# TruCluster Server

## Technical Overview

Part Number: AA-RHGVB-TE

**April 2000**

**Product Version:** TruCluster Server Version 5.0A

**Operating System and Version:** Tru64 UNIX Version 5.0A

This document describes the components and features of TruCluster Server Version 5.0A.

**Compaq Computer Corporation**
**Houston, Texas**

# Contents

## Tables

# About This Manual

This manual provides a concise summary of the features available in the TruCluster™ Server Version 5.0A product.

**Note**

The information in this manual does not supersede that found in the TruCluster Server *Software Product Description*, which is the definitive legal description of this product. Please check that document for the latest product information.

## Audience

This manual is for anyone who is interested in a descriptive overview of the TruCluster Server features and functions.

## Organization

This manual contains the following nine chapters and a glossary:

| | |
|---|---|
| *Chapter 1* | Provides an introduction to TruCluster Server. |
| *Chapter 2* | Describes the clusterwide file systems, 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. |
| *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 the TCP and UDP applications. |
| *Chapter 7* | Describes the Memory Channel interconnect, which provides a high-speed connection among all cluster members. |
| *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. |
| *Glossary* | Defines common terms used throughout the TruCluster Server documentation. |

## Related Documents

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

- TruCluster Server *Software Product Description* (SPD) — The legal description of the TruCluster Server Version 5.0A product. You can find the latest version of the SPD and other TruCluster Server documentation at the following URL:

  `http://www.unix.digital.com/faqs/publications/pub_page/cluster_list.html`

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

## Reader's Comments

Compaq 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`

- Mail:

  Compaq Computer Corporation
  UBPG Publications Manager
  ZKO3-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-2698

  A Reader's Comment form is located in the back of each printed manual. The form is postage paid if you mail it in the United States.

Please include the following information along with your comments:

- The full title of the book and the order number. (The order number is printed on the title page of this book and on its back cover.)
- 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 Compaq technical support office. Information provided with the software media explains how to send problem reports to Compaq.

## 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. |
| Mb/s | This symbol indicates megabits per second. |
| MB/s | This symbol indicates megabytes per second. |

# 1

# Introduction to TruCluster Server

TruCluster Server Version 5.0A is a highly integrated synthesis of Tru64 UNIX software, AlphaServer systems, and storage devices that operate as a single system. A TruCluster Server cluster acts as a single virtual system, even though it is made up of multiple systems. 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.

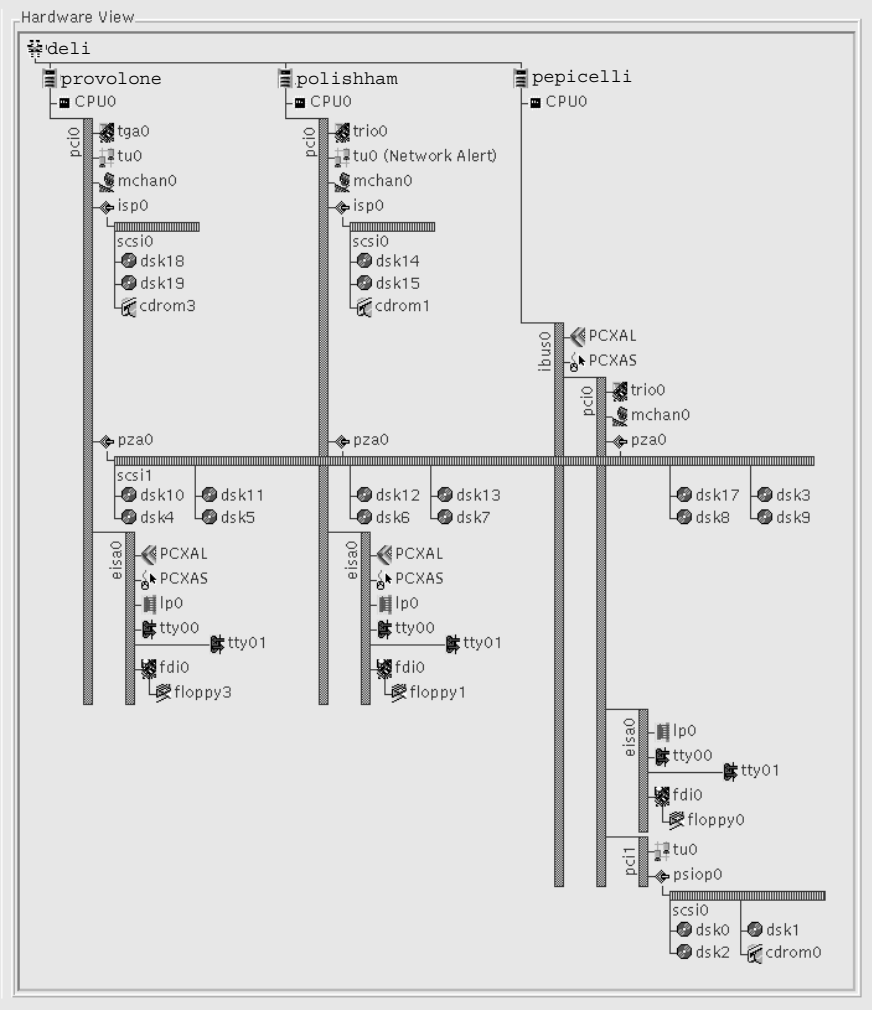
A TruCluster Server environment can be as simple or as feature-rich as you require. You configure a cluster that fits your needs, from a two-node cluster up to an eight-node cluster running high availability applications such as transaction processing systems, servers for network client/server applications, 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.

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 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 1–1 shows the SysMan Station hardware view for a cluster named `deli` with three members: `provolone`, `polishham`, and `pepicelli`.

**Figure 1–1: A Cluster's View of Hardware**




TruCluster Server preserves the following availability and performance features found in 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 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.

TruCluster Server Version 5.0A provides the features listed in Table 1–1.

**Table 1–1: Features in the TruCluster Server Version 5.0A 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.<br>See Section 2.2 for more information on CFS. See Section 2.4 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.<br>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.<br>See Section 2.7 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.<br>See Chapter 3 for more information on the connection manager. |

**Table 1–1: Features in the TruCluster Server Version 5.0A 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.<br>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.<br>In Version 5.0A, CAA introduces support for tape devices and media changers as monitored resources. CAA also provides additional features for existing resource types, such as application monitoring through the `check` entry point of a service's action script. CAA also supports additional profile attributes, including restart attempts, failover delay, failover threshold and interval, autostart, and active placement. |
| 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 alias is defined that addresses all cluster members. A site can define additional aliases that address some or all cluster members.<br>See Chapter 6 for more information on cluster aliases.<br>For Version 5.0A, a virtual Media Access Control (vMAC) address can be assigned to an alias. When vMAC support is enabled, an alias vMAC address follows the alias's 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. For more information on vMAC, see Section 6.7.<br>Cluster alias has also been enhanced in the way in which it handles ICMP packets. Other changes include enhancements in loopback performance and a reduction in the use of distributed lock manager (DLM) locks for single-instance service selection. |
| Highly available 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. Clients use the default cluster alias as the name of the NFS server when mounting file systems exported by the cluster.<br>See Section 6.3 for more information. |

**Table 1–1: Features in the TruCluster Server Version 5.0A Product (cont.)**

| Feature | Description |
|---|---|
| Memory Channel interconnect | The Memory Channel interconnect is a high-speed interconnect designed specifically for the needs of clusters. The Memory Channel interconnect provides both broadcast and point-to-point connections between cluster members.<br>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.<br>See Chapter 7 for more information on the Memory Channel interconnect. See the TruCluster Server *Highly Available Applications* manual for a description of the Memory Channel API. |
| Internode Communication Subsystem (ICS) optimized for Memory Channel | In Version 5.0A, the Internode Communication Subsystem (ICS) Memory Channel Transport (MCT) provides higher performance, better scalability, and quicker failure detection than the previous TruCluster Server implementation of ICS that used TCP/IP over Memory Channel. ICS MCT takes full advantage of the features offered by Memory Channel for optimized data transfer and copy avoidance. It also takes advantage of Memory Channel features for failure detection rather than relying on TCP/IP timeouts.<br>In addition, the ICSNET network driver provides a network interface for the Memory Channel. Because the driver uses ICS to communicate with other cluster members, it is interconnect-independent. |
| 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.<br>See Chapter 8 for a description of the DLM. See the TruCluster Server *Highly Available Applications* manual for a description of the DLM API. |
| Single-system management | Because a cluster uses CFS, all 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.<br>See Chapter 9 for an overview of cluster installation and administration. |
| Rolling Upgrade/Patch | TruCluster Server Version 5.0A contains the software infrastructure required to support rolling upgrades and patches. Customers who install TruCluster Server Version 5.0A will be able to perform a rolling upgrade to subsequent TruCluster Server releases, and roll patches onto a Version 5.0A cluster. |

**Table 1–1: Features in the TruCluster Server Version 5.0A Product (cont.)**

| Feature | Description |
|---|---|
| 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 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. |
| 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 new 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.

