

# TruCluster Server

---

## クラスタ管理ガイド

Part Number: AA-RM85D-TE

**2002 年 11 月**

ソフトウェア・バージョン: TruCluster Server バージョン 5.1B

オペレーティング・システム: Tru64 UNIX バージョン 5.1B

本書は、UNIX に精通したシステム管理者を対象とし、TruCluster Server ソフトウェアを実行する Tru64 UNIX システムの管理方法について説明します。本書では、SysMan などの管理ツールについて説明するとともに、クォーラムとポート、ネットワーク・サービス、高可用性アプリケーション、ファイル・システムなど、クラスタ管理のために日常行う管理作業について説明します。

---

© 2002 Hewlett-Packard Company

本書の著作権は日本ヒューレット・パッカード株式会社が保有しており、本書中の解説および図、表は日本ヒューレット・パッカードの文書による許可なしに、その全体または一部を、いかなる場合にも再版あるいは複製することを禁じます。

日本ヒューレット・パッカードは、弊社または弊社の指定する会社から納入された機器以外の機器で対象ソフトウェアを使用した場合、その性能あるいは信頼性について一切責任を負いかねます。

本書に記載されている事項は、予告なく変更されることがありますので、あらかじめご承知おきください。万一、本書の記述に誤りがあった場合でも、弊社は一切その責任を負いかねます。

本書で解説するソフトウェア(対象ソフトウェア)は、所定のライセンス契約が締結された場合に限り、その使用あるいは複製が許可されます。

COMPAQ, Compaq ロゴ, Digital ロゴは U.S. Patent and Trademark Office に登録されています。Alpha, AlphaServer, NonStop, TruCluster, および Tru64 は米国 Compaq Computer Corporation の商標です。

Microsoft, Windows および Windows NT は米国 Microsoft 社の登録商標です。Intel は米国 Intel 社の登録商標です。Motif, OSF/1, UNIX, The Open Group および X/Open は、The Open Group の米国ならびに他の国における商標です。

このドキュメントに記載されているその他の会社名および製品名は、各社の商標または登録商標です。

---

# 目次

## まえがき

## 1 クラスタ管理の概要

1.1	クラスタ用のコマンドとユーティリティ .....	1-2
1.2	クラスタで異なる動作をするコマンドと機能 .....	1-4

## 2 クラスタ管理ツール

2.1	利用可能な管理ツールとインタフェースの概要 .....	2-1
2.1.1	クラスタ・ツールのクイック・スタート .....	2-2
2.1.2	HP Insight Manager の統合 .....	2-3
2.1.3	HP Insight Manager XE の統合 .....	2-3
2.1.4	利用可能な管理ツールとインタフェース .....	2-4
2.2	クラスタ構成ツールと関連ユーザ・インタフェース .....	2-5
2.3	SysManの構成要素の概要 .....	2-6
2.3.1	SysMan Menu の紹介 .....	2-7
2.3.2	SysMan Station の紹介 .....	2-8
2.3.3	SysMan コマンド行の紹介 .....	2-9
2.4	クラスタでの SysMan Menu の使用 .....	2-10
2.4.1	フォーカスの取得 .....	2-10
2.4.2	コマンド行でのフォーカスの指定 .....	2-10
2.4.3	SysMan Menu の起動 .....	2-11
2.5	クラスタでの SysMan Station の使用 .....	2-13
2.5.1	SysMan Station の起動 .....	2-17
2.6	クラスタでの SysMan Java アプレットの使用 .....	2-18
2.6.1	SysMan Java アプレットの起動 .....	2-20

