` or `changer`

A tape drive or media changer. If you define a tape or media changer resource as a required resource for an application, the application always runs on a cluster member with direct connectivity to the tape device or changer. If the device fails, CAA attempts to relocate the application, or stops the application if relocation is not possible.

If you define a tape or media changer resource as an optional resource for an application, CAA attempts to start the application on a member with direct connectivity, but it also runs the application on a member that does not have direct connectivity to the device. Running on a member with direct connectivity to a tape device is desirable to maximize performance.

5.4 Resource Profiles

Each resource has a resource profile, which defines the resource, lists any dependencies, and provides instructions for how CAA should manage the resource. A resource profile is a simple text file containing a list of keyword/value pairs, which are described in `caa`(4). By default, all resource profiles are located in the `/var/cluster/caa/profile` directory.

A resource profile must be registered through the `caa_register` command in order for CAA to monitor and manage the resource.

The following sections describe the two types of resource profiles:

- Application resource profiles (Section 5.4.1)
- Nonapplication resource profiles (Section 5.4.2)

5.4.1 Application Resource Profiles

For an application resource, a resource profile can contain the application's type, name, check interval, monitoring thresholds, resource dependencies (required resources), optional resources, hosting member list, placement policy, restart attempts, failover delay, auto start value, active placement value, re-balance time, and name of the resource's action script. Some keywords are optional. For example, the following sample `named.cap` resource profile does not set an active placement value, which means that the placement of the application will not be reevaluated when a member boots into the cluster.

```
# cat named.cap
TYPE = application
NAME = named
DESCRIPTION = BIND Server
CHECK_INTERVAL =
FAILURE_THRESHOLD = 0
FAILURE_INTERVAL = 0
REQUIRED_RESOURCES =
OPTIONAL_RESOURCES =
HOSTING_MEMBERS =
PLACEMENT = balanced
RESTART_ATTEMPTS =
FAILOVER_DELAY =
AUTO_START =
ACTION_SCRIPT = named.scr
```

For detailed descriptions of each type of profile and keyword, see `caa`(4). In addition, see the *Cluster Highly Available Applications* manual and `caa_profile`(8) for more information on the contents and creation of application resource profiles.

Application profiles can be extended with User-defined Attributes. The value of these attributes can be defined either in an application profile, or on the command line of the CAA commands. The value of these attributes can then be accessed in an action script to customize how the action script executes during a start, stop or check of the resource. Any user-defined attributes are defined for all application resources in the application type definition file. For more information, see the *Cluster Highly Available Applications* manual.

The remainder of this section discusses placement policies, hosting members, active placement, and failure threshold and failure interval. Action scripts are described in Section 5.5.

An application's placement policy determines where the application is started. Supported policies are: `balanced`, `favored`, and `restricted`.

`balanced`

CAA favors starting or restarting the application resource on the member that is currently running the fewest application resources. Placement that is due to optional resources is considered first. Next, the host with the fewest application resources running is chosen. If no cluster member is favored by these criteria, any available member is chosen.

`favored`

CAA refers to the list of members in the `HOSTING_MEMBERS` attribute of the resource profile. Only cluster members that are both in this list and satisfy the required resources are eligible for placement consideration. Placement due to optional resources is considered first. If no member can be chosen based on optional resources, the order of the hosting members decides which member will run the application resource. If none of the members in the hosting member list are available, CAA favors placing the application resource on the member that is running the fewest application resources.

You must specify a hosting members list when you select a favored placement policy.

`restricted` This policy is similar to the favored placement policy, except that if none of the members on the hosting members list are available, CAA will not start or restart the application resource. A restricted placement policy ensures that the resource never runs on a member that is not on the list, unless you manually relocate it to that member.

You must specify a hosting members list when you select a restricted placement policy.

Hosting members are, in order of preference, members to consider when the application is (a) started, or (b) relocated. A hosting member list is used in conjunction only with the favored or restricted placement policies.

Active placement causes CAA to reevaluate the placement of an application when a new cluster member is added to a cluster or rebooted. If a more highly favored cluster member joins the cluster and active placement is on, then the application will stop on its current member and restart on the more favored member.

If you want application failback based on time of day, you can use the `REBALANCE` profile attribute instead of active placement. The application will be relocated to a preferred member at the time specified instead of when the cluster member rejoins the cluster.

Failure threshold and failure interval values are used together to stop an application that repeatedly fails. If an application fails too many times during the failure interval time, the application is not started again. These values are considered only when a check of the application fails, and not at initial start attempts.

The restart attempts value defines the maximum number of times that an application start or restart is attempted on one cluster member before that attempt is considered failed.

5.4.2 Nonapplication Resource Profiles

All other types of currently supported resources (network, tape, and media changer) have resource profiles that define which resource to monitor and specify the failure threshold and failure interval values. If a nonapplication resource fails too many times during the failure interval time, monitoring of the resource is stopped.

For tape and media changer resources, you define which tape to monitor by its device name; for a network resource you must define a subnet.

See the *Cluster Highly Available Applications* manual, `caa_profile`(8), and `caa`(4) for detailed descriptions of the contents and creation of resource profiles.

5.5 Action Scripts

An action script is a set of commands used by CAA to start, stop, and check an application. Only application resources have action scripts. The name of an action script is specified as the `ACTION_SCRIPT` value in the application's resource profile.

By default, action scripts are located in the `/var/cluster/caa/script` directory although they can be placed anywhere on a cluster file system. The file names of action scripts take the form *`resource_name`*`.scr`.

The *Cluster Highly Available Applications* manual provides examples of action scripts.

In function, an action script is similar to available server environment (ASE) scripts, and to the system initialization scripts located in the `/sbin/init.d` directory.

An action script has multiple entry points that are executed by the CAA commands when an application resource needs to be started or stopped. The `start` entry point is used by `caa_start` and `caa_relocate` to start an application, and the `stop` entry point is used by `caa_stop` and `caa_relocate` to stop an application. The `check` entry point is used by the resource manager to validate that an application is still running.

Each action script has an associated timeout value defined in its application resource profile. If the action script does not finish executing within this time, CAA considers the start attempt a failure and either attempts to start the application on another member or fails completely.

Optionally, standard output from a action scripts can be directed to standard output of the CAA command that invoked it.

Both the caa_profile command and the SysMan suite of applications can be used to create simple action scripts when creating resource profiles. You may need to edit these action scripts to customize the start, stop, and check procedures for an application.

6 Cluster Alias

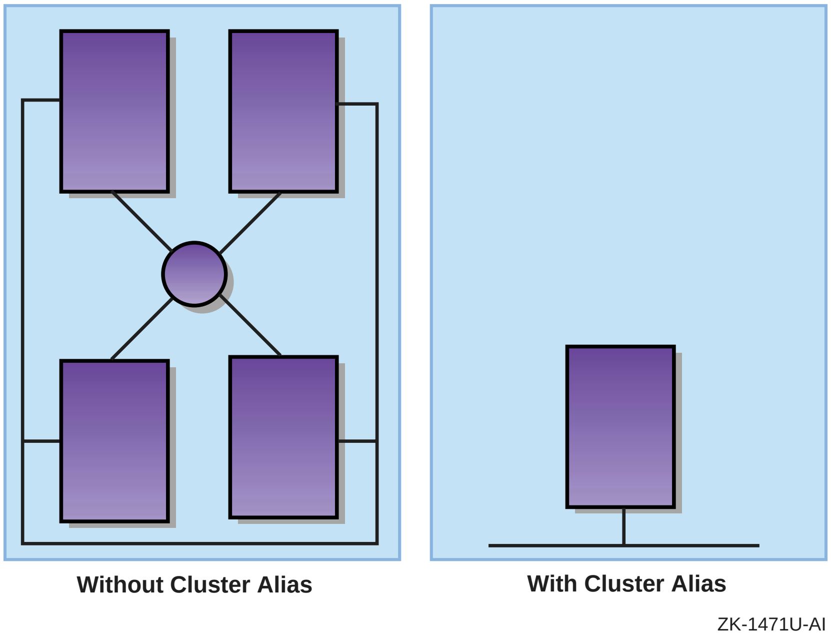
This chapter discusses the following topics:

- A general overview of cluster aliases (Section 6.1)
- Cluster alias subsystem components (Section 6.2)
- The default cluster alias (Section 6.3)
- The number of aliases per cluster (Section 6.4)
- The location of alias IP addresses (Section 6.5)
- Routing for alias IP addresses (Section 6.6)
- `in_single` and `in_multi` services (Section 6.7)
- Alias attributes (Section 6.8)
- Service port attributes (Section 6.9)
- vMAC support (Section 6.10)
- NFS and cluster alias (Section 6.11)
- RPC services and cluster alias (Section 6.12)
- ifconfig aliases and cluster aliases (Section 6.13)

6.1 Overview of Cluster Alias

A **cluster alias** is an IP address that makes some or all of the systems in a cluster look like a single system to Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) applications. Figure 6–1 shows how a network client views the systems in a cluster with and without a cluster alias.

Figure 6–1: Client View of a Cluster With and Without Cluster Alias



Cluster aliases free clients from having to connect to specific cluster members for services. Just like clients can request a variety of services from a single host, clients can request a variety of services from a cluster alias. For example, you can `telnet` or `rlogin` to a cluster alias like you do to a single host.

A cluster can have more than one cluster alias. One alias, the default cluster alias, is created during cluster installation and all members can receive packets that are addressed to this alias. You can create additional aliases as needed.

You can think of a cluster alias as a distributed virtual clusterwide network interface. In that sense, a cluster alias is conceptually similar to an `ifconfig` alias, where a single physical network interface responds to more than one IP address.

Each system in a cluster explicitly joins the aliases to which it wants to belong. After a system joins an alias, it is a **member** of that alias. Using the analogy that a cluster alias is similar to an address on virtual network interface, joining an alias is similar to issuing an `ifconfig up` command for that alias interface. The member can now receive packets addressed to the alias.

Clients send TCP connection requests or UDP messages to the IP address representing an alias. The cluster transparently routes the request or message to a cluster node that is a current member of that alias. The hop within the cluster uses the cluster interconnect, not network routing.

If a member of an alias is unavailable, the cluster stops sending packets to that member and routes packets to active members of that alias. As long as one member of an alias is active, the alias is available.

6.2 Cluster Alias Subsystem Components

The cluster alias subsystem has the following main components:

- The kernel portion of the cluster alias subsystem, `clua`, which is a configurable kernel subsystem loaded at boot time.
- A user-level daemon, `aliasd`. The kernel communicates with this daemon to manage routing for cluster aliases. The alias daemon transparently handles the routing configuration for cluster aliases, automatically adding any needed host routes (and network routes for alias addresses on virtual subnets) for cluster aliases to that member's `/etc/gated.conf.member`*n* file. The daemon starts `gated` using this file as `gated`'s configuration file rather than the member's `/cluster/members/`*{memb}*`/etc/gated.conf` file.

  The `cluamgr` command provides options that can modify the daemon's behavior. Each cluster member runs `aliasd`.

  Note

  The `aliasd` daemon supports only the Routing Information Protocol (RIP).

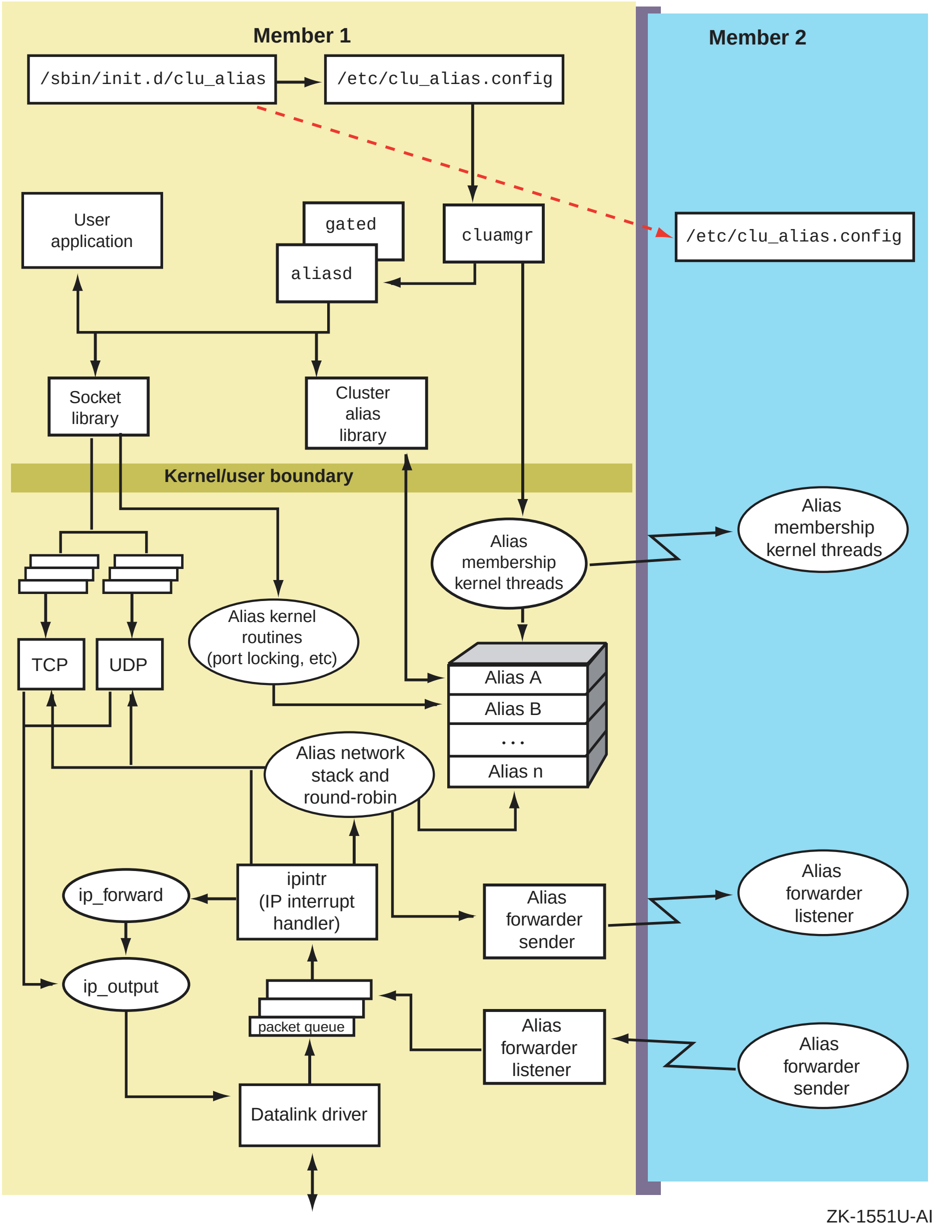
- An administrative interface that provides both a command-line and a graphical user interface (GUI) to manage aliases and alias attributes. The command-line interface is the `cluamgr` command. The GUI is accessed from the SysMan Menu.
- A member-specific alias configuration file, `/etc/clu_alias.config`, which contains the `cluamgr` commands that configure aliases, including the default cluster alias, for that member.
- A clusterwide application configuration file, `/etc/clua_services`, which assigns alias-related attributes to ports used by services. The `/etc/clua_services` file is the cluster alias extension of the `/etc/services` file. The `clua_services` file extends the `services` syntax to assign alias-related attributes to ports.
- A clusterwide file, `/etc/exports.aliases`, which contains the names of non-default cluster aliases that NFS clients can use. By default, only NFS requests directed to the default cluster alias are accepted by the