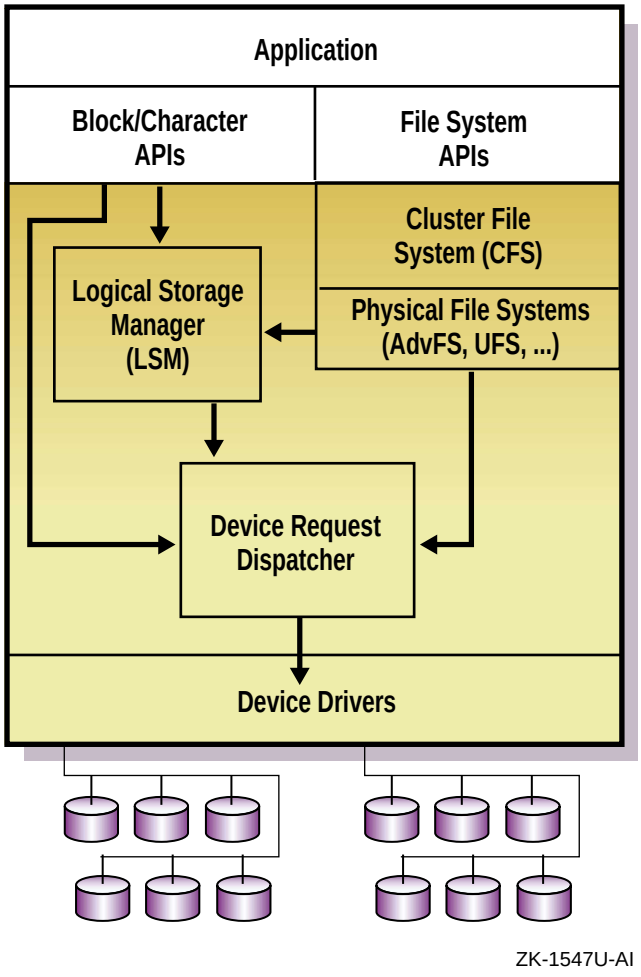
The clusterwide namespace is implemented by several new TruCluster Server technologies, including the cluster file system (CFS) and the device request dispatcher, both of which are described in this chapter.

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)
- Context-dependent symbolic links (CDSLs) (Section 2.4)
- Device names (Section 2.5)
- Worldwide ID (Section 2.6)
- Clusters and the Logical Storage Manager (LSM) (Section 2.7)

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. One important thing to note in this figure is that the device request dispatcher controls all I/O to physical devices; all cluster I/O passes through this subsystem. You should also note that CFS layers on top of existing file systems such as the Advanced File System (AdvFS).

**Figure 2–1: Storage Software Layering in a Cluster**




## 2.1 Supported File Systems

Table 2–1 summarizes how TruCluster Server supports different UNIX file systems.

**Table 2–1: UNIX File Systems Supported in a Cluster**

| Type | How Supported | Failure Characteristics |
|---|---|---|
| Advanced File System (AdvFS) | Read/write | A file domain is served by the member that first mounts it. 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. |
| Network File System (NFS) server | Read/write | External clients use the default cluster alias 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 | When an file system that has been NFS-mounted in a cluster fails, the file system is automatically unmounted. The client must remount the file system to make it available. If the client uses `automount`, the remount will happen automatically. |
| UNIX File System (UFS) | Read-only | A file system is served for read-only access by the member that first mounts it. Upon member or path failure, CFS selects a new server for the file system. Upon path failure, CFS uses an alternate device request dispatcher path to the storage. |
| CD-ROM File System (CDFS) | Read-only | A file system is served for read-only access by the member that mounts the CD-ROM 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. |
| PC-NFS server | Read/write | PC clients use the default cluster alias as the host name when mounting file systems NFS-exported by the cluster. File system failover and recovery is transparent to external NFS clients. |
| Memory File System (MFS) | Not supported | |
| `/proc` file system | Read/write (local) | Each cluster member has its own `/proc` file system, which is accessible only by that member. |

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

| Type | How Supported | Failure Characteristics |
|---|---|---|
| File-on-File Mounting (FFM) file system | Read/write (local) | Can be mounted and accessed only on the local member. |
| Named pipes | Read/write (local) | Reader and writer must be on the same member. |

## 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 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.

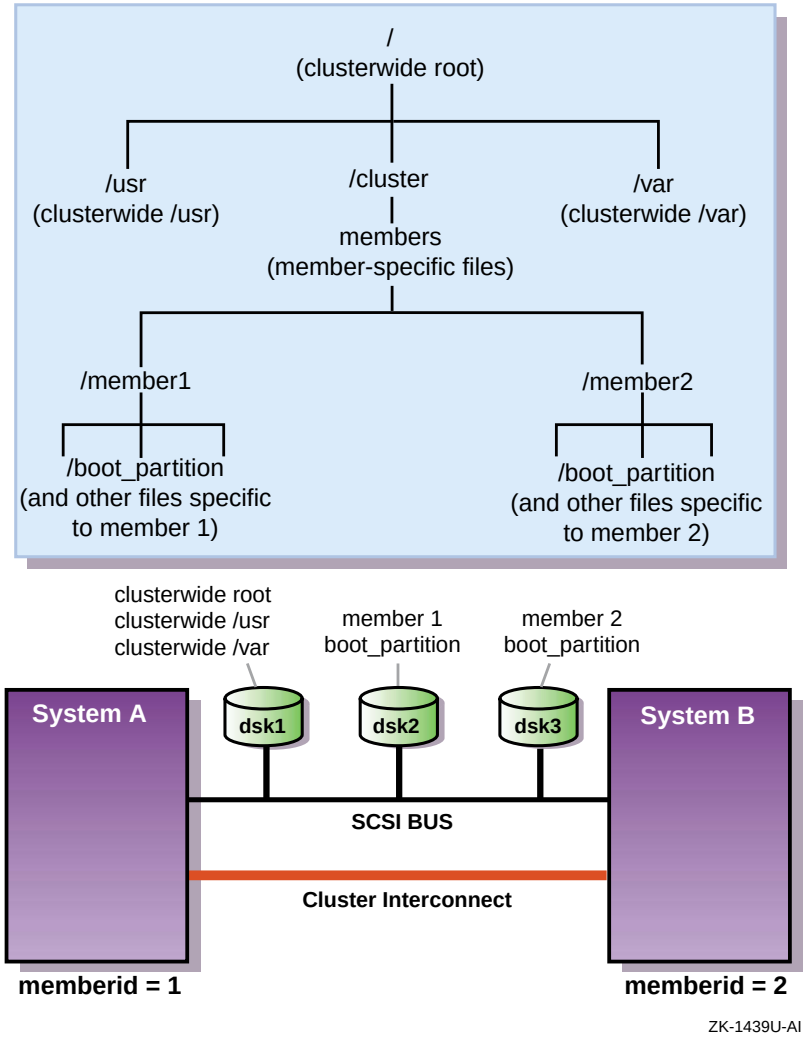
From the perspective of the CFS, each file system or AdvFS domain is served to the entire cluster by a single cluster member. Any cluster member can serve file systems on devices anywhere in the cluster. File systems mounted at cluster boot time are served by the first cluster member to have access to 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 2–2 shows the relationship between file systems contained by disks on a shared SCSI bus and the resulting cluster directory structure. Each member boots from its own boot partition, but then mounts that file system at its mount point in the clusterwide file system. Note that this figure is only an example to show how each cluster member has the same view of file systems in a cluster. There are many physical configurations possible, and a real cluster would provide additional storage to mirror the critical root (/), `/usr`, and `/var` file systems.

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




For Version 5.0A, CFS provides several performance enhancements, including elimination of double-caching at the CFS server, and support for read-ahead and larger I/O operations. Modifications to the token subsystem and CFS vnode operations improve the performance of common file-system operations.

Additionally, CFS supports the use of direct I/O (file system I/O bypassing the buffer cache) at the CFS server and the CFS client. This is primarily a performance enhancement for single-instance applications that can be collocated with the CFS server, although it will operate correctly when the CFS server is remote.

CFS has also been enhanced to choose the initial CFS server more wisely. In TruCluster Server Version 5.0, the member on which the mount command is issued is always selected as the file system's initial CFS server, regardless of whether that member has connectivity to the storage. In Version 5.0A, a member with connectivity will be chosen if the member on which the mount command is issued does not have connectivity.

Another Version 5.0A enhancement is the automatic cleanup of boot partition mount points in all cases when a member leaves the cluster. A boot partition will be forcibly unmounted, if necessary, once a member has left the cluster.

## 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 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 notion 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 check the device request dispatcher view of a device.

In the following example, device `dsk17` is on a shared bus, and is served by three cluster members.

```
# drdmgr dsk17

   View of Data from Node polishham as of 2000-01-04:16:03:46

                Device Name: dsk17
                Device Type: Direct Access IO Disk
              Device Status: OK
           Number of Servers: 3
```

```
                Server Name: provolone
               Server State: Server
                Server Name: polishham
               Server State: Server
                Server Name: pepicelli
               Server State: Server
         Access Member Name: polishham
        Open Partition Mask: 0
  Statistics for Client Node: polishham
   Number of Read Operations: 3336
  Number of Write Operations: 192
        Number of Bytes Read: 206864384
     Number of Bytes Written: 1572864
Statistics for Client Member: pepicelli
   Number of Read Operations: 1699
  Number of Write Operations: 96
        Number of Bytes Read: 103432192
     Number of Bytes Written: 786432
Statistics for Client Member: provolone
   Number of Read Operations: 5770
  Number of Write Operations: 336
        Number of Bytes Read: 360742912
     Number of Bytes Written: 2752512
```

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 Context-Dependent Symbolic Link

Although the single namespace greatly simplifies system management, there are some configuration files and directories that should not be shared by all cluster members. For example, a member's `/etc/sysconfigtab` 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 these 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 the behind-the-scenes sleight of hand needed to make sure that each member reads and writes its own copy of member-specific system configuration files.