2.7	クラスタでの SysMan Java PC アプリケーションの使用 .....	2-20
2.7.1	PC 上での SysMan Java アプリケーションの起動 .....	2-21
2.8	クラスタでの SysMan コマンド行インタフェースの使用 .....	2-21
2.9	クラスタでの HP Insight Manager の使用 .....	2-22
2.9.1	HP Insight Manager の起動 .....	2-24
2.10	Tru64 UNIX 構成レポートの使用 .....	2-24
<b>3</b>	<b>クラスタ別名サブシステムの管理</b>	
3.1	クラスタ別名の特長についての要約 .....	3-2
3.2	構成ファイル .....	3-5
3.3	一般的な設計に関する考慮事項 .....	3-6
3.4	クラスタ別名の作成準備 .....	3-9
3.5	クラスタ別名の指定と参加 .....	3-10
3.6	クラスタ別名属性とサービス・ポート属性の変更 .....	3-12
3.7	クラスタ別名のモニタ .....	3-13
3.8	クラスタ別名からのメンバの削除 .....	3-14
3.9	クラスタ別名の削除 .....	3-14
3.10	アプリケーションの負荷分散 .....	3-16
3.11	クラスタ単位のポート・スペースの拡張 .....	3-19
3.12	クラスタ別名の vMAC サポートの有効化 .....	3-19
3.13	ルーティング構成のガイドライン .....	3-22
3.14	ルーティング・デーモンと静的ルーティングのオプション ...	3-23
3.15	クラスタ別名と NFS .....	3-26
3.16	クラスタ別名と CAA (Cluster Application Availability) .....	3-27
<b>4</b>	<b>クラスタのメンバシップ管理</b>	
4.1	接続マネージャの概要 .....	4-1
4.2	クォーラムとポートの概要 .....	4-2
4.2.1	システムをクラスタのメンバにする方法 .....	4-2

4.2.2	期待ポート .....	4-3
4.2.3	現在のポート .....	4-4
4.2.4	ノード・ポート .....	4-4
4.2.5	クォーラム・ディスク・ポート .....	4-5
4.3	クラスタのクォーラム・ポートの計算の概要 .....	4-6
4.4	接続マネージャの動作例 .....	4-9
4.5	クォーラム・ディスクの使用 .....	4-13
4.5.1	障害が発生したクォーラム・ディスクの交換 .....	4-18
4.6	clu_quorum コマンドによるクラスタのクォーラム情報の表示 .....	4-19
4.7	クラスタ・ポートの割り当て例 .....	4-20
4.8	接続マネージャのモニタ .....	4-22
4.9	接続マネージャのパニック .....	4-23
4.10	不適切な期待ポートおよびノード・ポート設定に関するトラブルシューティング .....	4-24
4.10.1	クラスタのメンバまたはクォーラム・ディスクに障害が発生しクラスタがクォーラムを失った後にクラスタに参加させる .....	4-25
4.10.2	メンバがブートとクラスタ形成に必要な数のポートを持っていない場合のクラスタ形成 .....	4-30

## 5 クラスタ・メンバの管理

5.1	構成変数の管理 .....	5-2
5.2	カーネル属性の管理 .....	5-4
5.3	クラスタ内およびクラスタからのリモート・アクセスの管理 .....	5-6
5.4	クラスタのシャットダウン .....	5-7
5.5	クラスタの特定のメンバのシャットダウンと起動 .....	5-8
5.5.1	重要な投票メンバの識別 .....	5-8
5.5.2	重要な投票メンバの停止または削除の準備 .....	5-9
5.5.3	重要でないメンバの停止 .....	5-9
5.5.4	ホスト・メンバのシャットダウン .....	5-10

5.6	クラスタ・メンバをシャットダウンしてシングルユーザ・モードにする .....	5-10
5.7	クラスタ・メンバのリブート .....	5-12
5.8	クラスタからのメンバの削除 .....	5-12
5.9	クラスタからのメンバの削除とスタンドアロン・システムとしての復元 .....	5-15
5.10	別の IP サブネットへのクラスタの移動 .....	5-16
5.10.1	ファイル編集に関する表 .....	5-20
5.10.1.1	外部 IP アドレスだけの変更 .....	5-20
5.10.1.2	外部 IP アドレスとホスト名の変更 .....	5-21
5.10.1.3	外部および内部 IP アドレスとホスト名の変更 .....	5-22
5.10.2	属性のチェックリスト表 .....	5-24
5.10.2.1	外部ホスト名と IP アドレス .....	5-24
5.10.2.2	クラスタ名とクラスタ別名 .....	5-25
5.10.2.3	インタフェース IP 別名 .....	5-25
5.10.2.4	外部サーバ .....	5-26
5.10.2.5	チェックリスト .....	5-27
5.10.3	正常性の検証 .....	5-28
5.10.4	トラブルシューティング .....	5-29
5.11	クラスタ名または IP アドレスの変更 .....	5-30
5.11.1	クラスタ名の変更 .....	5-30
5.11.2	クラスタの IP アドレスの変更 .....	5-32
5.12	メンバ名, IP アドレス, クラスタのインターコネクト・アドレスの変更 .....	5-32
5.12.1	メンバ削除前のクォーラムのチェック .....	5-33
5.12.2	メンバ削除前の CAA の restricted 配置ポリシーのチェック .....	5-33
5.12.3	メンバの削除と追加 .....	5-34
5.13	ソフトウェア・ライセンスの管理 .....	5-35
5.14	レイヤード・アプリケーションのインストールと削除 .....	5-35