cluster. This file lets you use additional aliases as NFS server names. This is useful, for example, when not all members of a cluster are directly connected to the storage that contains exported file systems. In this case, you can create an alias that encompasses only those members directly connected to the storage, and then tell users on NFS client systems to use that cluster alias when requesting NFS services from the cluster.

- An application programming interface (API), `libclua`.

Figure 6–2 provides a functional overview of the cluster alias subsystem components.

Figure 6–2: Cluster Alias Functional Overview



6.3 Default Cluster Alias

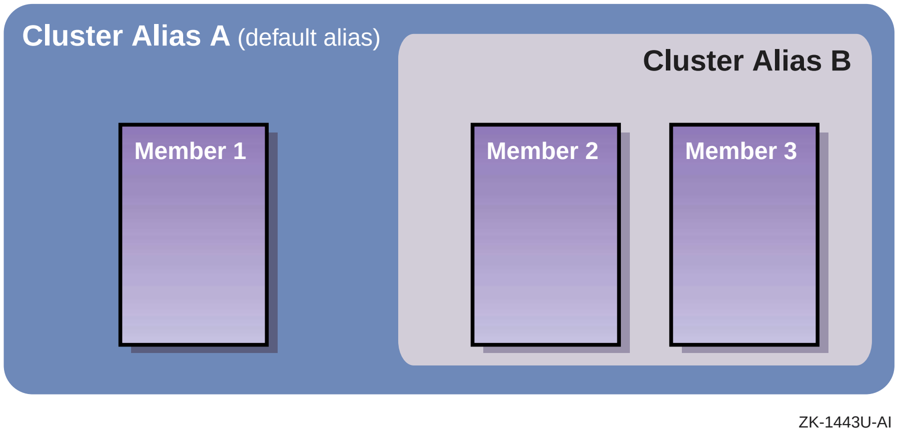
The **default cluster alias** is a special alias. During installation, the cluster is given a name, which is stored in `/etc/sysconfigtab` as the value of the cluster_name attribute. The installation procedure adds an entry to /etc/hosts, which associates this cluster name with a user-specified default cluster alias IP address. For example, for a cluster named deli whose alias IP address is 16.140.112.209, the installation procedure adds the following entry to /etc/hosts:

```
16.140.112.209   deli.zk3.dec.com   deli
```

Each cluster member is a member of the default cluster alias. The cluamgr command that causes a cluster member to join the default cluster alias is in each member's /etc/clu_alias.config file. Because each cluster member runs the commands in its clu_alias.config file at boot time, all cluster members automatically join the default cluster alias.

Figure 6–3 shows a three-node cluster with two cluster aliases. All members belong to alias A, the default cluster alias, but only two members belong to alias B.

Figure 6–3: Cluster Using Two Aliases



6.4 Number of Aliases per Cluster

After cluster installation, you can define as many aliases as are needed for a cluster. The default value for the clua subsystem max_aliasid attribute is 8, the maximum value is 102,400. In practice this upper limit is probably memory restricted, but the useful range probably meets all practical needs.

For many clusters, the default cluster alias provides sufficient access for cluster clients. Whether or not a cluster will benefit from having additional aliases depends on the symmetry (storage and network) of the cluster, and whether you want all members to handle client requests for all services. Additional aliases are useful in the following situations:

- In a heterogeneous cluster where some devices or applications are best served through a subset of cluster members.
- If you want to restrict services to a subset of cluster members in order to reduce the internal forwarding of requests and packets.
- In a cluster where all members are not directly connected to the storage containing files systems exported by the cluster. In this case, using an alias that encompasses just those systems that are directly connected to this storage reduces traffic across the cluster interconnect.

One suggestion is to use the default alias for a while, and then decide whether your site can benefit from additional aliases. In many cases, the default alias is sufficient; the *Cluster Administration* manual describes a situation where a site uses two aliases for load balancing.

6.5 Location of Alias IP Addresses

Cluster alias IP addresses, including that of the default cluster alias, must be on a network accessible to cluster clients; that is, clients must be able to route to this subnet. For this reason, cluster alias IP addresses cannot be on the cluster interconnect (the subnet used by the cluster for internal communication).

A cluster alias address can be in one of two types of subnets:

common subnet

A subnet to which one or more cluster systems are connected with physical network interfaces.

Using a common subnet for cluster aliases works well when the cluster is connected to only a single local area network, and that network is managed as a single IP address domain.

Cluster alias routing in a common subnet is based on proxy Address Resolution Protocol (ARP) support. For each alias, one cluster member acts as the proxy ARP master for that alias.

virtual subnet

A cluster alias resides in a virtual subnet if its address is in a subnet that is not associated with any physical interfaces. A virtual subnet is made visible to the physical network by `gated`, the gateway routing daemon.

If you use the `cluamgr` command to assign the `virtual` option to an alias address, the cluster member on which the command is run will advertise both a host route and a network route to the alias.

Multiple clusters on the same LAN can use the same virtual subnet.

Caution

A virtual subnet must not have any real systems in it.

The choice of subnet type depends mainly on whether the existing subnet to which the cluster is connected (that is, the common subnet) has enough addresses available for cluster aliases. If addresses are not available on an existing subnet, consider creating a virtual subnet. A lesser consideration is that if a cluster is connected to multiple subnets, configuring a virtual subnet has the advantage of being uniformly reachable from all of the connected subnets. However, this advantage is more a matter of style than substance. It does not make much practical difference which type of subnet you use for cluster alias addresses; do whatever makes the most sense at your site.

Regardless of the type of subnet, it must be configured so that packets from clients can be routed to alias addresses. Services that use cluster aliases will not be accessible to clients if those alias addresses are on a virtual or a common subnet that clients cannot reach.

A cluster alias address should not be a broadcast address or a multicast address, nor should it reside in the subnet used by the cluster interconnect. Although you can assign a cluster alias an IP address that resides in one of the private address spaces defined in RFC 1918, you must use the `cluamgr -r resvok` command in order for the alias subsystem to advertise a route to the alias address. (See `cluamgr`(8) for information on using the `resvok` flag and how to add an entry to `/etc/rc.config.common` to make the route advertising persist beyond reboots.)

6.6 Routing for Alias IP Addresses

This section discusses how routes to aliases are advertised and how packets addressed to aliases are taken off the wire:

- Advertising routes to aliases (Section 6.6.1)
- Routing for aliases on common subnets (Section 6.6.2)
- Routing for aliases on virtual subnets (Section 6.6.3)
- Summary of routing for aliases on common and virtual subnets (Section 6.6.4)
- Accepting and redirecting packets and connection requests addressed to an alias (Section 6.6.5)

- Routing example (Section 6.6.6)

The following terms, which are used in these sections, are defined in the glossary. If you are not familiar with the terms, read the glossary definitions before continuing.

- host route
- network route
- proxy ARP

6.6.1 Advertising Routes to Aliases

An alias router is a cluster member that makes a cluster alias address known to the network and receives incoming packets for that alias. By default, all cluster members are configured as alias routers for the default cluster alias at boot time. Any cluster member can be configured to advertise a host or a network route to any alias.

Note

By default, cluster members route only for cluster aliases; they are not configured as general purpose routers. Whether or not a site decides to configure one or more cluster members to route for non-alias traffic is the responsibility of the network administrators at that site.

A cluster member does not have to join an alias in order to route for that alias. In the following example, a cluster member specifies `alias1`, and specifies and joins `alias2`. The member will route packets addressed to either alias, but will only receive requests/packets addressed to `alias2`:

```
/usr/sbin/cluamgr -a alias=alias1
/usr/sbin/cluamgr -a alias=alias2,join
```

6.6.2 Routing for Aliases on Common Subnets

By default, a cluster member that has specified or joined a cluster alias uses `gated` to advertise a host route to that alias. The `aliasd` daemon automatically configures `/etc/gated.conf.member`*n* to advertise a host route at boot time based on the information in `/etc/clu_config.alias`.

Only one alias member at a time responds to Address Resolution Protocol (ARP) requests for a given cluster alias. This member is the proxy ARP master for the alias. If this system fails, another is elected to take over the role of proxy ARP master.

Note

Proxy ARP applies only to alias addresses configured on a common subnet (a physical network).

If multiple cluster alias addresses are defined, you can use the rpri alias attribute to balance the incoming load a bit by giving different cluster members the highest routing priority for different alias addresses. With a single alias and ARP-based routing, only one member acts as the alias router at a time. (Section 6.8 describes the router priority, rpri attribute.)

However, all cluster members that are aware of an alias use gated to set up a host route to each cluster alias on each of their network interfaces. Because host routes take precedence over the interface routes used with ARP, any client in the same subnet as the cluster that is running a route daemon sees the host routes. Depending on various random occurrences, such as which cluster members boot in what order, different clients may see a different cluster member's host route first. Therefore, clients might use different cluster members as their route to the cluster alias. (This is not guaranteed to be uniformly distributed. If the clients boot before the cluster, all clients will see and use the host route through the first cluster member that advertises one.) Client systems that do not run a route daemon, such as routed or gated, find the cluster alias using the ARP protocol, and are routed through the proxy ARP master.

In summary, if a client is using ARP, and not host routes, to resolve the route to the alias, the cluster member responding to ARP will be the system to which incoming packets will be initially directed and thus the system doing the tunneling. If a client uses host routes (and a router typically does), the rpri setting has no effect on the path taken by incoming packets.

6.6.3 Routing for Aliases on Virtual Subnets

By default, a cluster member that has specified or joined a cluster alias uses gated to advertise a host route to that alias. The alias daemon, aliasd, creates a /etc/gated.conf.member*n* file for each cluster member. The alias configuration process modified this configuration file to advertise each alias address in a virtual subnet as a host route. No manual modification is required; each member's alias daemon automatically modifies that member's /etc/gated.conf.member*n* file to advertise a route to each cluster alias host address through each network interface on that member.

If the cluamgr command's virtual option is assigned to an alias address, the cluster member also advertises a network route to the virtual subnet containing the alias address. To ensure that the virtual subnet's location is known to the network, make sure that at least one, and preferably all members, specify the cluamgr command virtual=t option for at least one alias in each virtual subnet.

Like common subnet route advertising, the routing load may be balanced across multiple cluster members, depending on which route advertisements the clients see in what order.

6.6.4 Summary of Routing for Aliases on Common and Virtual Subnets

Table 6–1 summarizes the types of routes that are advertised for cluster aliases on common and virtual subnets with the default routing daemon configuration, aliasd controlling gated.

Table 6–1: Summary of Routing for Aliases on Common and Virtual Subnets

Subnet Type	Address Domain	Proxy ARP	Host Route via gated	Network Route via gated
Common	A subnet to which a cluster is connected.	Yes[a]	Yes[b]	No
Virtual	A subnet with no physical connections that appears to exist "behind" the cluster.	No	Yes[b]	Yes[c]

[a] For any interface on that subnet on a cluster alias that has specified or joined the alias.
[b] For each interface on a cluster member that has specified or joined the alias.
[c] For each interface on a cluster member that has specified or joined the alias, and that has also specified the cluamgr option virtual=t for at least one cluster alias on that virtual subnet.

Note

The default routing daemon configuration for a cluster member is the gated daemon using the /etc/gated.conf.member*n* daemon configuration file for that member. The cluster member automatically advertises host routes (and network routes for virtual networks when virtual is specified) to cluster aliases. If you use the stock gated configuration file (/etc/gated.conf), another routing daemon (routed), or no routing daemon at all, you have to provide static routes in order for clients to access the cluster aliases.

For more information, see the *Cluster Administration* manual.