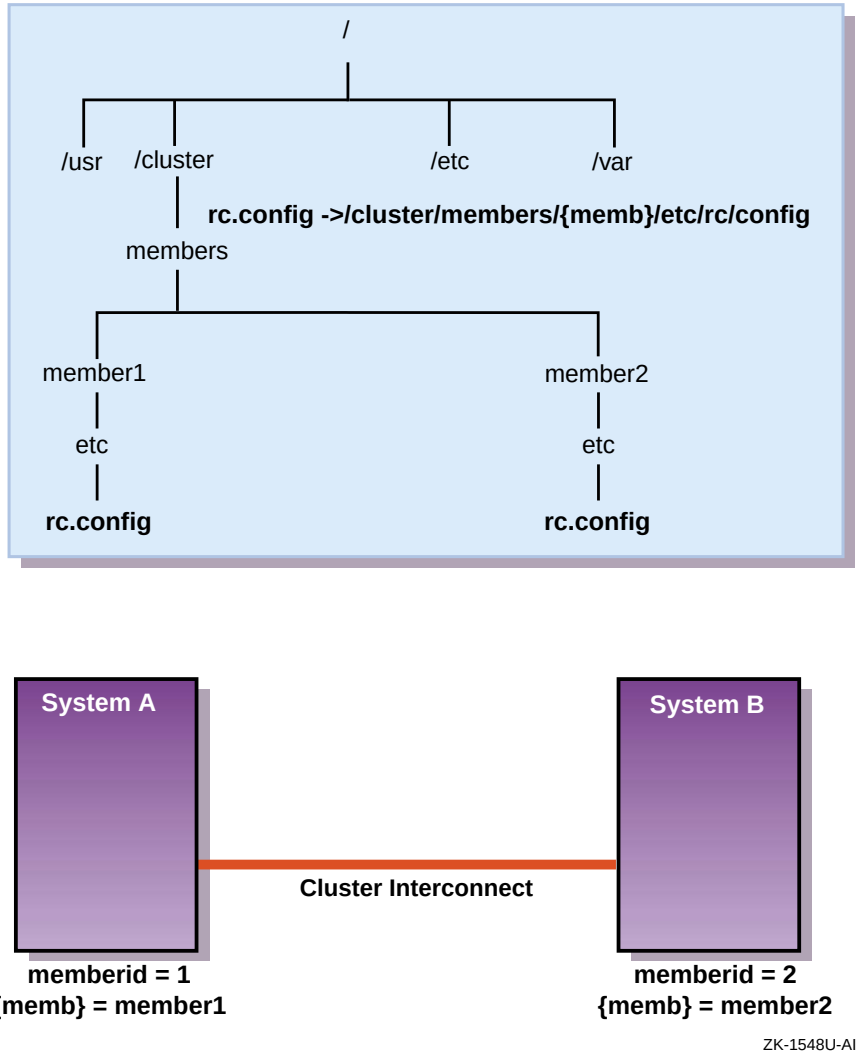
CDSLs contain a variable whose value is determined only during pathname resolution. The {memb} variable is used to access member-specific files in a cluster. The following example shows the CDSL for /etc/rc.config:

```
/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 2–3 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 TruCluster Server *Highly Available Applications* manual describes how applications can use CDSLs to maintain member-specific data sets and log files.

**Figure 2–3: 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 `/vmunix` as shown in the following example:

```
/vmunix -> cluster/members/{memb}/boot_partition/vmunix
```

If you were to move (instead of copy) a kernel to `/vmunix`, you would replace the symbolic link with the actual file; `/vmunix` would no longer be a symbolic link to `/cluster/members/{memb}/boot_partition/vmunix`.

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).

## 2.5 Device Names

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

Device names are consistent clusterwide; they are:

- Persistent beyond boot
- Stay with the device even when you move a disk or tape to a new location in the cluster

**Note**

Although Tru64 UNIX Version 5.0A supports the old-style device names as a compatibility option, TruCluster Server Version 5.0A supports only the new-style names. Applications that depend on old-style device names (or the structure of /dev) must be modified to use the new device-naming model.

Previously, device names were determined by the position of the I/O controller on the system bus, the position of the device on the I/O bus, and the device's **logical unit number (LUN)**. Starting with Tru64 UNIX Version 5.0, device names are established when the operating system first discovers the device (for example, at initial system boot time or when the device is first added), and these names are independent of the physical configuration and convey no information about the architecture or logical path.

For example, prior to Tru64 UNIX Version 5.0, disks were named as follows:

- /dev/rz2
- /dev/rz3
- /dev/rz4

This naming had encoded within it the bus and LUN of the **SCSI** disk. For example, disk 2 in bus 0 was rz2; disk 0 in bus 1 was rz8; disk 0 in bus 2 was rz16, disk 3 in bus 2 was rz19, and so on.

The new device-naming convention consists of a descriptive name for the device and an automatically assigned instance number. These two elements form the base name of the device, such as `dsk0`. Note that 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 as discovers devices.

Table 2–2 shows 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 exist in the class 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` |

  Note that with the new device special file naming for tapes, there is a direct mapping from the old name suffix to the new name suffix, as follows:

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

There are two sets of device names for tapes that 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, you can 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/disks/dsk*n* names. For more information on the hwmgr command, see hwmgr(8).

**Note**

The Logical Storage Manager (LSM) naming conventions did not change in Tru64 UNIX Version 5.0A.

## 2.6 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. Once a disk is recognized by the operating system, the disk's /dev/disk/dsk name will stay 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 controllers. If disks are moved within a TruCluster Server environment, their device names and how users access them remains the same.

**Note**

The names of disks behind RAID controllers are associated with both the WWID of their controller module and their own bus,

target, and LUN position. When they are moved, they do not retain their disk names. 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
```

## 2.7 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.

Note that there are some points to keep in mind when using LSM in a cluster. See the TruCluster Server *Cluster Administration* manual for configuration and usage issues that are specific to LSM in a TruCluster Server environment.

# 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.

Once 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 will at most allow 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 or maintain 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 may 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. This is because of the following rules enforced by the connection manager:

- A booting node cannot form a cluster unless it has quorum.
- Before the 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 would never be able to 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. Once 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 should expect when all configured votes are available. In other words, expected votes should be the sum of all node votes (see Section 3.1.2) configured in the cluster, plus the vote of the quorum disk, if one is configured (see Section 3.1.3). Each member brings its own notion of expected votes to the cluster; it is important that all members 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 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 Memory Channel 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:
   - Maximum member-specific expected votes value from the set of proposed members 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, 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 would be calculated as round_down((3+2)/2), or 2. In the case where the fourth member was added successfully, quorum votes would be calculated as 3 (round_down((4+2))/2).

   **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 (a node has joined the cluster, an existing member has been 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 Memory Channel interfaces are turned off. No commands that access a clusterwide resource work on that member. The member may appear to be hung.

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

Note that 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 may 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

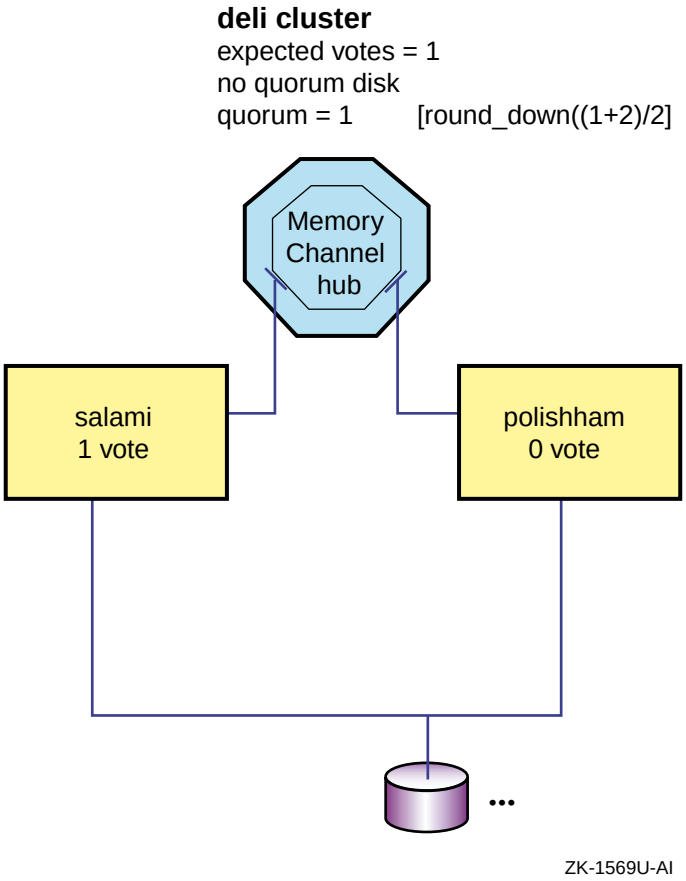
In a two-member cluster configuration, where each member has one member vote and expected votes has the value of 2, the loss of a single member will cause the cluster to lose quorum and all applications to be suspended. This type of configuration is not highly available.