5.15	課金サービスの管理 .....	5-36
<b>6</b>	<b>クラスタ内のネットワーク管理</b>	
6.1	ネットワーク・インタフェースの変更時の ifaccess.conf の更新 .....	6-1
6.2	ネットワーク・インタフェースのフェイルオーバ .....	6-3
6.3	IP ルータの実行 .....	6-4
6.4	ネットワークの構成 .....	6-5
<b>7</b>	<b>ネットワーク・サービスの管理</b>	
7.1	DHCP の構成 .....	7-1
7.2	NIS の構成 .....	7-3
7.2.1	クラスタ内の NIS マスタへのエンハンスド・セキュリティ機能の追加 .....	7-4
7.3	印刷の構成 .....	7-4
7.4	DNS/BIND の構成 .....	7-6
7.5	時刻同期の管理 .....	7-7
7.5.1	NTP 構成 .....	7-8
7.5.2	すべてのメンバでの同一外部 NTP サーバの使用 .....	7-8
7.5.2.1	時間のずれ .....	7-9
7.6	NFS の管理 .....	7-9
7.6.1	NFS の構成 .....	7-9
7.6.2	クラスタ内での NFS の使用上の考慮事項 .....	7-12
7.6.2.1	CFS での NFS ファイル・システムのサポート .....	7-12
7.6.2.2	クライアントはクラスタ別名を使用しなければならない .....	7-13
7.6.2.3	CDSL を使って NFS ファイル・システムをマウントする .....	7-13
7.6.2.4	ループバック・マウントはサポートされていない ....	7-15

7.6.2.5	NFS マウントしたパスに非 NFS ファイル・システム をマウントしてはならない .....	7-15
7.6.3	クラスタでの AutoFS の使用 .....	7-15
7.6.3.1	ファイル・システムの強制アンマウント .....	7-17
7.6.3.1.1	強制アンマウントが必要かどうかの判断 .....	7-17
7.6.3.1.2	問題の解決 .....	7-18
7.6.4	Automount から AutoFS への移行 .....	7-20
7.6.4.1	クラスタ・メンバのリブートなしの移行 .....	7-20
7.6.4.2	クラスタ・メンバをリブートする移行 .....	7-23
7.6.4.3	クラスタをリブートする移行 .....	7-27
7.7	inetd の構成の管理 .....	7-29
7.8	メール・サービスの管理 .....	7-30
7.8.1	メール・サービスの構成 .....	7-31
7.8.1.1	メール・ファイル .....	7-31
7.8.1.2	Cw マクロ (システム・ニックネーム・リスト) .....	7-32
7.8.1.3	クラスタ作成時のメール・サービスの構成 .....	7-32
7.8.1.4	クラスタ稼働後のメール・サービスの構成 .....	7-33
7.8.2	クラスタ・メンバ間でのメールの負荷分散 .....	7-33
7.9	RIS 用のクラスタの構成 .....	7-35
7.10	リモートへの X Window アプリケーションの表示 .....	7-37

## 8 高可用性アプリケーションの管理

8.1	リソースの状態の調査 .....	8-3
8.1.1	リソースの状態の調査 .....	8-5
8.1.2	特定のクラスタ・メンバ上のすべてのリソースの調査 ....	8-6
8.1.3	すべてのクラスタ・メンバ上のすべてのリソースの調査 .	8-7
8.1.4	障害回数と再起動回数およびターゲット状態の調査 .....	8-8
8.2	リソースの可用性計測レポート .....	8-9
8.3	アプリケーションの再配置 .....	8-11