6.6.5 Accepting and Redirecting Packets and Connection Requests Addressed to an Alias

Normal routing ensures that a packet addressed to a cluster alias arrives at exactly one cluster member. (One packet is not handled by multiple cluster members.) That member determines which alias member will receive and process the packet based on the following:

- Which members of the alias are available (the alias subsystem monitors calls to `bind()` and `listen()`)
- The port number
- The type of packet
- A weighted round-robin algorithm

The following table describes how packets are redirected within a cluster:

New TCP/IP connection	Look at the packet and make a list of eligible members for the target port. Use the weighted round-robin algorithm to select a member from the list of active listening members. Forward the packet to the selected member.
Existing TCP/IP connection	Determine which alias member owns this connection. Forward the packet to the member.
UDP	Look at the packet and make a list of eligible members for the target port. Use the weighted round-robin algorithm to select a member from the list of active listening members. Forward the packet to the selected member. (See Section 6.11 for information on how NFS over UDP is handled by the cluster alias subsystem.)
ICMP (some ICMP packets must be handled in cluster-alias context)	Look at the packet and determine whether to handle it or forward it to another member.

As an example, we can examine the routing for a TCP connection request for a multi-instance service. Assume that a client system will make a TCP connection to a cluster alias IP address, and that there is a router between the client and the cluster members.

- Only one node in the cluster will respond to ARP requests to the cluster alias, but all members of that alias will advertise host routes to it. Because the router will see and process RIP protocol, it will see the host routes, and will probably use that in preference to the ARP request. The cluster node in the router's route table for the alias address may or may not be the one responding to ARP requests, but the router (assuming RIP) will only have a single route table entry for that cluster alias. We'll assume that Member1 is in the router's route table entry for the cluster alias in question.

- Requests from the client destined for Member2 will pass through Member1 because of the route table entry in the router.

  Packets from Member2 will go by whatever is the most efficient route to the client, directly out its network interface card (NIC), not through Member1. In most cases where the cluster is a server, the bulk of data transfer is outbound; that is, a short client request inward gets tunneled (client→R→M1→M2; a long data response goes directly to the client (M2→R→client).

  Although an extra hop is involved for inbound traffic, no extra hop is required for outbound traffic. (The extra hop is lower overhead than a full network hop because the fully-formed `mbuf` packet is tunneled across the cluster interconnect to the network stack on Member2, not forwarded down out and back up again.) Applications that have large volumes of inbound data are more likely to see the effect of the extra hop than applications that have short inbound requests and large outbound data transfers.

  Specialized applications that want to load-balance using the cluster alias for assignments, but need to avoid even the extra inbound hop can use the cluster alias as a binding agent and reconnect to the node that accepts the initial connection. (See the *Cluster Highly Available Applications* manual for information about creating cluster-aware applications.)

- Each separate TCP connection has the potential to be distributed to any cluster node listening on that port. If a given instance of the service retains client context that must be retained across multiple connections (that is, the application is not per-connection idempotent), the application should either use a unique port number for those connections, or be configured as a single-instance service. A clusterwide service using a single port number clusterwide is presumed to be able to render the same service from any cluster node that is a member of the cluster alias.

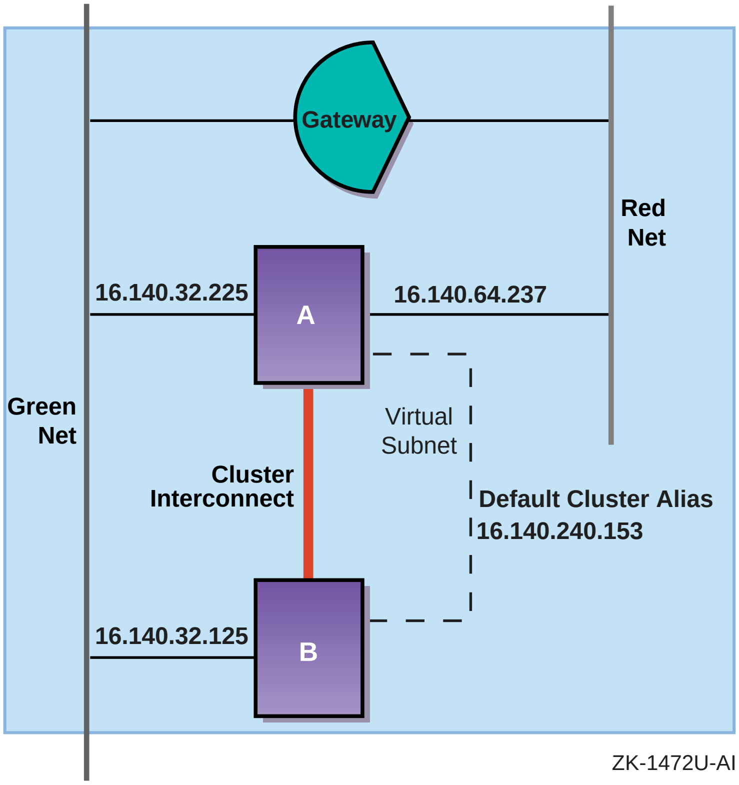
  Inbound traffic continues to use the member node that is in the router's route table for the alias because neither the router nor the client has any way of knowing which cluster node was assigned the connection (the address used in the connection does not change; it is the cluster alias address). The affinity information binding a connection to a member node is known only within the cluster.

- If Member1 fails and another member becomes the proxy ARP master, the new master will broadcast a MAC address associated with the cluster alias IP address to the router. A gratuitous ARP message is sent by the member assuming proxy ARP mastership. You can also use the virtual MAC (vMAC) option, which allows the cluster to create and use a unique MAC address for each cluster alias. The vMAC address is passed from master to master as needed. (vMAC is required only if the router does not respond to gratuitous ARP messages.) See Section 6.10 for information about vMAC.

6.6.6 Routing Example

Figure 6–4 shows a cluster with interfaces on three networks, two public common networks and one private virtual network. The default cluster alias IP address is on the virtual subnet.

Figure 6–4: Alias Routing Example



If the correct `cluamgr` commands are used to configure the alias, the `gated` daemon on each node will advertise on all connected networks:

- A host route to that address for the benefit of local nodes.
- A network route to the virtual network to ensure that nodes beyond these networks can locate the virtual subnet.

The following examples show the cluamgr commands run on hosts A and B to advertise routes to the cluster alias on the virtual subnet:

- To have gated advertise a host route for the alias:

```
# cluamgr -a alias=16.140.240.153
```

- To have gated advertise a host route and a network route (16.140.240.0) for the alias:

```
# cluamgr -a alias=16.140.240.153,virtual=t
```

- To have gated advertise a host route and a network route (16.140.240.0) for the alias, and to receive packets and connection requests addressed to the alias:

```
# cluamgr -a alias=16.140.240.153,virtual=t,join
```

6.7 in_single and in_multi Services

Service ports that are accessed through a cluster alias are defined as either **in_single** or **in_multi**. These service port attributes determine the routing of network requests to applications, not whether an application can run on more than one member at the same time. From the point of view of the cluster alias subsystem:

- When a service's port is designated as in_single, only one alias member receives connection requests or packets for that service. If that member becomes unavailable, the cluster alias subsystem selects another eligible member of that alias to receive all connection requests or packets.
- When a service's port is designated as in_multi, the alias subsystem distributes connection requests and packets among all eligible members of the alias.

By default, the cluster alias subsystem treats all services as in_single. For the cluster alias subsystem to treat a service's port as in_multi, the port must either be registered as in_multi in /etc/clua_services or through a call to clua_registerservice(). See Section 6.9 for more information on service port attributes.

A service whose port is designated as in_multi can take advantage of cluster aliasing to distribute incoming TCP connection requests and UDP packets among members of the alias. The alias subsystem provides load balancing through a weighted round-robin algorithm that distributes requests/packets among alias members. If one member of an alias cannot respond to client requests, the cluster alias software transparently distributes requests/packets among the remaining alias members. (The weighting portion of the algorithm is controlled by the selection weight, selw, that each member of an alias assigns to that alias. See Section 6.8 for information about selection weights.)

Note

Cluster alias and CAA are separate subsystems with complementary but different functions. CAA is an application-control tool; cluster alias is a routing tool. CAA decides where an application will run; cluster alias decides how to get there. You cannot use CAA to control routing within the cluster; you cannot use cluster aliases to control where an application is running in the cluster. The *Cluster Administration* manual provides more information on the differences between cluster alias and CAA.

The following two figures show how the cluster alias subsystem distributes client requests for `in_single` and `in_multi` services. For the `in_single` service (Figure 6–5), all requests are sent to the alias member currently running the service. For the `in_multi` service (Figure 6–6), requests are distributed among all alias members.

Figure 6–5: in_single Service Accessed Through Default Cluster Alias

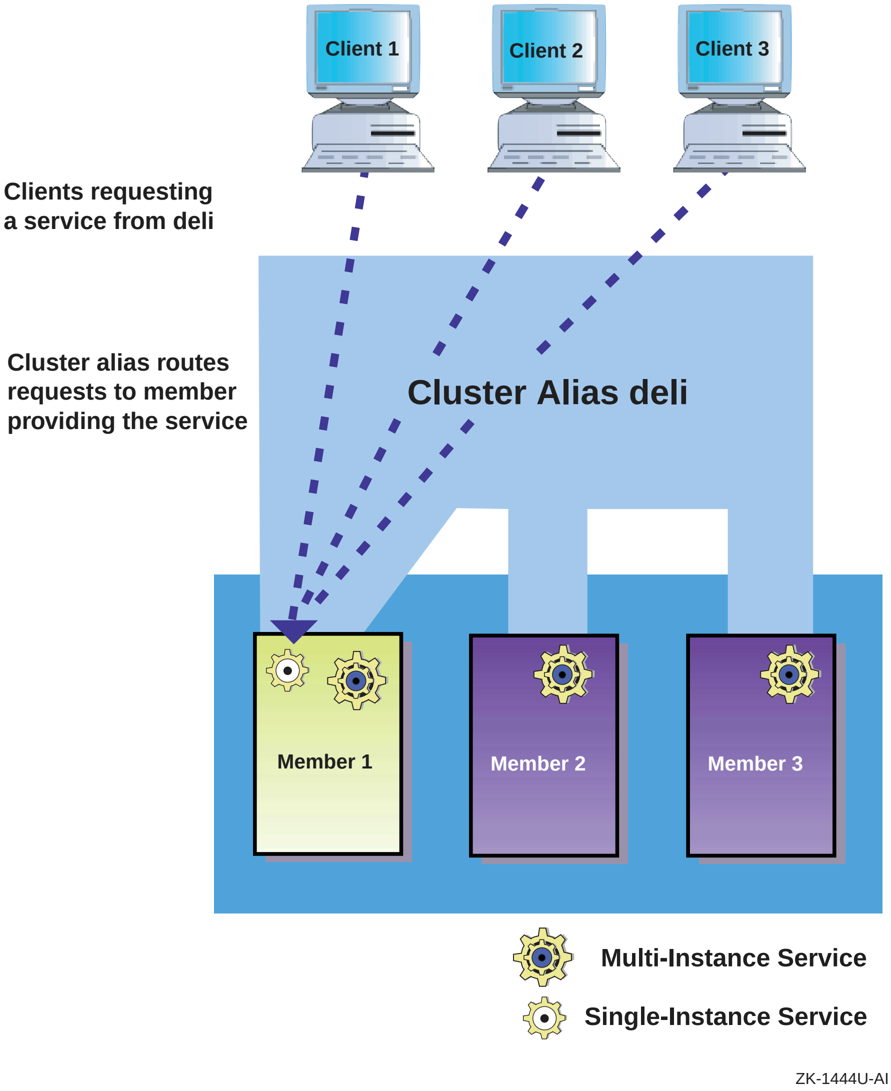
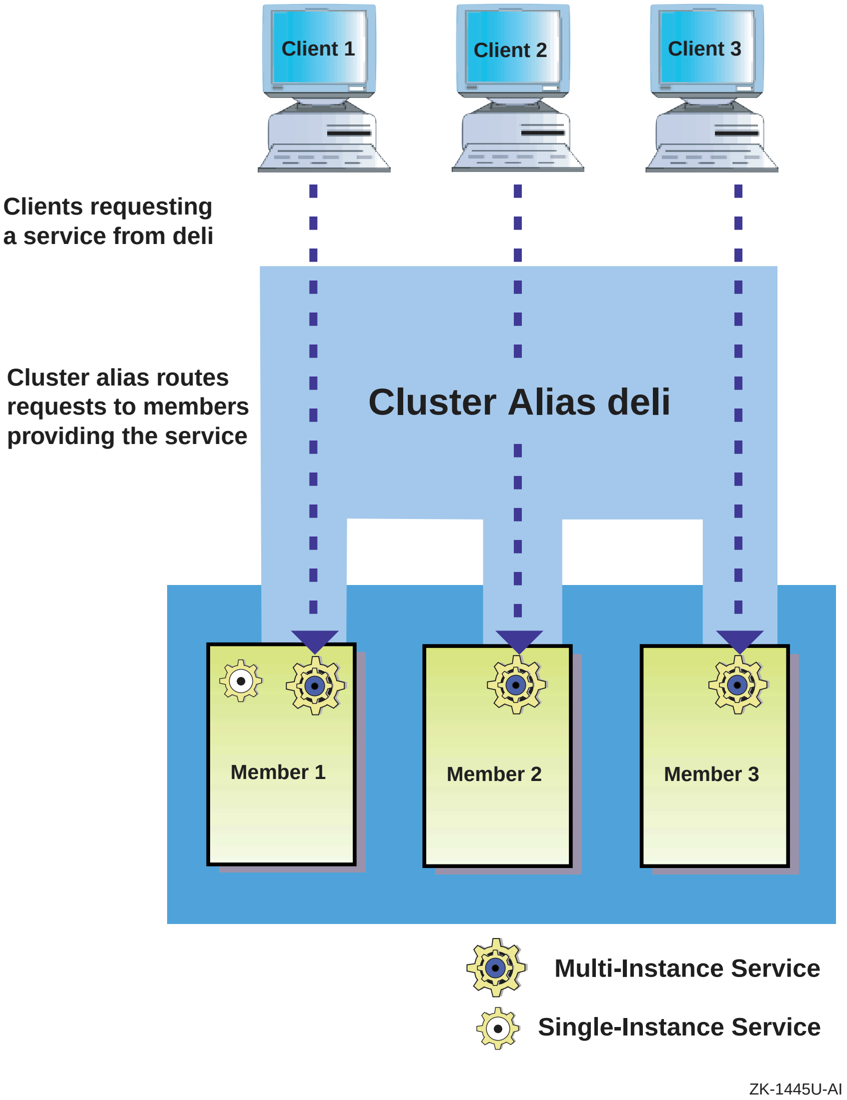


Figure 6–6: in_multi Service Accessed Through Default Cluster Alias



6.8 Alias Attributes

This section discusses alias attributes in general, and three in particular: router priority (`rpri`), selection priority (`selp`), and selection weight (`selw`).