A more realistic (but not substantially better) two-member configuration would assign one member 1 vote and the second member 0 (zero) votes. Expected votes would be 1. This cluster could 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**




One member contributes 1 node vote and the other contributes 0, 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
```

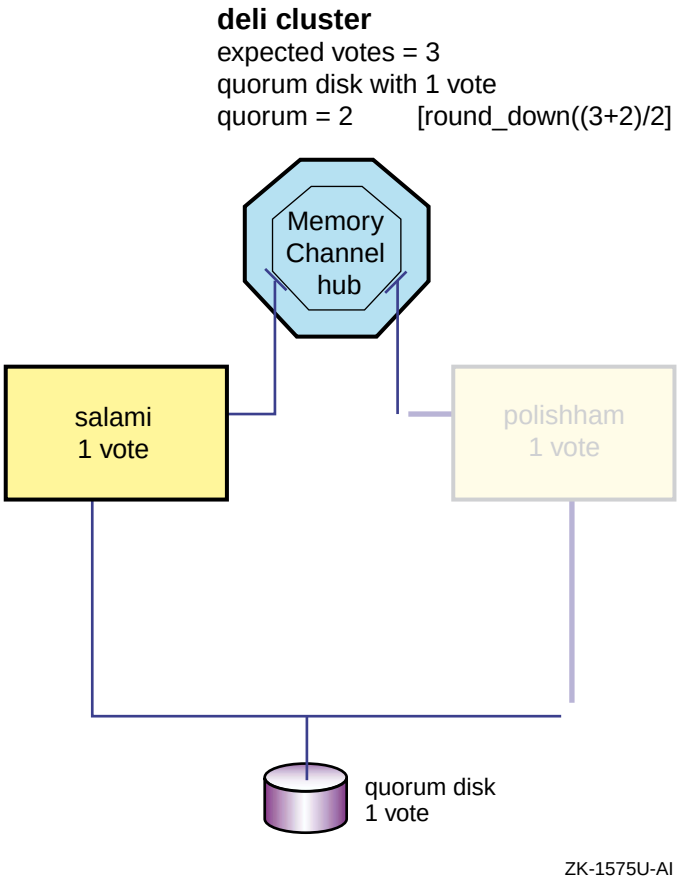
The failure or shutdown of member `salami` causes member `polishham` to lose quorum. Cluster operations are suspended.

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

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

Now, if either member or the quorum disk left the cluster, sufficient current votes will remain to keep the cluster from losing quorum. The cluster shown 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 at some other moment in the life of a cluster; 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.
- The quorum disk should be 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 will overwrite 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.

  This means that the member boot disks and the disk holding the clusterwide root (/) cannot be used 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 no votes. In general, a quorum disk should always be assigned a vote. 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. In order 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 TruCluster Server *Highly Available Applications* manual provides detailed information about moving applications from the TruCluster Software Version 1.* series of products to TruCluster Server Version 5.0A.

**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 multi-instance 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, Chapter 7, Chapter 8, and the TruCluster Server *Highly Available Applications* manual for more information on the subsystems and interfaces used to create distributed applications.

# 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

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).

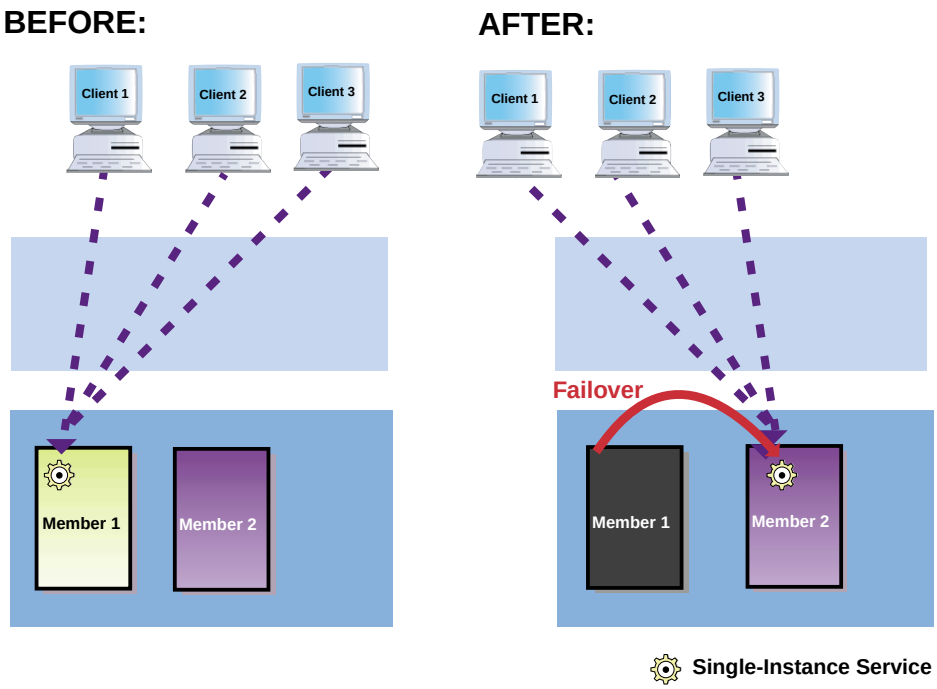
CAA can automatically relocate an application to another cluster member if a required resource, or the current member itself, becomes unavailable. This feature requires no changes to the application itself, and can be used with any single-instance application. CAA also monitors resources so that it can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

**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.

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. Once 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. Profiles that are created or modified using a text editor should be validated using `caa_profile -validate` to ensure correct syntax. 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 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 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`, 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**

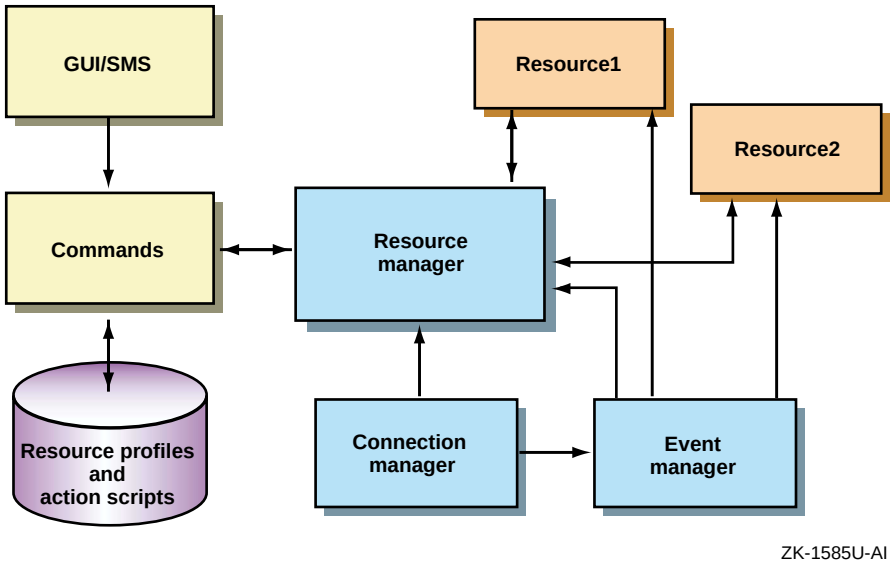
SysMan Menu and SysMan Station provide graphical user interfaces (GUIs) to perform system management tasks for the cluster, cluster members, and CAA applications. For more information on using the GUIs for performing system management tasks for CAA applications, see `sysman`(8) and the online help for 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(s). If CAA cannot restart the application on another member because the other member is down or the placement policy forbids starting the application on that member, the application is stopped. CAA will 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 will start 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 will also run 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 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, 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
```

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

The remainder of this section takes a brief look at 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 currently running the fewest application resources. Placement 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 running the fewest application resources.<br><br>You must specify a hosting members list when you select a favored placement policy. |
| `restricted` | 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 will never run on a member that is not on the list, unless you manually relocate it to that member.<br><br>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.

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 TruCluster Server *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. The file names of action scripts take the form *`resource_name`*`.scr`

The TruCluster Server *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 the application resource profile. If the action script does not finish executing within this time, CAA considers the start attempt a failure and will either attempt to start the application on another member or fail completely.