8.3.1	クラスタ・メンバ上のすべてのアプリケーションの手動再配置 .....	8-11
8.3.2	特定のアプリケーションの手動再配置 .....	8-12
8.3.3	依存関係にあるアプリケーションの手動再配置 .....	8-12
8.4	アプリケーション・リソースの起動と停止 .....	8-13
8.4.1	アプリケーション・リソースの起動 .....	8-13
8.4.2	アプリケーション・リソースの停止 .....	8-15
8.4.3	複数インスタンスが作成されないアプリケーション・リソース .....	8-16
8.4.4	caa_stop を使用した UNKNOWN 状態のリセット .....	8-16
8.5	アプリケーション・リソースの分散 .....	8-16
8.5.1	クラスタのすべてのアプリケーションを分散する .....	8-17
8.5.2	クラスタ・メンバのすべてのアプリケーションを分散する .....	8-17
8.5.3	指定されたアプリケーションを分散する .....	8-18
8.5.4	アプリケーションを特定の時刻で分散する .....	8-18
8.6	アプリケーション・リソースの登録と登録取り消し .....	8-19
8.6.1	リソースの登録 .....	8-19
8.6.2	リソースの登録取り消し .....	8-20
8.6.3	登録の更新 .....	8-20
8.7	ネットワーク, テープ, およびメディア・チェンジャ・リソース .....	8-21
8.8	SysMan を使用した CAA 管理 .....	8-22
8.8.1	SysMan Menu での CAA 管理 .....	8-22
8.8.1.1	[CAA Management] ダイアログ・ボックス .....	8-23
8.8.1.2	[Start] ダイアログ・ボックス .....	8-24
8.8.1.3	[Setup] ダイアログ・ボックス .....	8-25
8.8.2	SysMan Station での CAA 管理 .....	8-26
8.8.2.1	SysMan Station でのアプリケーションの起動 .....	8-28
8.8.2.2	SysMan Station でのリソース設定 .....	8-28

8.9	起動時およびシャットダウン時の CAA に関する考慮事項 ....	8-29
8.10	CAA デーモンの管理 .....	8-30
8.10.1	ローカルの CAA デーモンの状態の確認 .....	8-30
8.10.2	CAA デーモンの再起動 .....	8-31
8.10.3	CAA デーモンのメッセージのモニタ .....	8-31
8.11	EVM による CAA のイベントの表示 .....	8-31
8.11.1	CAA のイベントの表示 .....	8-32
8.11.2	CAA のイベントのモニタ .....	8-34
8.12	イベントに関するトラブルシューティング .....	8-34
8.13	コマンド行メッセージのトラブルシューティング .....	8-35
<b>9</b>	<b>ファイル・システムとデバイスの管理</b>	
9.1	CDSL の使用 .....	9-2
9.1.1	CDSL の作成 .....	9-2
9.1.2	CDSL の保守 .....	9-3
9.1.3	カーネル構築と CDSL .....	9-4
9.1.4	CDSL のエクスポートとマウント .....	9-4
9.2	デバイスの管理 .....	9-4
9.2.1	デバイス・スペシャル・ファイルの管理 .....	9-5
9.2.2	デバイス位置の確定 .....	9-5
9.2.3	クラスタへのディスクの追加 .....	9-7
9.2.4	他社製ストレージの管理 .....	9-8
9.2.5	テープ装置 .....	9-9
9.2.6	クラスタでのディスクットのフォーマット .....	9-11
9.2.7	CD-ROM と DVD-ROM .....	9-11
9.3	クラスタ・ファイル・システムの管理 .....	9-11
9.3.1	ファイル・システムがフェイルオーバーできない場合 .....	9-12
9.3.2	ダイレクト・アクセス・キャッシュド・リード .....	9-13
9.3.3	CFS の負荷分散 .....	9-14