Aliases and their attributes are managed through the `cluamgr` command and the SysMan Menu. The SysMan Menu calls `cluamgr` as needed.

Alias attributes are member-specific. Each cluster member has its own view of an alias. For example, one cluster member can route for an alias but not be a member of that alias, but another cluster member can both route for that alias and be an end recipient for requests or messages addressed to that alias. In a like manner, alias attributes are also alias-specific. A cluster member can join two aliases and assign a different selection weight to each alias, thus ensuring that the member system receives a higher proportion of connections addressed to the alias with the higher selection weight.

The following attributes control the routing and distribution of connection requests and packets among members of an alias. The descriptions are paraphrased from those in `cluamgr`(8), which describes these and other alias attributes.

router priority

The router priority (`rpri`) controls the proxy ARP router selection for an alias on a common subnet. (This option is irrelevant for an alias whose address is in a virtual subnet.)

For each alias in a common subnet, the cluster member with the highest router priority for that alias responds to ARP requests for that alias. This option does not control which members broadcast host or network routes for aliases.

When a cluster has more than one cluster alias, you can use router priority to spread the proxy ARP response overhead for aliases among cluster members.

selection priority

The selection priority (`selp`) identifies subsets of members of an alias for the assignment of new connection requests.

The selection priority establishes a hierarchy within the members of an alias. Connection requests are distributed among those members sharing the highest selection priority value. If an alias has three members, two with `selp=10` and one with `selp=5`, no connection requests or messages are given to the `selp=5` member as long as either of the `selp=10` members is available.

You can use selection priority values to set up a failover order for members of a particular cluster alias.

Selection priority applies only to applications that are registered as `in_multi` services. (Selection priority constrains the set of nodes in the round-robin process, and the round-robin process only applies to multi-instance applications.) It has no effect on the distribution of connection requests for an `in_single` service.

selection weight

The selection weight (`selw`) provides a simple, static method for controlling which members of an alias get the most connections.

Selection weight indicates the number of connections (on average) that this member is given before connections are given to the next alias member with the same `selp` value. (The `selp` value determines the order in which members are eligible to receive requests or messages; the `selw` value determines how many requests or messages a member gets after it is eligible.)

If node A is larger than node B and can handle 50 percent more connections, then assign, for instance, `selw=3` to an alias on node A, and `selw=2` to an alias on node B.

Selection weight applies only to applications that are registered as `in_multi` services. (All traffic for an `in_single` service must go to the cluster member running that service.)

Selection weight and routing priority address two different load balancing issues; selection weight balances application overhead within a cluster, and router priority balances proxy ARP response overhead within a cluster.

In general, the default routing priority provides acceptable performance. The selection weight is probably more useful when balancing application loads within a heterogeneous cluster consisting of both large and small systems.

6.9 Service Port Attributes

The `/etc/clua_services` file is a shared file that is read by all cluster members. The file is similar in concept and syntax to the `/etc/services` file. The `clua_services` file provides a method for associating alias-related attributes with the port numbers used by services. (When application source code is available, the `clua_registerservice()` function serves the same purpose.) Any service with a fixed port assignment can have an entry in `/etc/clua_services`.

You can associate the following attributes with a service's port:

in_single

A service that, from the cluster alias point of view, runs on only one cluster member at a time, but can fail over to another instance of the service on another member if the active service goes away. (Active, in this context, relates only to messages addressed to the cluster alias. All instances of a service are always active for their node's local IP addresses unless the `in_nolocal` attribute is also set.) As each service binds to the application's port, the first is flagged as active for the alias, and the others are flagged as inactive. If the active service fails, one of the inactive service daemons is marked as active.

Any port that is not explicitly listed in the `clua_services` file as `in_multi`, or registered as `in_multi` through a call to the `clua_registerservice()` function, is treated as `in_single`.

in_multi

Indicates a service that can run concurrently on two or more cluster members. For a service using UDP, each packet might go to a different alias member. For a service using TCP, each connection is bound to a single alias member, but different connections to the service from the same client might be established on different alias members.

An `in_multi` service must be explicitly registered, either in the `/etc/clua_services` file or through the `clua_registerservice()` function.

in_noalias

Indicates that the port does not honor connection requests to alias addresses.

in_nolocal

Indicates that the port does not honor connection requests to nonalias addresses. For TCP, the port does not accept connections; for UDP, the port drops messages.

out_alias

Indicates that the default cluster alias is used as the source address whenever this port is used as a destination. Normally, outbound connections (or UDP messages) use the local IP address of the cluster member on which the client is running. It is often beneficial to use the cluster alias address as the source address for outbound traffic from the cluster (for example, to simplify authentication). Several standard Internet services, such as `telnet` and `login`, use the IP address of the default cluster alias as the source address for outgoing packets.

The `out_alias` attribute applies only when the connection (assuming TCP, not UDP) is originated from the cluster; that is, the cluster is the client. If a process running on a cluster member initiates an outbound connection, and the destination port (the port representing that half of the connection that is not in the cluster) is flagged in the cluster's `/etc/clua_services` file as `out_alias`, the connection uses the default alias as its source address.

The same logic holds true when the outbound traffic is a UDP send, because each send can be viewed as a microconnection.

static Indicates that the port cannot be assigned as a dynamic port. This option is assigned to ports between 512 and 1024 that are used by well-known, multi-instance network services that are always started at boot time.

With the exception of the `out_alias` attribute, these attributes apply to services accessed through any cluster alias. The `out_alias` attribute, which applies only to connections originating from the cluster, is specific to the default cluster alias.

The `in_multi`, `in_single`, and `in_noalias` attributes are mutually exclusive. The `in_nolocal` and `in_noalias` attributes are mutually exclusive. See `clua_services`(4) and `clua_registerservice`(3) for more information about the use of these attributes.

6.10 vMAC Support

When a cluster alias IP address is configured in a common subnet, one cluster member in that subnet acts as the alias's proxy ARP master, responding to local ARP requests addressed to the alias. If another member of the alias takes over the proxy ARP master role, the new master broadcasts a gratuitous ARP packet to inform other systems about the new hardware media access control (MAC) address associated with the alias's IP address. The other local systems then update their ARP tables to reflect this new cluster-alias-to-MAC association.

However, this broadcast packet is a problem for systems that do not understand gratuitous ARP packets. These systems do not become aware of changes in the cluster alias-to-MAC association, and continue to send alias traffic to the stale MAC address until the normal timeout interval for their ARP tables has elapsed. A solution is to provide a virtual hardware address (vMAC address) for each cluster alias.

A virtual MAC address is a unique hardware address that can be automatically created for each alias IP address. An alias vMAC address follows the cluster alias proxy ARP master from node to node as needed. Regardless of which cluster member is serving as the proxy ARP master for an alias, the alias's vMAC address does not change.

The *Cluster Administration* manual describes how to enable vMAC support for a cluster alias.

6.11 NFS and Cluster Aliases

When a cluster is configured as an NFS server, NFS client requests must be directed either to the default cluster alias or to an alias listed in `/etc/exports.aliases`. NFS mount requests directed at individual cluster members are rejected.

As shipped, the default cluster alias is the only alias that NFS clients can use. However, you can make additional cluster aliases available for use by NFS clients by putting the alias names in the `exports.aliases` file. This feature is useful when some members of a cluster are not directly connected to the storage containing exported file systems. In this case, creating an alias with only directly connected systems as alias members can reduce the number of internal hops required to service an NFS request.

The remainder of this section discusses how NFS and the cluster alias subsystem interact for TCP and UDP NFS traffic. (We recommend that, whenever possible, you use UDP as the network transport.) In the following scenarios, assume that the client's first interaction with the cluster is to mount a file system exported by the cluster, and that all cluster members are connected both to a common network and to storage.

- Getting packets off the network (Section 6.11.1)
- Mount requests (Section 6.11.2)
- NFS over TCP (Section 6.11.3)
- NFS over UDP (Section 6.11.4)

6.11.1 Getting Packets Off the Network

NFS requests using TCP or UDP are addressed to the default cluster alias or to an alias whose name is in `/etc/exports.aliases`.

By default, a cluster member advertises a host route to each alias that it has specified or joined. Because a client tends to cache the first host route to an alias that it sees, as long as that route is available the client will send all packets for an alias to the same cluster member.

All packets pass through this member on the way in, but not necessarily on the way out. The cluster member that puts the response packet on the wire inserts the cluster alias address as the source address, so the client is satisfied: send a packet to an alias, receive a packet from that alias.

6.11.2 Mount Requests

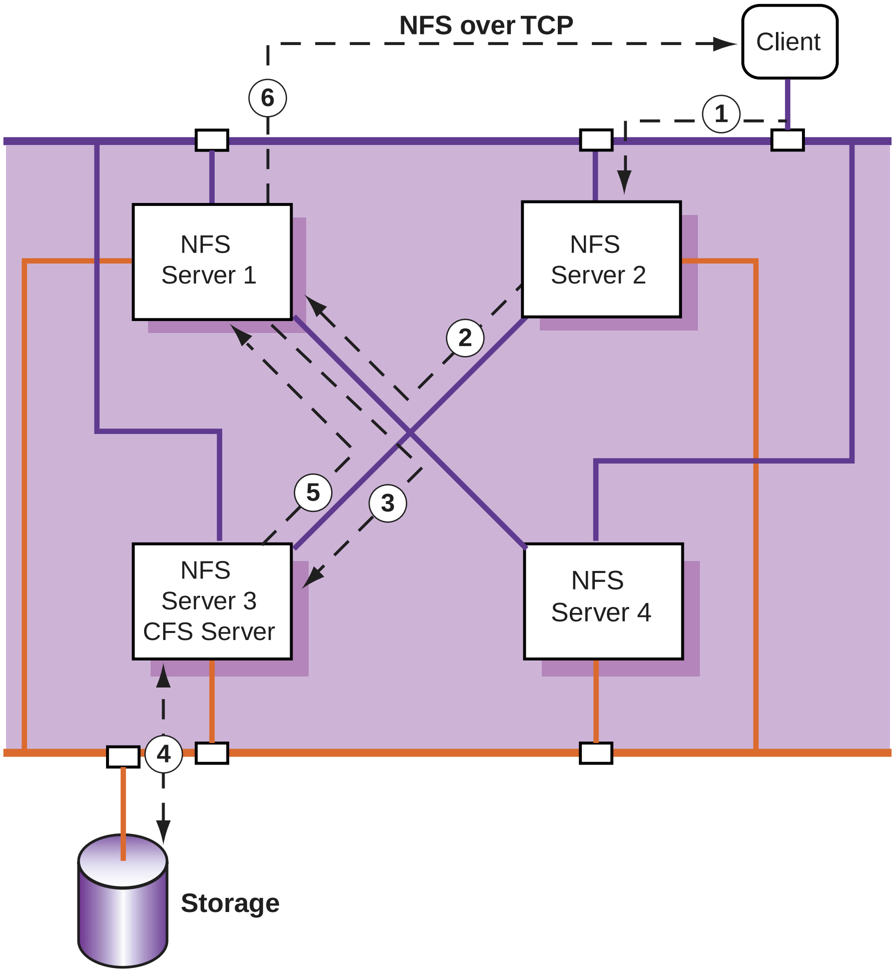
The mount daemon, `mountd`, is a multi-instance service that handles incoming UDP and TCP mount requests. The cluster alias subsystem decides which `mountd` instance services a mount request. The node handling the mount request has no relationship to the node that will eventually service the incoming NFS packets, although by chance they may end up being the same node.

6.11.3 NFS over TCP

TCP connection requests are assigned to a member based on the alias round-robin algorithm and alias selection weights. (Because the connection request does not include file system information, the cluster cannot route the request to the member that is currently the CFS server for the file system.) All subsequent TCP NFS packets for that file system from that client are handled by the same member that was assigned the connection, regardless of file-system relocations.

Use Figure 6–7, and the callouts that follow the figure, to trace the path of an NFS TCP connection:

Figure 6–7: NFS over TCP



ZK-1801U-AI

1. Because clients cache routes, the NFS request most likely goes through the same member as the mount request.

2. Because a TCP connection has already been established by the initial request, the member that takes the packet off the wire automatically tunnels it to the NFS server member that is handling the connection. (The node that takes the packet off the wire has no CFS server lookup.)

Note

The following steps are based on the assumption that the NFS server member is not the CFS server for the file system.

3. The NFS server sends a CFS request across the interconnect to the member that is the CFS server.
4. The CFS server member handles the I/O to storage.
5. The CFS server member returns the results across the interconnect to the NFS server member.
6. The NFS server member replies to the client (using the alias address as the source address).