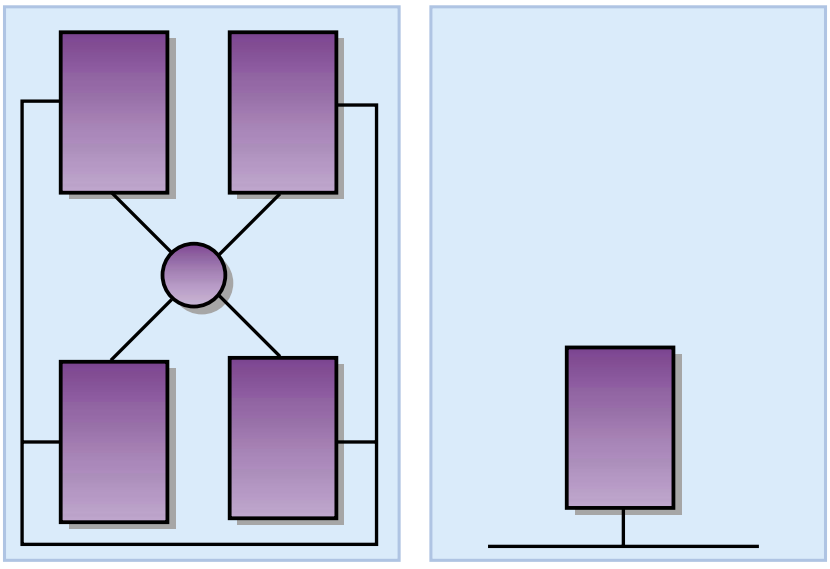
Both the `caa_profile` command and the SysMan suite of applications can be used to create simple action scripts when creating resource profiles. You may need to edit these action scripts to customize the start, stop, and check procedures for an application.

# 6

# Cluster Alias

A **cluster alias** is an IP address that makes some or all of the systems in a cluster look like a single system to TCP and 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's View of a Cluster With and Without Cluster Alias**




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.

**Note**

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.

---

Each system in a cluster can receive packets addressed to a cluster alias. The receiving system silently redirects packets to a cluster member running the requested application or service.

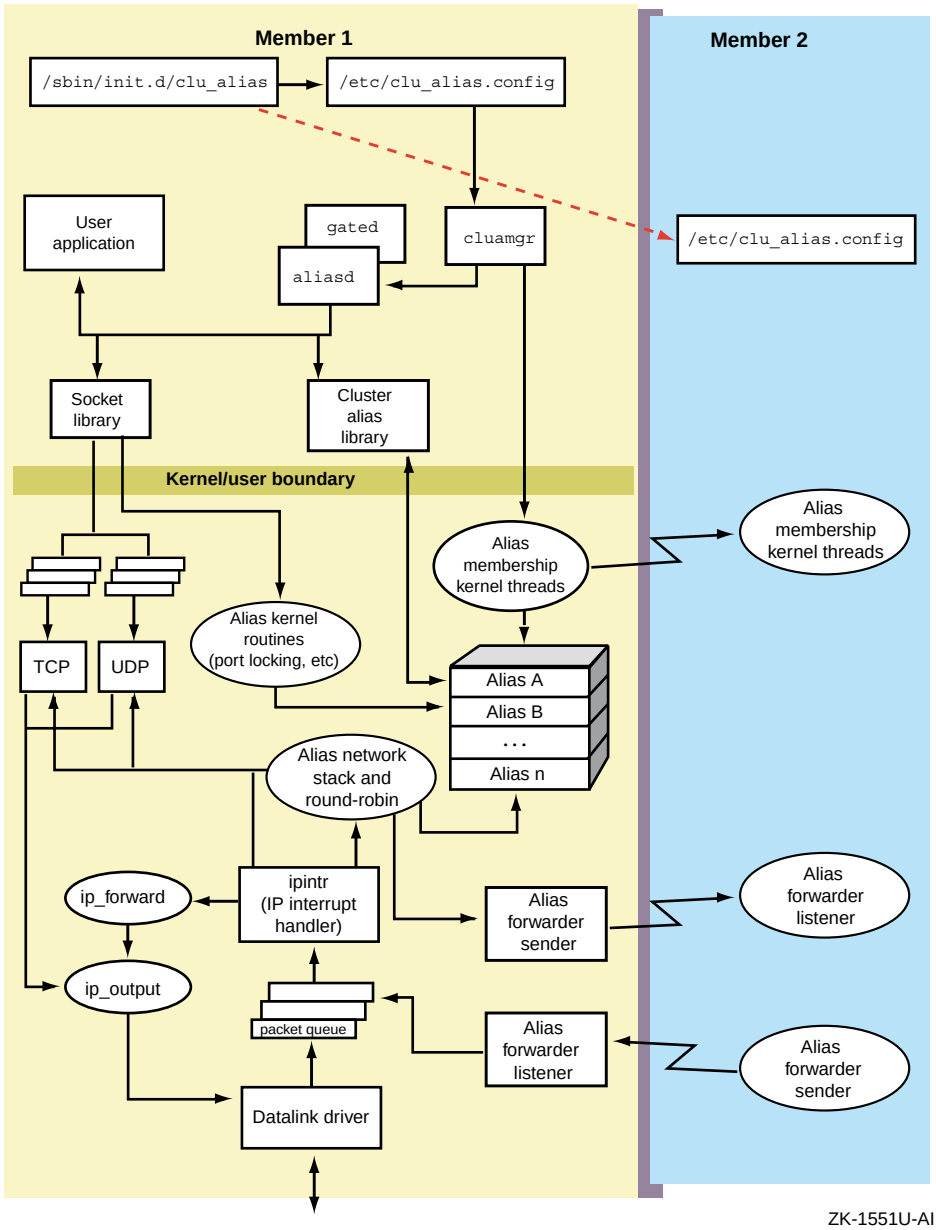
A cluster can have more than one cluster alias. One alias, the default cluster alias, is created during cluster installation and all members can receive packets addressed to this alias. A cluster administrator can create additional aliases as needed. One suggestion is to use the default alias for a while, and then decide whether your site can benefit from additional aliases. In many cases, the default alias is sufficient; the TruCluster Server *Cluster Administration* manual describes a situation where a site uses two aliases for load balancing.

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




ZK-1551U-AI

This remainder of this chapter discusses the following topics:

- A general overview of cluster aliases (Section 6.1)
- The default cluster alias (Section 6.2)

- The number of aliases a site can have (Section 6.4)
- The location of alias IP addresses (Section 6.5)
- Routing for alias IP addresses (Section 6.6)
- Cluster alias vMAC support (Section 6.7)
- How the cluster alias subsystem routes for `in_single` and `in_multi` services (Section 6.8)
- The attributes assigned to aliases (Section 6.9)
- The attributes assigned to services (Section 6.10)
- How to run an RPC application on multiple cluster members (Section 6.11)
- How packets are redirected within the cluster (Section 6.12)

## 6.1 Overview

Cluster aliases free clients from having to connect to specific cluster members for services. Just as 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 as you would to single host.

Each system in a cluster explicitly joins the aliases to which it wants to belong. Once 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 Transmission Control Protocol (TCP) connection requests or User Datagram Protocol (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 The Default Cluster Alias

There is one special alias, called the **default cluster 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

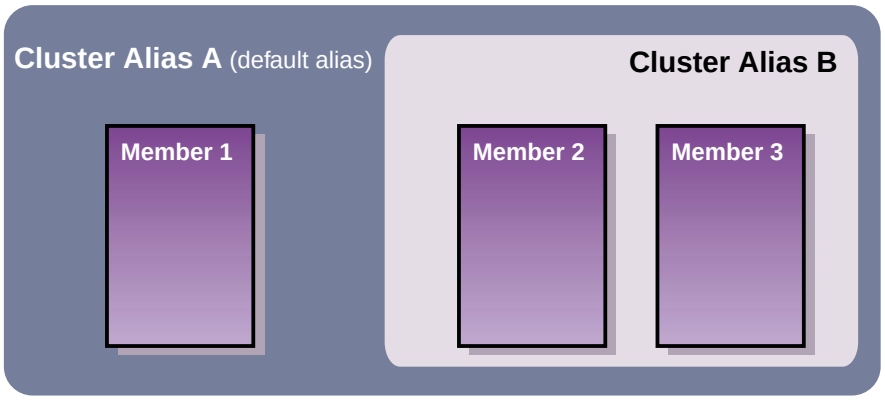
this cluster name with a user-specified default cluster alias IP address. For example, for a cluster named `deli` whose alias IP address is `16.140.112.209`, the installation procedure adds the following entry to `/etc/hosts`:

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

Each cluster member is a member of the default cluster alias. The command that makes a cluster member a member of the default cluster alias is in each member's `/etc/clu_alias.config` file. All cluster members automatically join the default cluster alias at boot time.

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**




ZK-1443U-AI

Several standard Internet services use the IP address of the default cluster alias as the source address for outgoing packets. Therefore, all 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 Memory Channel subnet used by the cluster for internal communication.

## 6.3 NFS and the Default Cluster Alias

In order to preserve single-system semantics and avoid NFS locking problems, when a cluster is configured as NFS server, the default cluster alias is the IP address through which clients must request NFS services. NFS mount requests directed at individual cluster members are rejected.

UDP NFS packets are redirected to the cluster member that is serving the file system. All UDP NFS traffic for that file system will be 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.

TCP connection requests are assigned to a member based on the alias round-robin algorithm and alias selection weights. (Because there is no file system information in the connection request, 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.

## 6.4 Number of Aliases

After cluster installation, a site can define as many aliases as it needs. The cluster administrator determines which cluster members join which aliases.

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 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.

## 6.5 Location of Alias IP Addresses

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

**common subnet** A subnet connected to a physical network interface.

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 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 registered as a normal subnet with local routers.