6.11.4 NFS over UDP

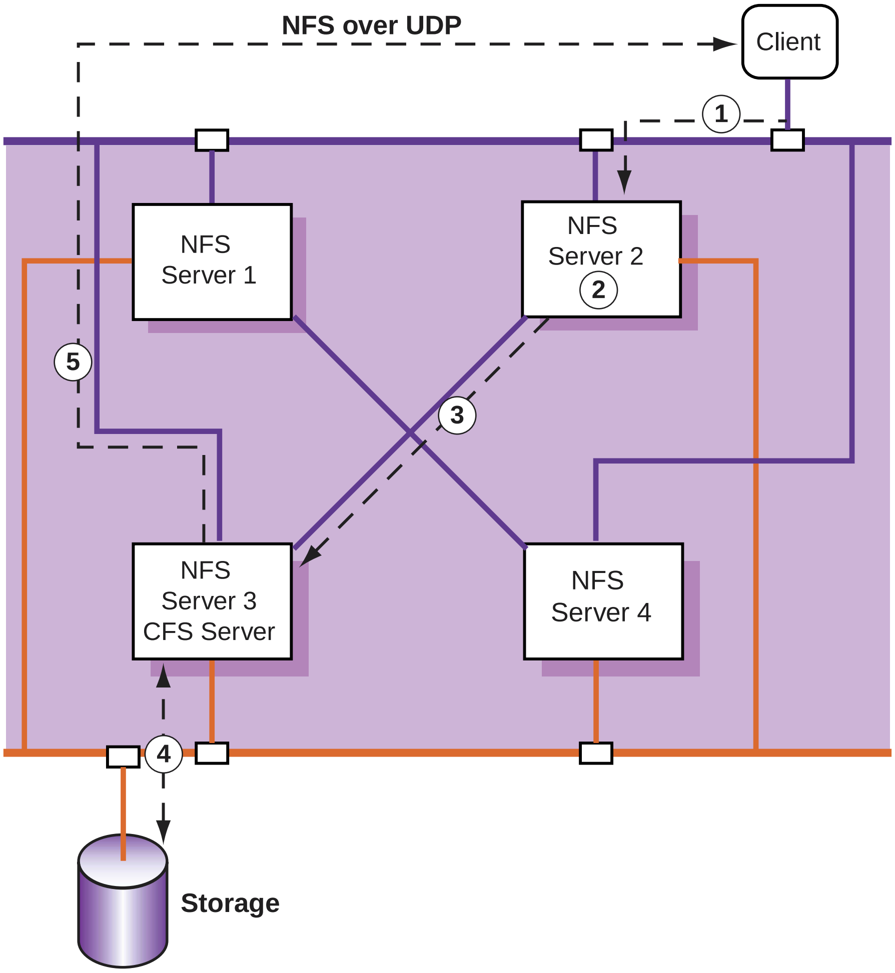
UDP NFS packets are redirected to the cluster member that is serving the file system. All UDP NFS traffic for that file system is handled on that member. If another cluster member becomes the CFS server for the file system, UDP packets are tunneled to the new server. UDP packets always follow the CFS server for the file system.

Note

Some clients, for example PCs, broadcast UDP requests when trying to find an NFS server. The cluster responds to these requests by returning the IP address of the default cluster alias. This ensures that later NFS client requests are sent to the default cluster alias.

Use Figure 6–8, and the callouts that follow the figure, to trace the path of an NFS request over UDP:

Figure 6–8: NFS over UDP



ZK-1800U-AI

1. Because clients cache routes, the NFS request most likely goes through the same member as the mount request.

2. Because an NFS UDP packet contains all the data needed for the NFS request, the cluster alias software on the member that takes the packet off the wire can determine which file system contains the file, and performs a CFS-callout to determine which cluster member is the CFS server for the file system.

Note

> The remaining steps are based on the assumption that the CFS server is a member of the cluster alias to which the packet was addressed.

3. The packet is tunneled to the NFS server that is also the CFS server for the file system. CFS can service the request locally. (UDP can provide better performance than TCP because the UDP packet makes just one trip across the interconnect. Because the NFS server is also the CFS server, no extra trips are needed to handle the CFS I/O.)
4. The NFS server/CFS server member handles the I/O to storage.
5. The NFS server/CFS server member replies directly to the client (using the alias address as the source address).

Note

> If the CFS server for the file system is not a member of the alias, the receiving node round-robins the packet in the same manner that it handles TCP connection requests. In this case, UDP performance will be worse than TCP performance because, with TCP, all incoming packets from a client are tunneled to the node servicing the connection. As a result, all I/O to the same file from the same client is handled by the same node. However, with UDP, if the CFS server is not a member of the alias being used, each I/O request for a given file will end up being handled by a different cluster member. In this case, the CFS clients cannot cache data; they will be constantly invalidating each other's caches and writing through the CFS server node.

6.12 RPC Services and Cluster Aliases

RPC services can call either the `clusvc_getcommport()` function or the `clusvc_getresvcommport()` function to bind to a port. (Use `clusvc_getresvcommport()` when binding to a reserved (privileged) port, a port number in the range 0-1023.) Both functions call `clua_registerservice()` to automatically set the `CLUASRV_MULTI` (`in_multi`) attribute on the port.

Use the `clusvc_getcommport()` and `clusvc_getresvcommport()` functions in the following circumstances:

- The RPC service does not use a well-known port (services that use well-known ports can have `in_multi` entries in `/etc/clua_services`).

- Multiple instances of the RPC service will run in a cluster.
- Requests for the RPC service will be directed to a cluster alias, which will provide load balancing among the instances of the service.

These two functions make it possible to run an RPC application accessed via a cluster alias on multiple cluster members. In addition to ensuring that each instance of an RPC application uses the same common port, the functions also inform the `portmap` daemon that the application is a multi-instance, alias application.

If you do not use one of these functions to bind to the port, you can still run multiple instances of the application, but only one instance will receive requests directed to a cluster alias.

6.13 ifconfig Aliases and Cluster Aliases

Before TruCluster Server Version 5.0, TruCluster products used the `asemgr` command to control application failover. The `asemgr` command ran the `ifconfig` command to create IP aliases as needed. Because the cluster alias subsystem creates and manages aliases on a clusterwide basis, there is no longer any need to explicitly establish and remove IP aliases with `ifconfig` when an application fails over.

Cluster alias addresses are not designed with a one-IP-address-one-service mindset. A cluster alias is an address that encompasses the cluster as a whole (or whatever subset of the cluster chooses to define the particular alias), with the design center being applications that can run multiple copies concurrently (multi-instance services). When single-instance services are necessary, they are best configured with CAA so that failover of the service can be more easily managed.

If you are familiar with ASE services, you can continue to define application-specific interface alias addresses in CAA scripts using `ifconfig alias`. These aliases are independent of cluster aliases and do not create any conflicts.

The advantage of using the default cluster alias is that you do not need to migrate an application's address when moving the application within the cluster, because all applications are using the same address (the default cluster alias) and the cluster alias code will always find where the application is running in the cluster. Furthermore, if an application can run multi-instance (concurrently on multiple members for enhanced scaling), all instances can be transparently accessed using the same cluster alias without the client knowing multiple members are involved.

One advantage of using an ASE-style per-service interface alias (defined in the scripts and migrated with the service by the script) is that traffic is always routed directly to the node running the service (with the cluster alias, traffic often takes one hop within the cluster). Whether this hop outweighs defining an address for each service and migrating it manually depends on the application's throughput needs.

7

Cluster Interconnect

A cluster must have a dedicated cluster interconnect to which all cluster members are connected. This interconnect serves as a private communication channel between cluster members. For hardware, the cluster interconnect can use either Memory Channel or a private local area network (LAN), but not both.

This chapter describes the purpose of a cluster interconnect, its uses, methods of controlling traffic over it, and how to decide what kind of interconnect to use. The chapter discusses the following topics:

- Controlling storage traffic across the cluster interconnect (Section 7.1)
- Controlling application traffic across the cluster interconnect (Section 7.2)
- Controlling cluster alias traffic across the cluster interconnect (Section 7.3)
- Understanding the effect of cluster size and cluster interconnect traffic (Section 7.4)
- Selecting a cluster interconnect (Section 7.5)
- Understanding the Memory Channel interconnect (Section 7.6)
- Understanding the LAN interconnect (Section 7.7)

In general, the cluster interconnect is used for the following high-level functions:

- Health, status, and synchronization messages

  The connection manager uses these messages to monitor the state of the cluster and its members and to coordinate membership and application placement within the cluster. This type of message traffic increases during membership transitions (for example, when a member joins or leaves the cluster), but is minimal in a steady-state cluster. (See Section 7.4 for additional information.)

- Distributed lock manager (DLM) messages

  TruCluster Server uses the DLM to coordinate access to shared resources. User applications can also use this coordination function through the DLM application programming interface (API) library. The message traffic required to coordinate these locking functions is transmitted over the cluster interconnect. Although an application can make heavy use of this capability, the DLM traffic created by the cluster software itself is minimal.

- Accessing remote file systems

  TruCluster Server software presents a uniform picture of the availability of storage devices across all cluster members. Storage located on one member's private storage bus is visible to all cluster members. Reads and writes from other cluster members to file systems on this storage are transmitted by means of the cluster interconnect. Whenever possible, I/O requests (reads, in particular) to files on shared storage are sent directly to the storage and bypass the cluster interconnect. How file systems and storage are configured within the cluster can significantly impact the throughput requirements placed on the cluster interconnect. (See Section 7.1 for additional information.)

- Application-specific traffic

  The cluster interconnect has a TCP/IP address associated with a virtual network interface (`ics0`) on each member. User applications can use this address to communicate over the interconnect. The load that this traffic places on the interconnect varies with the application mix. (See Section 7.2 for additional information.)

- Cluster alias routing

  Although a cluster alias presents a single TCP/IP address that clients use to reference all or a subset of cluster members, the alias subsystem balances establishes each TCP/IP connection to a process on a given cluster member. For example, while multiple simultaneous network file system (NFS) operations to a cluster alias are balanced across cluster members, each individual NFS operation directed at the cluster alias is served by an NFS daemon on one member. The cluster interconnect is used when it is necessary to route the TCP/IP packets addressed to the cluster alias to the specific member that is hosting the connection. The bandwidth requirements that the cluster alias places on the interconnect depend upon the degree to which the cluster alias is being used. (See Section 7.3 for additional information.)

Considering these high-level uses, the communications load of the cluster interconnect can be seen as being heavily influenced both by the cluster's storage configuration and by the set of applications the cluster runs.

Table 7–1 compares a LAN interconnect and a Memory Channel interconnect with respect to cost, performance, size, distance between members, support of the Memory Channel application programming interface (API) library, and redundancy. Subsequent sections discuss how to manage cluster interconnect bandwidth and make an appropriate choice of interconnect based on several factors.

Table 7–1: Comparison of Memory Channel and LAN Interconnect Characteristics

Memory Channel	LAN
Higher cost	Generally lower cost
High bandwidth, low latency	Medium bandwidth, medium to high latency (100 Mb/s); high bandwidth, medium to high latency (1000 Mb/s).
Up to eight members, limited by the capacity of the Memory Channel hub	Up to eight members initially; will support more in the future
Up to 20 meters (65.6 feet) between members with copper cable; up to 2000 meters (1.2 miles) with fiber-optic cable in virtual hub mode; up to 6000 meters (3.7 miles) with fiber-optic cable using a physical hub	The length of a network segment is determined by the capabilities of and options allowed for the Ethernet hardware that is being used. Using the requirements and configuration guidelines for LAN interconnect hardware discussed in the *Cluster Hardware Configuration* manual, see the individual network adapter's *QuickSpecs* at **`http://www.compaq.com/quickspecs`** for this information.
Supports the use of the Memory Channel application programming interface (API) library	Does not support the Memory Channel API library. Some applications may find the general mechanism, introduced in TruCluster Server Version 5.1B, for sending signals from one cluster member to another (clusterwide `kill`) sufficient for communicating between members.
Multirail (failover pair) redundant Memory Channel configuration	Redundancy by configuring multiple network adapters as a redundant array of independent network adapters (NetRAIN) virtual interface on each member, distributing their connections across multiple switches

7.1 Controlling Storage Traffic Across the Cluster Interconnect

The cluster file system (CFS) coordinates accesses to file systems across the cluster by designating a cluster member as the CFS server for a given file system. The CFS server performs all accesses, reads or writes, to that file system on behalf of all cluster members.

Starting in TruCluster Server Version 5.1A, read accesses to a given file system can bypass the CFS server and go directly to the disk, thus not having to pass over the cluster interconnect. If all storage in the cluster is equally accessible from all cluster members, this feature minimizes the bandwidth read operations require of the cluster interconnect. Although some read accesses can bypass the interconnect, all non-direct-I/O write accesses to a file system served by another member must pass through the interconnect. To mitigate this traffic, we recommend that, where possible, applications that write large quantities of data to a file system be located on the same member that is the CFS server for that file system. Given these recommendations, the file system I/O that must traverse the interconnect is limited to remote writes. Understanding the application mix, the CFS server placement, and the volume of data that will be remotely written, can help you determine the most appropriate interconnect for the cluster.

An application, such as Oracle Parallel Server (OPS), can avoid traversing the cluster interconnect to the CFS server by having its disk writes sent directly to disk. This direct-I/O method (enabled by the application's specifying the `O_DIRECTIO` flag on a file open) asserts to CFS that the application is coordinating its own writes to this file across the entire cluster. Applications that use this feature can both increase their clusterwide write throughput to the specified files and eliminate their remote write traffic from the cluster interconnect.

This method is useful only to those applications, such as OPS, that otherwise cannot obtain the performance benefit of data caching, read-aheads, or asynchronous writes. Application developers considering using this flag must be very careful, however. Setting this flag means that the operating system will not apply its normal write synchronization functions to this file for the duration of it being opened by the application. If the application does not perform its own cache management, locking, and asynchronous I/Os, severe performance degradation and data corruption can ensue.

See the *Cluster Administration* manual for additional information on the use of the cluster interconnect by CFS and the device request dispatcher and on the optimizations provided by the direct-I/O feature.

7.2 Controlling Application Traffic Across the Cluster Interconnect

Applications use a cluster's compute resources in different ways. In some clusters, members can be considered as separate islands of computing that share a common storage and management environment (for example, a timesharing cluster in which users are running their own programs on one system). Other applications, such as OPS, use distributed processing to focus the compute power of all cluster members onto a single clusterwide application. In this case, you need to understand how the distributed application's components communicate:

- Do they communicate information by means of shared disk files?
- Do they communicate through direct process-to-process communications over the interconnect?
- How often do these pieces communicate and how much data is transferred per unit of time?
- What does the application require in terms of transmission latency?

With the answers to these questions, you can map the application's requirements to the characteristics of the interconnect options. For example, an application that requires only 10,000 bytes per second of coordination messaging can fully utilize the compute resources of even a large cluster without stressing a LAN interconnect. On the other hand, distributed applications with high data rate and low latency requirements, such as OPS, benefit from having a Memory Channel as the interconnect, even in smaller clusters.

7.3 Controlling Cluster Alias Traffic Across the Cluster Interconnect

The mix of applications that will use a cluster alias, the amount of data being sent to the cluster via the cluster aliases, and the cluster network topology (for example, are members symmetrically or asymmetrically connected to the external networks?) are important factors to consider when deciding which type of cluster interconnect is appropriate.

Some common uses for the cluster alias (such as `telnet`, `ftp`, and Web hosting) typically make only small communication demands of the interconnect. For such applications, the amount of data sent to the cluster's alias is generally far outweighed by the amount of data returned to clients from the cluster. Only the incoming data packets might need to traverse the interconnect to reach the process serving the request. All outgoing packets go directly to the external network and thus do not need to be conveyed over the interconnect. (This presumes that all members have connectivity to the external network.) Applications like these, in most cases, place low bandwidth requirements on the interconnect.

The network file system (NFS), on the other hand, is a commonly used application that can place a significant bandwidth requirement on the cluster interconnect. While reads from the served disks do not cause much interconnect traffic (only the read request itself potentially traverses the interconnect), disk writes through NFS can create interconnect traffic. In this case, the incoming data that might need to be delivered over the interconnect is comprised of disk blocks. If the cluster is going to serve NFS volumes, compare the average rate that disk writes are likely to occur with the bandwidth offered by the various interconnect options.

TruCluster Server Version 5.1B introduces a feature that can lessen the impact of NFS writes. For the purposes of NFS serving, you can assign alternate cluster aliases to subsets of cluster members. This allows a selected set of cluster members to be identified as the NFS servers, thus lowering the average number of inbound packets that must be sent over the interconnect to reach that connection's serving process. (In a randomly distributed four-member cluster, an average of 75 percent of the disk writes will traverse the interconnect. If two of those members are assigned an alternate cluster alias for their NFS serving, the average number of writes traversing the interconnect drops to 50 percent.)

See the *Cluster Administration* manual for information on how to use and tune a cluster alias.

7.4 Effect of Cluster Size on Cluster Interconnect Traffic

You cannot consider solely the number or size of the members in a cluster when determining the most appropriate interconnect, but must also look at how the cluster's use will affect the load placed on the interconnect. Although larger clusters tend to have higher data transfer requirements for a given application mix, how the cluster's storage is configured and the characteristics of its applications are better guides to determining the proper interconnect. However, one aspect of cluster size can impact the interconnect bandwidth requirements. Presuming a perfectly random (and unmanaged) distribution of work across the cluster and an equally random distribution of CFS servers, the percentage of disk writes that must traverse the cluster interconnect increases as the cluster size increases. In a two-member cluster, for example, 50 percent of the average writes might go over the interconnect. In a four-node cluster, this increases to 75 percent. In Section 7.1 we recommend the system that will be performing most writes to a file system be the CFS server for that file system. This recommendation minimizes the number of writes that must be sent over the interconnect and is appropriate regardless of which type of interconnect is used. To the degree that you can meet this recommendation, the less interconnect bandwidth the disk writes will require.

However, there is one situation in which the size of the cluster (measured both in terms of the number of members and the number of disks in use) has a direct impact on the interconnect traffic: cluster membership transitions. In particular, when a member leaves the cluster, the remaining members must pass coordination messages to the other cluster members. Due to the lower latency characteristics of the Memory Channel interconnect, these transitions can be completed faster on a Memory Channel-based cluster. When deciding which interconnect to use, consider how often you expect membership transitions to occur (for example, whether cluster members will routinely be rebooted).

7.5 Selecting a Cluster Interconnect

In addition to the recommendations provided in the previous sections, the following rules and restrictions apply to the selection of a cluster interconnect:

- All cluster members must be configured either to use a LAN interconnect or to use Memory Channel. You cannot mix interconnect types within a cluster.
- Applications using the Memory Channel API library require Memory Channel. A cluster using a LAN interconnect can also be configured with a Memory Channel that is used by Memory Channel API applications only. Use of the Memory Channel API also generates some slight TCP/IP traffic over the cluster interconnect.
- A Fast Ethernet (100Base-T) LAN interconnect is required when configuring one or more AlphaServer DS10L systems in a cluster. An AlphaServer DS10L system is shipped with two 10/100 Mb/s Ethernet ports, one 64-bit peripheral component interconnect (PCI) expansion slot, and a fixed internal integrated device electronic (IDE) disk. When you configure an AlphaServer DS10L in a cluster, we recommend that you use the single PCI expansion slot for the shared storage (where the cluster root, member boot disks, and optional quorum disk reside), one Ethernet port for the external network, and the other 10/100 Mb/s Ethernet port for the LAN interconnect.
- Replacing a Memory Channel interconnect with a LAN interconnect (or vice versa) requires some cluster downtime. Similarly, replacing a Fast Ethernet LAN interconnect with a Gigabit Ethernet LAN interconnect (or vice versa) requires cluster downtime.
- Although the Logical Storage Manager (LSM) provides for transparent mirroring and highly available access to storage, LSM is not a suitable data replication technology in an extended cluster. Although a disaster-tolerant configuration using a LAN-based or Memory Channel-based interconnect and LSM is not supported, there are supported configurations using the StorageWorks Data Replication Manager (DRM) solution.

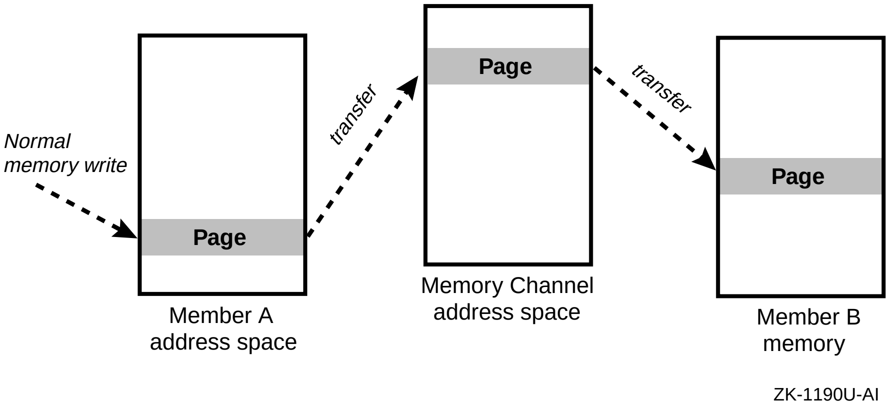
7.6 Memory Channel Interconnect

The Memory Channel interconnect is a specialized interconnect designed specifically for the needs of clusters. This interconnect provides both broadcast and point-to-point connections between cluster members. The Memory Channel interconnect:

- Allows a cluster member to set up a high-performance, memory-mapped connection to other cluster members. These other cluster members can, in turn, map transfers from the Memory Channel interconnect directly into their memory. A cluster member can thus obtain a write-only window into the memory of other cluster systems. Normal memory transfers across this connection can be accomplished at extremely low latency (3 to 5 microseconds).
- Has built-in error checking, virtually guaranteeing no undetected errors and allowing software error detection mechanisms, such as checksums, to be eliminated. The detected error rate is very low (on the order of one error per year per connection).
- Supports high-performance mutual exclusion locking (by means of spinlocks) for synchronized resource control among cooperating applications.

Figure 7–1 shows the general flow of a Memory Channel transfer.

Figure 7–1: Memory Channel Logical Diagram



A Memory Channel adapter must be installed in a PCI slot on each member system. A link cable connects the adapters. If the cluster contains more than two members, a Memory Channel hub is also required.

A redundant, multirail Memory Channel configuration can further improve reliability and availability. It requires a second Memory Channel adapter in each cluster member, and link cables to connect the adapters. A second Memory Channel hub is required for clusters containing more than two members.

The Memory Channel multirail model operates on the concept of physical rails and logical rails. A physical rail is defined as a Memory Channel hub with its cables and Memory Channel adapters and the Memory Channel driver for the adapters on each node. A logical rail is made up of one or two physical rails.

A cluster can have one or more logical rails, up to a maximum of four. Logical rails can be configured in the following styles:

- Single-rail
- Failover pair

If a cluster is configured in the single-rail style, there is a one-to-one relationship between physical rails and logical rails. This configuration has no failover properties; if the physical rail fails, the logical rail fails. Its primary use is for high-performance computing applications using the Memory Channel application programming interface (API) library and not for highly available applications.

If a cluster is configured in the failover pair style, a logical rail consists of two physical rails, with one physical rail active and the other inactive. If the active physical rail fails, a failover takes place and the inactive physical rail is used, allowing the logical rail to remain active after the failover. This failover is transparent to the user. The failover pair style is the default for all multirail configurations.

A cluster fails over from one Memory Channel interconnect to another if a configured and available secondary Memory Channel interconnect exists on all member systems, and if one of the following situations occurs in the primary interconnect:

- More than 10 errors are logged within 1 minute.
- A link cable is disconnected.
- The hub is turned off.

After the failover completes, the secondary Memory Channel interconnect becomes the primary interconnect. Another interconnect failover cannot occur until you fix the problem with the interconnect that was originally the primary.

If more than 10 Memory Channel errors occur on any member system within a 1-minute interval, the Memory Channel error recovery code attempts to determine whether a secondary Memory Channel interconnect has been configured on the member as follows:

- If a secondary Memory Channel interconnect exists on all member systems, the member system that encountered the error marks the primary Memory Channel interconnect as bad and instructs all member systems (including itself) to fail over to their secondary Memory Channel interconnect.

- If any member system does not have a secondary Memory Channel interconnect configured and available, the member system that encountered the error displays a message indicating that it has exceeded the Memory Channel hardware error limit and panics.

See the *Cluster Hardware Configuration* manual for information on how to configure the Memory Channel interconnect in a cluster.

The Memory Channel API library implements highly efficient memory sharing between Memory Channel API cluster members, with automatic error handling, locking, and UNIX style protections. See the *Cluster Highly Available Applications* manual for a discussion of the Memory Channel API library.

7.7 LAN Interconnect

Any Ethernet adapter, switch, or hub that works in a standard LAN at 100 Mb/s or 1000 Mb/s probably will work within a LAN interconnect.

Note

Fiber Distributed Data Interface (FDDI), ATM LAN Emulation (LANE), and 10 Mb/s Ethernet are not supported in a LAN interconnect.

The following features are required of Ethernet hardware participating in a cluster LAN interconnect:

- The LAN interconnect must be private to cluster members. A packet that is transmitted by one cluster member's LAN interconnect adapter can be received only by other members' LAN interconnect adapters.
- A LAN interconnect can be a single direct full-duplex connection between two cluster members or can employ either switches or hubs (but not both). One or more switches are required for a cluster of three or more members and for a cluster whose members use a redundant array of independent network adapters (NetRAIN) virtual interface for their cluster interconnect device.

Note

Although hubs and switches are interchangeable in most LAN interconnect configurations, switches are recommended for performance and scalability. Because hubs run in half-duplex mode, their use in a LAN interconnect may limit cluster performance. Additionally, hubs do not provide the features required for a dual redundant LAN interconnect configuration. Overall, using a switch, rather than a hub, in a LAN interconnect provides greater scalability for clusters with three or more members.

- Adapters and switch ports must be configured compatibly for 100 Mb/s or 1000 Mb/s full-duplex operation.

  If you are using a switch with any of the DE60*x* family of adapters (which have a console name of the form `eix0`) or a DEGPA-*xx* adapter, use a switch that supports autonegotiation. If you are using a switch with network adapters in the DE50*x* family (which have a console name of the form `ewx0`) that do not autonegotiate properly, the switch must be capable of disabling autonegotiation.

- If you use two crossover cables to link two switches in a fully redundant LAN cluster interconnect, you must configure the switches to avoid packet-forwarding problems caused by the routing loop created by the second link. Typical switches provide at least one of the following three mechanisms for support of parallel interswitch links. In order of decreasing desirability for cluster configurations, the mechanisms are:

  Link aggregation — Treats multiple physical links between a pair of switches as a single link and distributes packet traffic among them.

  Link resiliency — Treats multiple physical links between a pair of switches as an active link and one or more standby links and fails over between them.

  Spanning Tree Protocol — Employs a distributed routing algorithm that allows switches to cooperate to discover and remove routing loops.

- Although it may be used to eliminate routing loops on switch ports used for parallel links between switches, Spanning Tree Protocol (STP) must be disabled on all Ethernet switch ports connected to cluster members, whether the members are using single adapters or multiple adapters included in NetRAIN devices. If this is not the case, cluster members will be flooded by broadcast messages which, in effect, create denial-of-service symptoms in the cluster.