If the `cluamgr virtual` option is assigned to an alias address, a cluster member will advertise a host route and a network route to the alias.

Note that multiple clusters on the same local area network (LAN) can use the same virtual subnet.

**Note**

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

The choice of subnet type depends mainly on whether the existing subnet (that is, the common subnet) has enough addresses available for cluster aliases. If addresses are not easily 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 Memory Channel subnet.

## 6.6 Routing for Alias Addresses

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 at boot time.

A cluster member does not have to be a member of an alias in order to route for that alias. However the cluster member must have an entry for that alias

in its /etc/clu_alias.config file. In the following example, this cluster member routes for alias1 and alias2, but will receive requests/packets only for alias1:

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

Which cluster members route for which aliases in common subnets is determined by the router priority assigned to each alias on each cluster member. Section 6.9 describes router priority.

The cluster alias daemon, aliasd, transparently handles the routing configuration for cluster aliases, automatically adding any needed host routes 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.

**Note**

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

### 6.6.1 Common Subnet Routing

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

The node elected to advertise the alias configures the alias's IP address to be advertised using proxy ARP over a network interface in the same subnet. Other cluster members that join the alias, and that are configured to be routing members, become capable of taking over the proxy ARP function if necessary. When a designated alias router or network interface fails, other potential routers are notified. They then elect a new router for the alias.

If a cluster is connected to two external common subnets, and if an alias address resides on one of those common subnets, interfaces directly connected to that subnet will use proxy ARP to advertise the alias. Interfaces connected to all subnets will also use host routes for the alias.

### 6.6.2 Virtual Subnet Routing

The alias daemon, aliasd, creates a /etc/gated.conf.member*n* file for each cluster member. The alias configuration process modifies 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

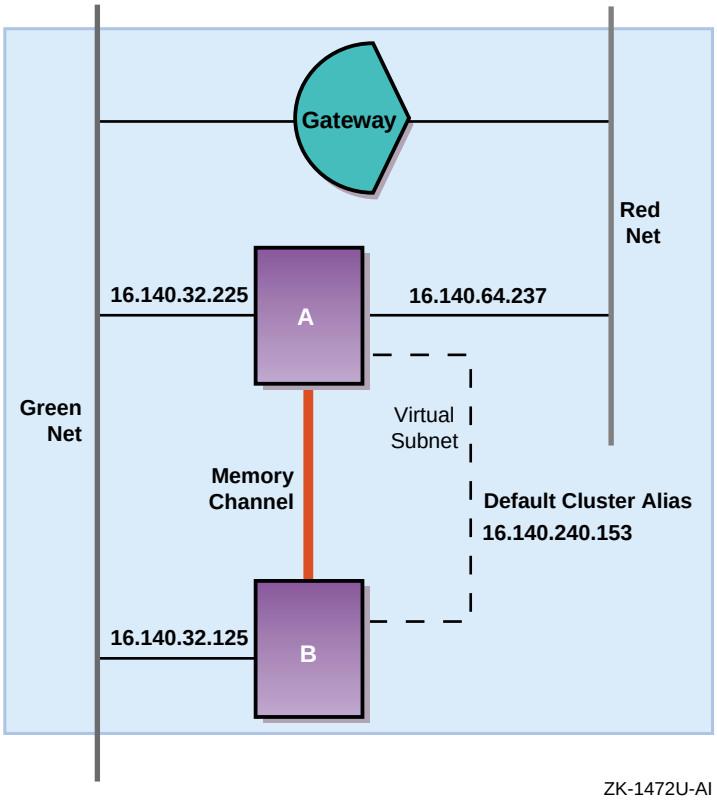
advertise a route to each cluster alias host address through each network interface on that member.

If the `cluamgr virtual` option is assigned to an alias address, the cluster member will advertise a network route to the virtual subnet.

### 6.6.3 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 a virtual subnet. Because clients on the Red and Green networks are not on the same subnet as the alias, a host route to the address is advertised on both networks.

**Figure 6–4: Alias Routing Example**




Note that although all alias addresses on virtual subnets are advertised through host routes, using a host route does not necessarily mean that an address resides on a virtual subnet.

## 6.7 Cluster Alias vMAC Support

When a cluster alias IP address is configured in a common subnet, one cluster member in that subnet will, based on its router priority (`rpri`) value for that alias, act as the alias's proxy ARP master. This member will respond to local ARP requests addressed to the alias, and will broadcast a gratuitous ARP packet to let other systems know the 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 cluster-alias-to-MAC association.

However, this broadcast packet is a problem for systems that do not understand gratuitous ARP packets. These systems will not become aware of changes in the cluster alias-to-MAC association 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 TruCluster Server *Cluster Administration* manual describes how to enable vMAC support for a cluster alias.

## 6.8 in_single and in_multi Services

Service ports 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 will receive connection requests or packets for that service. If that member becomes unavailable, the cluster alias subsystem selects another member of that alias to receive all connection requests or packets.
- When a service's port is designated as `in_multi`, the alias subsystem routes connection requests and packets to all eligible members of the alias.

By default, the cluster alias subsystem treats all services as `in_single`. In order 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 6.10 for more information on service 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.

**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 TruCluster Server *Cluster Administration* manual provides more information on the differences between cluster alias and CAA.

The following two figures show how the 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**

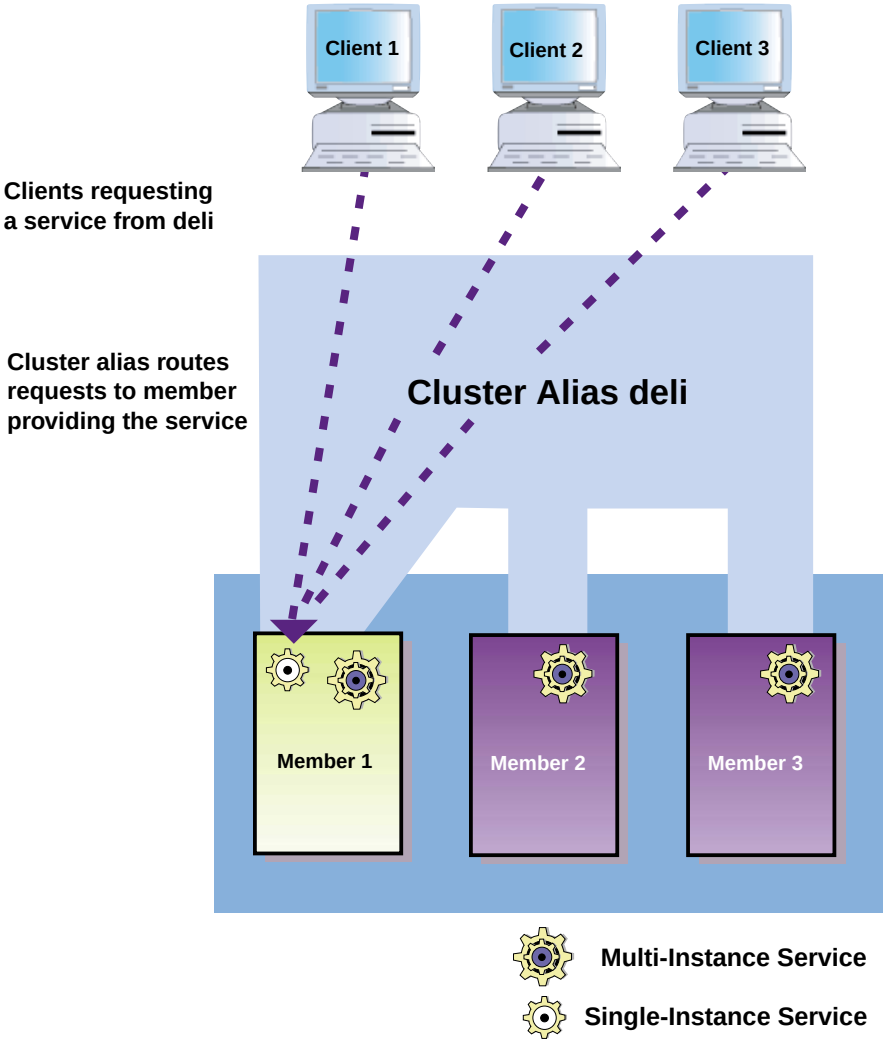
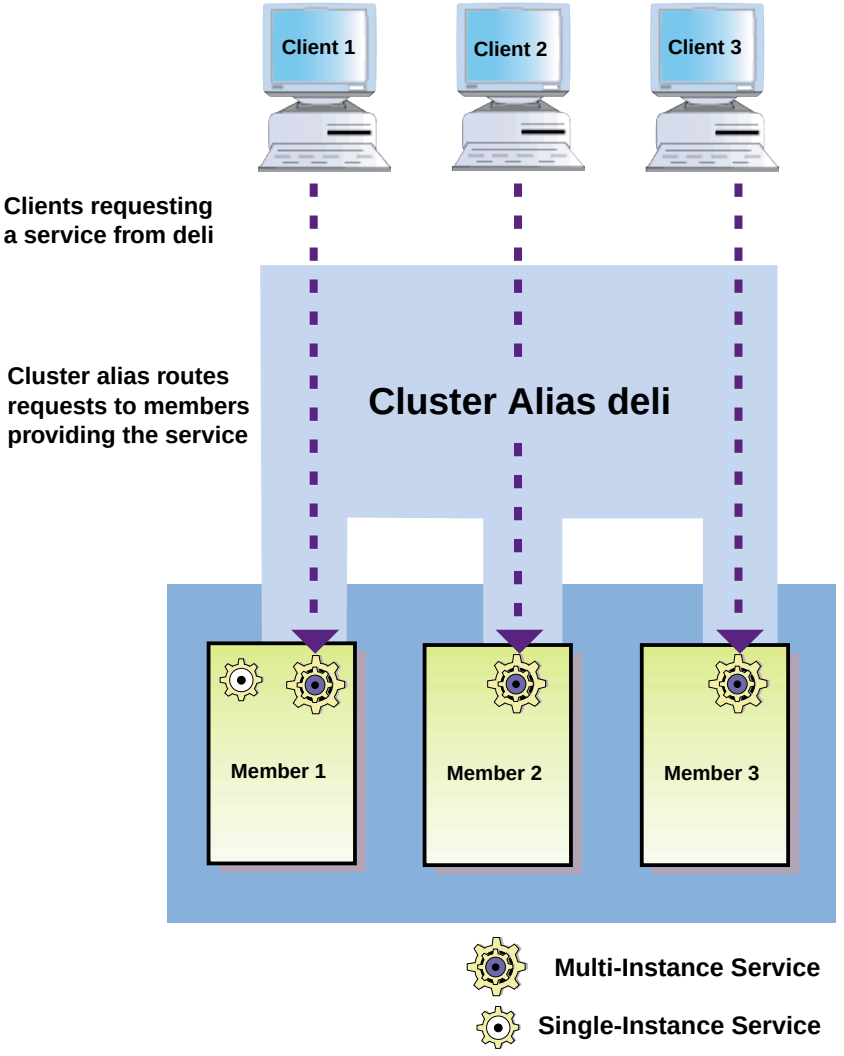