- All cluster members must have at least one point-to-point connection to all other members. If the Ethernet adapters that are used for the LAN interconnect fail on a given member, that member loses communication with all other members. A cluster interconnect configuration that requires a member to route interconnect traffic from another member to a different subnet is unsupported. That is, you cannot replace a switch with a member system.

- Up to two switches are allowed between two cluster members. You must not introduce unacceptable latencies by using, for example, a satellite uplink or a wide area network (WAN) as the path between two components of a LAN interconnect.
- Link aggregation of Ethernet adapters using Tru64 UNIX features (including the `lagconfig` command) is not supported for a LAN interconnect.
- To simplify management, configure the LAN interconnect network adapters symmetrically on all cluster members. Installing the same type of adapter in each member in the same relative position with respect to other network adapters helps ensure that the adapters have similar names across cluster members. In a fully redundant LAN interconnect configuration using two or more interconnected switches, and NetRAIN virtual interfaces as member interconnect devices, you should uniformly connect the first network adapter listed in each member's NetRAIN set to the first switch and the second network adapter to the second switch. This simplifies the identification of the adapters for monitoring and maintenance. Additionally, it ensures that the active adapters of each member are connected to the same switch when the cluster is initially booted. One method for guarding against a network partition of the cluster in certain failure conditions is to ensure that all active adapters in the LAN interconnect are connected to the same switch. See the *Cluster Hardware Configuration* manual for additional information.

8 Distributed Lock Manager

The distributed lock manager (DLM) provides functions that allow cooperating processes in a cluster to synchronize access to a shared resource, such as a raw disk device or a program. For the DLM to effectively synchronize access to a shared resource, all processes in the cluster that share the resource must use DLM functions to control access to the resource. For example, a distributed database application might use lock manager services to coordinate access to the shared disks participating in a database.

An application secures a lock on a named shared resource. Resource names can be single-dimensional or tree-structured. You can use a resource tree to create a hierarchy of locks and sublocks that reflect the structure of a shared resource. The DLM supplies functions that:

- Provide mutual exclusion, restricted sharing, and full sharing of data access
- Notify a process holding a lock when its lock is blocking another process's access to a resource
- Notify a process that has queued a lock request to a resource when its request has been granted
- Convert a lock's mode between less restrictive and more restrictive lock modes
- Return information about locks

The DLM employs a distributed, centralized tree design. It does not replicate lock information on each cluster member. Rather, the cluster member that manages a lock tree maintains all information about that tree. The member that holds a given lock is aware of only its contribution of that lock to the resource. Any member system can serve as the master for any lock tree, which distributes the overall lock management load.

The DLM uses a distributed directory service to quickly locate the directory node for a resource tree. A directory table associates a root resource name with the cluster member that is the manager of the resource. This directory table is identical on all cluster members.

The DLM is designed to handle member failures. If a lock holder fails, its locks are released. If a member system fails, a new lock master for locks that were previously mastered on that member is chosen and provided with all pertinent lock information.

9 Cluster Installation and Administration

This chapter provides an overview of cluster installation (Section 9.1) and administration (Section 9.2).

9.1 Installation

TruCluster Server Version 5.1B supports three upgrade paths:

1. A full installation of TruCluster Server Version 5.1B.
2. A **rolling upgrade** from TruCluster Server Version 5.1A. A rolling upgrade is a software upgrade of a cluster that is performed while the cluster is in operation. One member at a time is rolled and returned to operation while the cluster transparently maintains a mixed-version environment for the Tru64 UNIX base operating system, cluster, and Worldwide Language Support (WLS) software.

   To shorten the time required to upgrade a cluster, you can roll multiple members in parallel. The number of members rolled simultaneously is limited only by the requirement that the members not being rolled (Member state = UP) have sufficient votes to maintain quorum. With parallel rolls you can upgrade an eight-member cluster in about the same time it takes to upgrade a four-member cluster by rolling one member at a time.

   The rolling upgrade procedure is used for three major tasks:

   a. Rolling from the prior version of the Tru64 UNIX base operating system and cluster software to the current version.
   b. Rolling patch kits into the cluster.
   c. Rolling New Hardware Delivery (NHD) kits into the Version 5.1B cluster.

   Rolling in a patch kit or an NHD kit uses the same procedure as rolling in a new release of the base operating system and cluster software. The `clu_upgrade` command controls rolling upgrades. See `clu_upgrade`(8) for a description of the `clu_upgrade` command.
3. Three upgrade procedures for those upgrading from the TruCluster Production Server Software or TruCluster Available Server Software Version 1.5 or Version 1.6 products. Two of these options use scripts

that are specifically designed to facilitate the migration of storage from the old cluster (rz* style device names) to the new cluster (dsk* style device names). The *Cluster Installation* manual also describes an upgrade path for TruCluster Memory Channel Software products that have little or no shared storage.

One major difference when creating a TruCluster Server Version 5.*x* cluster is that you install Tru64 UNIX on only one system in the cluster. Because CFS creates shared clusterwide file systems, after a cluster is created, additional members boot into the cluster and have access to these files. (Before TruCluster Server Version 5.0, you had to install the base operating system on all cluster members, and there were no clusterwide file systems.)

For TruCluster Server, the initial creation of a cluster, the adding of members, and the removing of members are accomplished through three interactive installation scripts: clu_create, clu_add_member, and clu_delete_member. The scripts provide online help and write log files to the /cluster/admin directory.

The following list outlines the steps needed to install and create a new TruCluster Server cluster:

1. Using the information in the *Cluster Hardware Configuration* manual, configure the system and storage hardware and firmware.
2. Using AdvFS file systems, install Tru64 UNIX on a private disk on the system that will become the first cluster member.
3. Configure the Tru64 UNIX system, including network and time services. Load and configure the applications that you plan to use in the cluster.
4. Load the TruCluster Server license and software.

Notes

Each cluster member must have both a Tru64 UNIX license and a TruCluster Server license.

If there are any patch or NHD kits available, you can install them after loading the cluster software but before running clu_create. Installing them before running clu_create means that you do not have to roll them into the cluster later.

5. Run the clu_create command to create the boot disk for the first cluster member, and to create and populate the clusterwide root (/), /usr, and /var AdvFS file systems.
6. Halt the Tru64 UNIX system and boot the disk containing the first member’s cluster boot partition. As the system boots, it forms a

single-member cluster and mounts the clusterwide root (/), /usr, and /var file systems.

7. Log in to the single-member cluster and run the clu_add_member command to add members to the cluster. Boot each new member before adding the next.

See the *Cluster Installation* manual for more information on installing TruCluster Server.

9.2 Administration

Having a clusterwide file namespace greatly simplifies cluster management. A cluster has just one copy of most system configuration files. For example, a cluster is managed as a single security domain through one /etc/group file and one /etc/passwd file.

User access to files is independent of which node a user is logged in on, and which node is serving the file. File permissions and access control lists (ACLs) are uniform across the cluster.

Audit logs are kept in a common location; each member's host name is appended to its log files to avoid confusion when tracking audit events.

In most cases, the fact that you are administering a cluster rather than a single system becomes apparent because of the occasional need to manage one of the following aspects of a TruCluster Server environment. In the following list, each area of administration is followed by one or more of the cluster-specific commands used to manage or monitor it. (You can use the SysMan Menu and SysMan Station GUIs to perform most command-line functions; you must use the cluster installation commands to install a cluster.)

- Cluster creation, which supports installing the initial cluster member, adding and deleting members, and querying the cluster configuration (clu_create, clu_add_member, clu_delete_member, and clu_check_config).
- Cluster application availability (CAA), which lets you define and manage highly available applications (caa_profile, caa_register, caa_unregister, caa_start, caa_stop, caa_relocate, and caa_stat).
- Cluster aliases, which provide a single system view from the network (cluamgr).
- Cluster quorum and votes, which determine what constitutes a valid cluster and membership in that cluster, and thereby control access to cluster resources (clu_quorum).

- Optional load-balancing of CFS servers (`cfsmgr`).
- Optional load-balancing of the device request dispatcher subsystem (`drdmgr`).

In addition to the previous items, there are some command-level exceptions to the Single System Image (SSI) model. (SSI means that, when possible, the cluster appears to the user like a single computer system.) For example, when you execute the `wall` command, the message is sent only to users who are logged in on the cluster member where the command executes. To send a message to all users who are logged in on all cluster members, use the `wall -c` command. The same logic applies to the `shutdown` command; you can shut down an individual member or the entire cluster.

See the *Cluster Administration* manual for more information on configuring and managing a TruCluster Server cluster.

Glossary

The terms in this glossary are commonly used in a TruCluster Server environment.

A

action script

Shell scripts used by CAA to control how applications are started, stopped, and checked. By default, action scripts are located in the `/var/cluster/caa/script` directory. The file names of action scripts take the form *`resource_name`*`.scr`.

adapter

A device that converts the protocol and hardware interface of one bus type into that of another bus.

address switches

Electrical switches on some disk drives that determine the SCSI address setting for the drive.

alias router

A cluster member that makes a cluster alias address known to the network and receives incoming packets for that alias. By default, all cluster members are configured as alias routers at boot time.

availability

The characteristic of a computing system that allows it to provide computing services (such as applications) to clients with little or no disruption.

See also *highly available*

B

bus

Flat or twisted-wire cable or a backplane composed of individual parallel circuits. A bus connects computer system components to provide communications paths for addresses, data, and control information.

C

CAA

cluster application availability. A subsystem that provides high availability for single-instance applications and monitoring of the state of other types of resources (such as network interfaces). A single instance of any application that can run on Tru64 UNIX can be made highly available in a cluster with CAA.

CDSL

context-dependent symbolic link. A special form of a symbolic link whose target pathname includes an environment variable, such as `{memb}`, which is resolved at run time. In a cluster, CDSLs make it possible to maintain per-system configuration and data files within the shared CFS root (`/`), `/usr`, and `/var` file systems.

CFS

cluster file system. A virtual file system that sits above the physical file systems and provides clusterwide access (with assistance from the device request dispatcher) to all mounted file systems in a cluster. CFS maintains cache coherency across all cluster members, which ensures that all members have an identical, consistent view of file systems directly connected to the cluster.

client

A computer system that uses resources provided by another computer, called a server.

cluster

A loosely coupled collection of servers that share storage and other resources that make applications and data highly available. A cluster consists of communications media, member systems, peripheral devices, and applications. The systems communicate over a cluster interconnect.

cluster alias

An IP address that is used to address all or a subset of the members in a cluster. A cluster alias makes some or all of the systems in a cluster look like a single system to the outside world.

cluster application availability (CAA)

See *CAA*

cluster expected votes

See *expected votes*

cluster file system (CFS)

See *CFS*

cluster interconnect

A private interconnect that cluster members use for intracluster communication.

cluster member

The basic computing resource in a cluster. A member system must be physically connected to a cluster interconnect.

In common usage, a system configured with TruCluster Server software that is capable of forming or joining a cluster. From the point of view of the connection manager, a system that has either formed a single-member cluster or has been granted membership in an existing cluster. The connection manager dynamically determines cluster membership based on communications among the cluster members. Only an active cluster member can access the shared resources of a cluster.

cluster partition

An abnormal condition in which nodes in an existing cluster divide into multiple independent clusters.

common subnet

In the context of cluster aliases, an existing physical subnet. Cluster alias IP addresses are either in a common subnet or in a virtual subnet.

connection manager

The cluster software component that coordinates participation of systems in the cluster, and maintains cluster integrity when systems join or leave the cluster.

context-dependent symbolic link (CDSL)

See *CDSL*

current votes

The number of votes contributed by current cluster members and by the quorum disk as seen by this member.

D

dedicated port

See *locked port*

default cluster alias

A special cluster alias created during cluster installation. All cluster members are, by default, members of the default cluster alias.

device request dispatcher

A kernel subsystem that controls all I/O access to storage devices in a cluster. The device request dispatcher supports clusterwide access to both character and block disk devices.

Note: Do not confuse the device request dispatcher with the Distributed Raw Disk (DRD) services provided in the TruCluster Production Server product. The device request dispatcher is fully integrated with the kernel, and removes the need for having a specific service to make storage accessible to cluster members.

differential SCSI bus

A SCSI bus where the signal's level is determined by the voltage differential between two wires.

direct-access cached reads

A performance enhancement for AdvFS file systems. Direct-access cached reads allow CFS to read directly from storage simultaneously on behalf of multiple cluster members.

direct-access I/O device

An I/O device that supports simultaneous access from multiple cluster members.

See also *single-server device*

distributed application

An application that is specifically designed to run on a cluster, using different members for specific purposes. These applications use the Memory Channel, distributed lock manager (DLM), and cluster alias application programming interfaces to integrate application with the cluster resources.

distributed lock manager (DLM)

See *DLM*

DLM

distributed lock manager. A software component that synchronizes access to shared resources among cooperating processes throughout the cluster.

E

Event Manager (EVM)

See *EVM*

EVM

Event Manager. A facility that lets kernel-level and user-level processes and components post events, and provides a means for processes to subscribe for notification when selected events occur. The facility provides an event viewer, an API, and command-line utilities. See EVM(5) for more information.

expected votes

The sum of all node votes held by cluster members, plus the vote of the quorum disk, if one is defined.

F

failover

A transfer of the responsibility to provide services. A failover occurs when a hardware or software failure causes a service to restart on another member system.

failover pair

A Memory Channel logical rail configuration that consists of two physical rails, with one physical rail active and the other inactive. If the active physical rail fails, a failover takes place and the inactive physical rail is used.

fast SCSI

An optional mode of SCSI-2 that allows transmission rates of up to 10 MB per second.

file system partitioning

Mounting an AdvFS file system so that it is accessible to only a single cluster member.

File system partitioning is provided to ease migration from TruCluster Production Server Software or TruCluster Available Server Software Version 1.5 or Version 1.6. File system partitioning is not intended as a general purpose method for restricting file system access to a single member.

H

highly available

In the TruCluster Server software, the ability to survive any single hardware or software failure.

A cluster can be considered highly available if the hardware and software provide protection against any single failure, such as a system failure, disk failure, or a SCSI cable disconnection.

A service can be considered highly available if the hardware that it depends on provides protection against any single failure, and the service is configured to fail over in case of a failure.

host route

When discussing routing for cluster aliases, an advertised route (using RIP) to a cluster alias IP address via a local IP address on a cluster node, using a subnet mask of all 1s.

hot swap

The ability to replace a device on a shared bus while the bus is active.

I

in_multi service

A designation on a service port that causes the cluster alias subsystem to route connection requests and packets to all eligible members of the alias.

in_noalias service

A designation on a service port that causes the cluster alias subsystem to ensure that the port will not receive inbound alias messages.

in_nolocal service

A designation on a service port that causes the cluster alias subsystem to ensure that the port will not honor connection requests to a nonalias address.

in_single service

A designation on a service port that causes the cluster alias subsystem to ensure that only one alias member will receive connection requests or packets for that service.

L

local bus

See *private bus*

lock file

A file that indicates that operations on one or more other files are restricted or prohibited. The presence of the lock file can be used as the indication, or the lock file can contain information describing the nature of the restrictions.

locked port

A port in the clusterwide port space that is dedicated for use by a single node in the cluster.

logical rail

One or more Memory Channel physical rails. Logical rails are configured as a single-rail or as a failover pair.

Logical Storage Manager (LSM)

See *LSM*

logical unit number (LUN)

See *LUN*

LSM

Logical Storage Manager. A disk storage management tool that protects against data loss, improves disk I/O performance, and customizes the disk configuration.

System administrators use LSM to perform disk management functions without disrupting users or applications accessing data on those disks.

LSM disk group

A group of Logical Storage Manager (LSM) disks that share a common configuration. The configuration information for an LSM disk group consists of a set of records describing objects including LSM disks, LSM volumes, LSM plexes, and LSM subdisks that are associated with the LSM disk group. Each LSM disk group has an administrator-assigned name that can be used to reference that LSM disk group.

LSM plex

A copy of an LSM volume's logical data address space, which is sometimes known as a mirror. An LSM volume can have up to eight LSM plexes associated with it. A read can be satisfied from any LSM plex, while a write is directed to all LSM plexes.

LSM volume

A special device that contains data used by a UNIX file system, a database, or other applications. LSM transparently places an LSM volume between applications and a physical disk. Applications then operate on the LSM volume rather than on the physical disk. For example, a file system is created on an LSM volume rather than on a physical disk.

An LSM volume presents block and raw interfaces that are compatible in their use with disk partition special devices. Because an LSM volume is a virtual device, it can be mirrored, spanned across disk drives, moved to use different storage, and striped using administrative commands. The configuration of an LSM volume can be changed using LSM utilities without disrupting applications or file systems that are using the LSM volume.

LUN

logical unit number. A physical or virtual peripheral device addressable through a target. LUNs use their target's bus connection to communicate on a SCSI bus.

M

Mb/s

Megabits per second

MB/s

Megabytes per second

member

See *cluster member*

member ID

An integer, in the range 1-63, that identifies a cluster member system. Each member has a unique member ID, which is assigned during the installation procedure.

Memory Channel interconnect

A peripheral component interconnect (PCI) cluster interconnect that provides fast and reliable communications between cluster members. Physically, the interconnect consists of a Memory Channel adapter installed in a PCI slot in each member system, one or more Memory Channel link cables to connect the adapters, and an optional Memory Channel hub.

multi-instance application

An application that can run on multiple cluster members at the same time. A multi-instance application is, by definition, highly available because the failure of one cluster member does not affect the instances of the application running on other members.

N

network route

When discussing routing for cluster aliases, an advertised route (using RIP) to a virtual subnet in which one or more cluster alias IP addresses reside.

node votes

The fixed number of votes that a given member contributes toward quorum.

nonvoting member

A cluster member with 0 (zero) votes.

See also *voting member*

O

out_alias service

A designation on a service port that causes the cluster alias subsystem to ensure that the default cluster alias is used as the source address whenever the port is used as a destination.

P

PCI

peripheral component interconnect. An industry-standard expansion I/O bus that is a synchronous, asymmetrical I/O channel.

peripheral component interconnect (PCI)

See *PCI*

personality module

The module on a storage shelf that provides the interface between a differential SCSI bus and the storage shelf single-ended SCSI bus. Switches on the module enable SCSI bus termination and control SCSI bus IDs for the storage shelf.

physical rail

A Memory Channel hub with its cables and Memory Channel adapters and the Memory Channel driver for the adapters on each node.

See also *logical rail*

placement policy

A policy that determines where an application under CAA control runs. Supported policies are: `balanced`, `favored`, and `restricted`.

private bus

A bus that connects private storage to the local system.

private storage

A storage device on a private bus. Storage devices include hard disks, floppy disks, tape drives, and other devices.

proxy ARP

The mechanism that cluster members use to handle requests that are addressed to cluster aliases whose addresses reside on common subnets. The Address Resolution Protocol (ARP) maps dynamically between IP addresses and Ethernet addresses. An ARP request contains the IP address of an interface on the target host. The host that recognizes this IP address should respond with its Ethernet address. All other hosts should ignore the ARP request.

Proxy ARP is, in essence, when a system or router lies about being the system with an interface that matches the IP address in the ARP request. The proxy ARP system responds to the ARP request by returning its own Ethernet address. The system then routes the packets to the real target system. Proxy ARP is useful for subnetting and also when adding routers to a topology where some hosts are not yet configured to use the routers.

The cluster member that is acting as the proxy ARP master for an alias responds to an ARP request for a cluster alias IP address using one of its own network interfaces.

Q

quorum

A cluster state in which members are allowed to access clusterwide shared resources and thus perform useful work. The cluster has quorum when the connection manager determines that the sum of the node votes and quorum disk votes in the cluster equals or exceeds the required number of quorum votes.

See also *quorum votes*

quorum algorithm

A mathematical method that the connection manager uses to determine the circumstances under which a given member can participate in a cluster, safely accessing clusterwide resources and performing useful work.

quorum disk

A disk whose `h` partition contains cluster status and quorum information. Each cluster can have a maximum of one quorum disk. The quorum disk is assigned votes that are used when calculating quorum.

quorum disk votes

The number of votes that a quorum disk contributes towards quorum.

quorum loss

A cluster state in which no member is allowed to access clusterwide shared resources. A cluster enters a quorum loss state when the connection manager determines that the member and quorum disk votes in the cluster are less than the required number of quorum votes.

See also *quorum votes*

quorum votes

The number of votes that are required to form or maintain a cluster. The formula for calculating quorum votes is:

```
quorum votes = round_down((cluster-expected-votes+2)/2)
```

R

RAID

redundant array of independent disks. A technique that organizes disk data to improve performance and reliability. RAID has three attributes:

- The user views this set of physical disks as a single logical device or multiple logical devices.
- Disk data is distributed across the physical set of drives in a defined manner.
- Redundant disk capacity is added so data can be recovered if a drive fails.

redundant

A term to describe duplicate hardware that provides spare capacity that can be used when a component fails.

redundant array of independent disks (RAID)

See *RAID*

resource

A cluster hardware or software component that provides a service to end users or to other software components. Examples of resources are disks, tapes, file systems, network interfaces, and application software.

resource manager

All the CAA daemons running on cluster members. These daemons are independent but they communicate with each other, sharing information about the status of the resources.

The resource manager communicates with all the components of the CAA subsystem, as well as the connection manager and the Event Manager (EVM). The resource manager also uses the resource monitors to monitor the status of a particular type of resource.

resource monitor

A monitor that is loaded by the resource manager at boot time. There is one resource monitor for each type of resource (application, network, tape, and media changer).

resource profile

A file that contains an application's resource requirements. The file contains keyword/value pairs that are used by CAA to monitor resources and control application failover. Each application under CAA control has a resource profile. Resource profiles are located in the `/var/cluster/caa/profile` directory. The file names of resource profiles take the form *`resource_name`*`.cap`.

RIP

Routing Information Protocol. A protocol that is used to exchange routing information among gateways and other hosts. The protocol was defined in RFC 1058, and updated in RFCs 1388 and 1723.

rolling upgrade

A software upgrade of a cluster that is performed while the cluster is in operation. One member at a time is rolled, or multiple members may be rolled in parallel, and returned to operation while the cluster transparently maintains a mixed-version environment for the base operating system, cluster, and Worldwide Language Support (WLS) software. Clients accessing services are not aware that a rolling upgrade is in progress.

router priority

A method that controls the proxy ARP router selection for a cluster alias on a common subnet. For each alias in a common subnet, the cluster member with the highest router priority for that alias will route for that alias.

Routing Information Protocol (RIP)

See *RIP*

S

SCSI

Small Computer System Interface. A standard, maintained by the American National Standards Institute (ANSI), that provides a standard interface for connecting disks and other peripheral devices to a computer system. SCSI-based devices can be configured in a series, with multiple devices on the same bus.

SCSI-2

An extension to the original SCSI standard featuring multiple systems on the same bus and hot swap. The SCSI-2 standard is ANSI standard X3.T9.2/86-109.

SCSI adapter

A storage adapter, commonly referred to as a host bus adapter (HBA), that provides a connection between an I/O bus and a SCSI bus.

SCSI bus

A bus that supports the transmission and signaling requirements of a SCSI protocol.

SCSI bus speed

The data transfer speed for a SCSI bus. SCSI bus speed can be slow, up to 5 MB/s; fast, up to 10 MB/s; fast and wide, up to 20 MB/s; or UltraSCSI, up to 40 MB/s.

SCSI controller

See *SCSI adapter*

SCSI device

A SCSI adapter, peripheral controller, or intelligent peripheral that can be attached to a SCSI bus.

SCSI ID

A unique address, from 0 through 15, that identifies a device on a SCSI bus.

selection priority

A priority assigned to a cluster alias that determines the order in which members of the alias receive new connection requests. The selection priority establishes a hierarchy within the members of an alias. Connection requests are distributed among those members sharing the highest selection priority value.

selection weight

The number of connections (on average) a member is given before connections are given to the next alias member with the same selection priority value.

shared bus

A bus that is connected to more than one member system and, optionally, to one or more storage devices.

shared storage

Disks that are connected to a shared bus.

signal converter

A device that converts signals between a single-ended SCSI bus and a differential SCSI bus.

single-ended SCSI bus

A signal path in which one data lead and one ground lead are utilized to make a device connection. This transmission method is economical, but is more susceptible to noise than a differential SCSI bus.

single-instance application

An application that is run on only one cluster member at a time. The cluster application availability (CAA) subsystem can provide high availability for single-instance applications by controlling their initial startup and failover characteristics.

single rail

A Memory Channel logical rail configuration where there is a one-to-one relationship between physical rails and logical rails. This configuration has no failover properties; if the physical rail fails, the logical rail fails.

single-server device

A device that supports access from only a single member.

See also *direct-access I/O device*

Small Computer System Interface (SCSI)

See *SCSI*

SRM

The external interface to console firmware for operating systems that expect firmware compliance with the Alpha System Reference Manual (SRM).

standard hub mode

A Memory Channel interconnect configuration that uses a Memory Channel hub to connect Memory Channel adapters. To set up a Memory Channel interconnect in standard mode, use a link cable to connect each Memory Channel adapter to a linecard installed in a Memory Channel hub.

static service

A designation on a service port that causes the cluster alias subsystem to ensure that the port will not be assigned as a dynamic port.

subset

An installable software module that is compatible with the Tru64 UNIX `setld` software installation utility.

T

terminator

A resistor array device that terminates a SCSI bus. A SCSI bus must be terminated at its two physical ends.

trilink connector

A connector that joins two cables to a single device, or allows terminating a shared SCSI bus external to the adapter or to the RAID controller.

tunneling

In the context of cluster aliases, moving an `mbuf` chain between cluster members after receipt.

U

UltraSCSI

A differential SCSI bus standard that uses smaller diameter cables with smaller connectors and allows bus speeds up to 40 MB/s at 25 meters (approximately 82 feet).

UltraSCSI hub

A specialized signal converter with multiple connectors. An UltraSCSI hub converts differential input SCSI signals from a host bus adapter to single-ended, then converts the single-ended signals back to differential for the output connection to a RAID array controller. An UltraSCSI hub allows radial connection of UltraSCSI devices and increases the separation between host and storage.

V

virtual hub mode

A Memory Channel interconnect configuration that does not use a Memory Channel hub to connect Memory Channel adapters. Virtual hub mode is supported only for clusters that have two member systems. To set up a Memory Channel interconnect in virtual hub mode, use a Memory Channel link cable to connect the Memory Channel adapter in one member system to the corresponding Memory Channel adapter in the other member system.

virtual subnet

In the context of cluster aliases, a subnet with no physical connections. Cluster alias IP addresses are either in a common subnet or in a virtual subnet.

vMAC

In the context of cluster aliases, a unique hardware address that can be automatically created for each alias IP address. An alias vMAC (virtual Media Access Control) address follows the cluster alias proxy ARP master from node to node as needed. Regardless of which cluster member is serving as the proxy ARP master for an alias, the alias's vMAC address does not change.

votes

See *quorum*

voting member

A cluster member with a vote.

See also *nonvoting member*

W

worldwide ID (WWID)

See *WWID*

WWID

Worldwide identifier. A unique ID that a manufacturer assigns to a disk.

Y

Y cable

A cable that joins two cables to a single device, or allows terminating a shared SCSI bus external to the adapter or RAID controller.

Index