**Figure 6–6: in_multi Service Accessed Through Default Cluster Alias**




## 6.9 Alias Attributes

Alias attributes are member-specific. Each cluster member has its own view of an alias. For example, one cluster member could route for an alias but not be a member of that alias, but another cluster member could both route for that alias and be an end recipient for requests or messages addressed to that alias.

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

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. For each alias in a common subnet, the cluster member with the highest router priority for that alias will route for that alias. (This option is not valid for an alias whose address is in a virtual subnet.)

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

**selection priority** The selection priority (`selp`) determines the order in which members of an 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. 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 weight** The selection weight (`selw`) indicates the number of connections (on average) 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 once it is eligible.)

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 is a way to balance application overhead within a cluster, and router priority is a way to balance alias-routing 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.

## 6.10 Service Attributes

The `/etc/clua_services` 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`.

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.

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 should the active service go 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 address(es) 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 flagged as inactive. If the active service fails, one of the inactive service daemons is marked as active.

Any port not explicitly listed in `clua_services` 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 that will not honor connection request to alias addresses.

**in_nolocal** Indicates that the port will not honor connection requests to nonalias addresses. For TCP, the port will not accept connections; for UDP, the port will drop messages unless addressed to a cluster alias.

**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).

It is important to remember that 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 will use 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. Any well-known port between 512 and 1024 (> 512 and < 1024) that is either assigned to a specific network service in `/etc/services` or is actively listened to by an application should be declared as `static` in `/etc/clua_services`.

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.

## 6.11 RPC Services and Cluster Alias

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 on multiple cluster members, all accessible via a cluster alias. In addition to ensuring that each instance of an RPC application uses the same common port, the functions also inform the portmapper 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.

## 6.12 Redirecting Packets Within a Cluster

A packet addressed to a cluster alias can arrive at any cluster member. This member must determine which cluster member should receive and process the packet. The alias subsystem monitors calls to bind() and listen(), and bases its decision on which members of an alias are available and the type of packet received. The following table shows 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. Look for active listens on the 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 alias. Use the weighted round-robin algorithm to select the member that should get this packet. Forward the packet to the member. |
| 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. If needed, forward the packet to the member. |

# 7

# Memory Channel

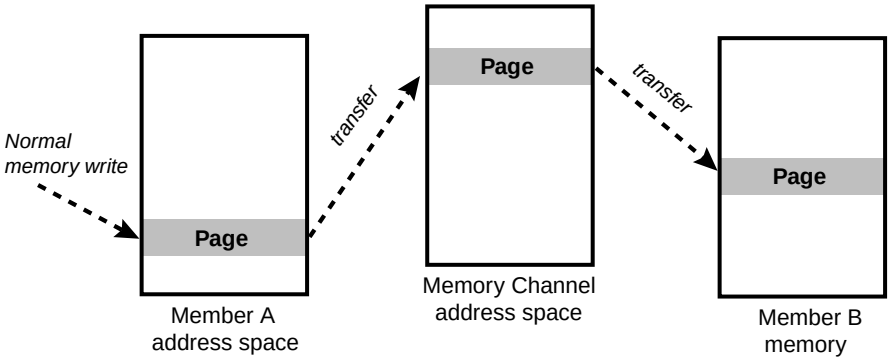
All cluster members must have a direct connection to all other members in order to facilitate communications among members, and to provide a fast and reliable transport for passing messages throughout the cluster. This version of the TruCluster Server product supports the Memory Channel interconnect, a specialized interconnect designed specifically for the needs of clusters. The Memory Channel interconnect provides both broadcast and point-to-point connections between cluster members. Future releases will support other types of cluster interconnect.

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 one of the following situations occurs in the primary interconnect:

- More than ten errors are logged within one 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 ten Memory Channel errors occur on any member system within a one-minute interval, the Memory Channel error recovery code attempts to determine if 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 TruCluster Server *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 TruCluster Server *Highly Available Applications* manual for a discussion of the Memory Channel API library.

# 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. A resource tree allows you 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 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 and administration.

## 9.1 Installation

Previous versions of TruCluster supported three installation types: full installation, rolling upgrade, and simultaneous upgrade. The rolling upgrade and simultaneous upgrade procedures provided a way to preserve the existing cluster or ASE configuration information while installing a later version of the product.

TruCluster Server Version 5.0 and Version 5.0A support two installation types: a full installation and an upgrade procedure. Rolling upgrade is not supported for these initial releases because there are significant changes to both the base operating system and the cluster architectures. For this reason, the recommended installation path is a full installation of the base operating system followed by a full installation of the TruCluster Server.

The TruCluster Server *Software Installation* manual describes how to upgrade a Version 5.0 cluster to a Version 5.0A cluster. That manual also provides three options for customers upgrading to Version 5.0A from TruCluster Software Version 1.5 or Version 1.6 products. Two of these upgrade options use scripts specifically designed to facilitate the migration of storage from the old cluster (`rz*` style device names) to the new cluster (`dsk*` style device names).

TruCluster Server Version 5.0A incorporates the software infrastructure required to support future rolling upgrades. Customers who install TruCluster Server Version 5.0A will be able to perform a rolling upgrade to the next TruCluster Server release. As part of preparing for rolling upgrades, TruCluster Server Version 5.0A provides a new command, `clu_upgrade`, which will control the rolling of a cluster to the next release. Note that this command will be of use only when you are installing the release that follows Version 5.0A. See `clu_upgrade`(8) for a description of the `clu_upgrade` command.

One major difference in installing TruCluster Server Version 5.* is that you install Tru64 UNIX on only one system in the cluster. Because CFS creates shared clusterwide file systems, once a cluster is created, additional members boot into the cluster and have access to these files. (In previous

releases, 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 form a new TruCluster Server cluster:

1. Using the information in the TruCluster Server *Hardware Configuration* manual, configure the system and storage hardware and firmware.
2. Selecting 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 you plan to use in the cluster.
4. Load the TruCluster Server license and software.

**Note**

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

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.

See the TruCluster Server *Software 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. Each item is followed by one or more of the cluster-specific commands used to manage or monitor it. With the exception of the installation scripts, you can use the SysMan Menu and SysMan Station GUIs to perform the related command-line functions.

- Cluster creation and configuration, which supports creating 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 allows you to 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 allows access to cluster resources (`clu_quorum`).
- Optional load-balancing of the device request dispatcher subsystem (`drdmgr`).
- Optional load-balancing of CFS servers (`cfsmgr`).

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 logged in on the cluster member where the command executes. To send a message to all users logged in on all cluster members, use the `wall -c` form of the command. The same logic applies to the `shutdown` command; you can shut down an individual member or the entire cluster.

See the TruCluster Server *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 software environment.

**action script**
Shell scripts used by CAA to control how applications are started, stopped, and checked. 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 the side or rear of 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*

**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.

**CDSL**
See *context-dependent symbolic link (CDSL)*

**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 high-performance interconnect.

**cluster alias**

An IP address 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)**

The CAA subsystem 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.

**cluster expected votes**

See *expected votes*

**Cluster File System (CFS)**

A cluster 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.

**cluster interconnect**

Private physical bus employed by cluster members for intracluster communications.

**cluster member**

The basic computing resource in a cluster. A member system must be physically connected to a cluster interconnect and at least one shared SCSI bus.

In common usage, a system configured with TruCluster Server software that is capable of 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**

A situation in which an existing cluster can divide into two or more clusters.

**cluster router**
In the context of cluster aliases, a cluster member that makes a cluster alias IP address known to the network and receives incoming packets addressed to the alias. By default, all cluster members are cluster routers.

**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)**
A special form of a symbolic link whose target pathname includes an environment variable, {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.

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

**dedicated port**
See *locked port*

**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.

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

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

**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**

The cluster software component that synchronizes access to shared resources among cooperating processes throughout the cluster.

**DLM**

See *distributed lock manager*

**expected votes**

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

**event manager**

The event manager (EVM) facility 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.

**EVM**

See *event manager*

**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.

**fast bus speed**

A bus speed that uses the fast synchronous transfer option, enabling I/O devices to attain high peak-rate transfers (10 MB per second) in synchronous mode.

**firmware**

Software code stored in hardware.

**highly available**
In the TruCluster software, the ability to survive any single hardware or software failure.

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

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

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

**in_multi service**
When a service's port is designated as `in_multi`, the cluster alias subsystem routes connection requests and packets to all eligible members of the alias.

**in_noalias service**
When a service's port is designated as `in_noaliase`, the cluster alias subsystem ensures that the port will not receive inbound alias messages.

**in_nolocal service**
When a service's port is designated as `in_nolocal`, the cluster alias subsystem ensures that the port will not honor connection requests to a nonalias address.

**in_single service**
When a service's port is designated as `in_single`, the cluster alias subsystem ensures that only one alias member will receive connection requests or packets for that service.

**local bus**
See *private SCSI 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

The Logical Storage Manager (LSM) is 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.

### 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.

### LSM

See *Logical Storage Manager*

### 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 volume

A Logical Storage Manager (LSM) volume is 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.

### LSM plex

A Logical Storage Manager (LSM) plex is a copy of an LSM volume's logical data address space, 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.

### LUN

See *logical unit number*

**member**

See *cluster member*

**member ID**

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

**member votes**

The number of quorum votes assigned to a cluster member.

**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.

**mount point**

A directory file that is the name of a mounted file system.

**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.

**network**

Two or more computing systems that are linked for the purpose of exchanging information and sharing resources.

**network interface**

The network adapter and the software that allows a system to communicate over a network.

**nonvoting member**

A cluster member with 0 (zero) votes is considered to be a nonvoting member.

See also *voting member*

**out_alias service**

When a service's port is designated as `out_alias`, the cluster alias subsystem ensures that the default cluster alias is used as the source address whenever the port is used as a destination.

**partition**

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

**peripheral component interconnect**

A peripheral component interconnect (PCI) bus is an industry-standard expansion I/O bus that is a synchronous, asymmetrical I/O channel.

**PCI**

See *peripheral component interconnect*

**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 placement policy determines where an application under CAA control is run. Supported policies are: `balanced`, `favored`, and `restricted`.

**private SCSI bus**

A SCSI bus that connects private storage to the local system.

**private storage**

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

**proxy ARP**

The Address Resolution Protocol (ARP) is used to map 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.

In a cluster, proxy ARP is the mechanism used by the physical cluster members to handle requests addressed to cluster aliases whose addresses reside on common subnets.

**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 member and quorum disk votes in the cluster equal or exceed 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 required to form or maintain a cluster. The formula for calculating quorum votes is:

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

**RAID**

See *redundant array of inexpensive disks (RAID)*

**RAID array controller**

A SCSI bus adapter between a differential SCSI bus and the single-ended RAID array storage shelves. It responds to host commands to access the RAID array disk or tape devices.

**redundant array of inexpensive disks (RAID)**

A technique that organizes disk data to improve performance and reliability. RAID has three attributes:

- It is a set of physical disks viewed by the user 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**

Describes duplicate hardware that provides spare capacity that can be used when a component fails.

**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**

The resource manager consists of 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 that monitor the status of a particular type of resource.

**resource monitor**

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

**resource profile**

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

**router priority**

Router priority 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.

**script**

A program that is interpreted and executed by the shell.

**SCSI**

See *Small Computer System Interface*

**SCSI-2**

An extension to the original SCSI standard featuring multiple systems on the same bus and hot swap. Hot swap is the ability to replace a device on a shared bus while the bus is active. 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 either 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 controller, peripheral controller, or intelligent peripheral that can be attached to a SCSI bus.

**SCSI ID**

Unique address from 0-15 that identifies a device on a SCSI bus.

**selection priority**

Selection priority determines the order in which members of a cluster 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**

Selection weight indicates the number of connections (on average) this member is given before connections are given to the next alias member with the same selection priority value.

**server**

A computing system that provides a specific set of applications or data to clients.

**shared SCSI bus**

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

**shared storage**
Disks that are connected to a shared SCSI bus.

**signal converter**
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 the initial startup and failover characteristics for a single-instance application.

**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.

**Small Computer System Interface**
An American National Standards Institute (ANSI) 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.

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

**standard 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**
When a service's port is designated as `static`, the cluster alias subsystem ensures that the port will not be assigned as a dynamic port.

**StorageWorks**
The modular storage subsystem (MSS), which consists of a family of mass storage products that can be configured to meet current and future storage needs.

**subset**
A software module that can be installed, which is compatible with the Tru64 UNIX `setld` software installation utility.

**system bus**
The private (nonshared) interconnect used on the CPU subsystem. This bus connects the processor module, the memory module, and the I/O module.

**target**
A device that can be addressed by a SCSI ID on a SCSI bus.

**terminator**
Resistor array device used for terminating 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 RAID controller.

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

**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.

**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.

**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.

**vMAC**
In the context of cluster aliases, a vMAC (virtual Media Access Control) 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.

**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.

**votes**
Votes (either 0 or 1) are contributed to the cluster by cluster members and by the **quorum disk** if one is configured. The connection manager uses votes to calculate quorum.

**voting member**
Each member with a vote is considered to be a voting member of the cluster.

See also *nonvoting member*

**worldwide ID**
A worldwide ID (WWID) is a unique identifier assigned to a disk by its manufacturer.

**WWID**
See *worldwide ID*

**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

## D



## E



## F



## G



## H



## I



## L

## M



## N



## O



## P



## Q



## R



## S

## U



## V



## W

# How to Order Tru64 UNIX Documentation

You can order documentation for the Tru64 UNIX operating system and related products at the following Web site:

**http://www.businesslink.digital.com/**

If you need help deciding which documentation best meets your needs, see the Tru64 UNIX *Documentation Overview* or call **800-344-4825** in the United States and Canada. In Puerto Rico, call **787-781-0505**. In other countries, contact your local Compaq subsidiary.

If you have access to Compaq's intranet, you can place an order at the following Web site:

**http://asmorder.nqo.dec.com/**

The following table provides the order numbers for the Tru64 UNIX operating system documentation kits. For additional information about ordering this and related documentation, see the *Documentation Overview* or contact Compaq.

| Name | Order Number |
|---|---|
| Tru64 UNIX Documentation CD-ROM | QA-6ADAA-G8 |
| Tru64 UNIX Documentation Kit | QA-6ADAA-GZ |
| End User Documentation Kit | QA-6ADAB-GZ |
| Startup Documentation Kit | QA-6ADAC-GZ |
| General User Documentation Kit | QA-6ADAD-GZ |
| System and Network Management Documentation Kit | QA-6ADAE-GZ |
| Developer's Documentation Kit | QA-6ADAG-GZ |
| Reference Pages Documentation Kit | QA-6ADAF-GZ |

# Reader's Comments

**TruCluster Server**
Technical Overview
AA-RHGVB-TE

Compaq welcomes your comments and suggestions on this manual. Your input will help us to write documentation that meets your needs. Please send your suggestions using one of the following methods:

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

If you are not using this form, please be sure you include the name of the document, the page number, and the product name and version.

**Please rate this manual:**

| | Excellent | Good | Fair | Poor |
|---|---|---|---|---|
| Accuracy (software works as manual says) | ☐ | ☐ | ☐ | ☐ |
| Clarity (easy to understand) | ☐ | ☐ | ☐ | ☐ |
| Organization (structure of subject matter) | ☐ | ☐ | ☐ | ☐ |
| Figures (useful) | ☐ | ☐ | ☐ | ☐ |
| Examples (useful) | ☐ | ☐ | ☐ | ☐ |
| Index (ability to find topic) | ☐ | ☐ | ☐ | ☐ |
| Usability (ability to access information quickly) | ☐ | ☐ | ☐ | ☐ |

**Please list errors you have found in this manual:**

| Page | Description |
|---|---|
| _____ | _____ |
| _____ | _____ |
| _____ | _____ |
| _____ | _____ |

**Additional comments or suggestions to improve this manual:**

_____
_____
_____
_____

**What version of the software described by this manual are you using?** _____

Name, title, department _____
Mailing address _____
Electronic mail _____
Telephone _____
Date _____

Do Not Cut or Tear - Fold Here and Tape

COMPAQ

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

# BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPAQ COMPUTER CORPORATION
UBPG PUBLICATIONS MANAGER
ZKO3-3/Y32
110 SPIT BROOK RD
NASHUA NH 03062-2698

Do Not Cut or Tear - Fold Here

Cut on This Line