9.3.3.1	cfspd の起動と停止 .....	9-17
9.3.3.2	EVM イベント .....	9-18
9.3.3.3	/etc/cfsd.conf 構成ファイルの変更 .....	9-18
9.3.3.4	cfspd 分析の理解と実装に関する推奨事項 .....	9-21
9.3.3.5	自動再配置 .....	9-23
9.3.3.6	CAA リソースとの関連性 .....	9-24
9.3.3.7	cfspd 以外での CFS の負荷分散 .....	9-24
9.3.3.8	cfsmgr による CFS サーバの負荷分散 .....	9-27
9.3.4	mount -o コマンドによるファイル・システムの分散 ....	9-28
9.3.5	クローン作成前のデーモンのフリーズ .....	9-29
9.3.5.1	ドメインがフリーズされているかどうかの判断 .....	9-30
9.3.6	CFS 性能の最適化 .....	9-31
9.3.6.1	先読みおよび後書きスレッド数の変更 .....	9-31
9.3.6.2	直接入出力の利用 .....	9-32
9.3.6.2.1	クラスタとスタンドアロン AdvFS 直接入出力の違い .....	9-33
9.3.6.2.2	直接入出力モードでオープンしたファイルを持つ ファイルセットのクローンを作成する .....	9-33
9.3.6.2.3	直接入出力に関する統計情報の収集 .....	9-34
9.3.6.3	CFS メモリ使用量の調整 .....	9-37
9.3.6.4	メモリ・マップ・ファイルの使用 .....	9-40
9.3.6.5	フル・ファイル・システムの回避 .....	9-40
9.3.6.6	その他の方法 .....	9-40
9.3.7	MFS および UFS ファイル・システムのサポート .....	9-41
9.3.8	ファイル・システムのパーティショニング .....	9-42
9.3.9	ブロック・デバイスとキャッシュの整合性 .....	9-44
9.3.10	CFS の制限 .....	9-45
9.4	デバイス要求ディスパッチャの管理 .....	9-46
9.4.1	ダイレクト・アクセス入出力と単一サーバ・デバイス ....	9-46

9.4.1.1	ダイレクト・アクセス入出力をサポートするデバイス	9-49
9.4.1.2	ダイレクト・アクセス入出力ディスクとしての RZ26 , RZ28 , RZ29 , RZ1CB-CA のうちいずれか 1 つの交換	9-50
9.4.1.3	共用バス上の HSZ ハードウェアのサポート .....	9-51
9.5	クラスタでの AdvFS の管理 .....	9-51
9.5.1	新たに追加されたメンバからの AdvFS ファイルの統合 ..	9-51
9.5.2	クラスタ・ルート・ドメインにファイルセットを 1 つだけ 作成する .....	9-52
9.5.3	メンバのブート・パーティションへファイルセットを追加 する (非推奨) .....	9-52
9.5.4	メンバのルート・ドメインへボリュームを追加しない ....	9-53
9.5.5	クラスタでの addvol コマンドと rmvol コマンドの使用	9-53
9.5.6	ユーザとグループのファイル・システムに対するクォータ のサポート .....	9-55
9.5.6.1	クォータのハード限界値 .....	9-56
9.5.6.2	quota_excess_blocks の値の設定 .....	9-57
9.5.7	ストレージの接続性と AdvFS ボリューム .....	9-57
9.6	新しいファイル・システムを作成するときの注意事項 .....	9-58
9.6.1	ディスクの接続性の確認 .....	9-59
9.6.2	利用可能なディスクの調査 .....	9-60
9.6.2.1	クォーラム・ディスクの場所の調査 .....	9-60
9.6.2.2	メンバ・ブート・ディスクとクラスタ単位の AdvFS ファイル・システムの場所の調査 .....	9-60
9.6.2.3	メンバ・スワップ領域の調査 .....	9-62
9.6.3	/etc/fstab の編集 .....	9-62
9.7	CDFS ファイル・システムの管理 .....	9-63
9.8	ファイルのバックアップとリストア .....	9-63
9.8.1	バックアップするファイルに対する推奨事項 .....	9-64
9.9	スワップ領域の管理 .....	9-65
9.9.1	性能向上のためのスワップ・デバイスの設定 .....	9-66

9.10	ブート・パラメータによる問題の修正 .....	9-66
9.11	クラスタでの verify ユーティリティの使用 .....	9-67
9.11.1	クラスタ・ルートでの verify ユーティリティの使用 .....	9-68
<b>10</b>	<b>クラスタでの LSM (Logical Storage Manager) の使用</b>	
10.1	クラスタとスタンドアロン・システムにおける LSM の管理の 違い .....	10-2
<b>11</b>	<b>クラスタのトラブルシューティング</b>	
11.1	問題の解決 .....	11-1
11.1.1	ライセンスなしのシステムのブート .....	11-1
11.1.2	メンバが動作したままになるシャットダウン .....	11-1
11.1.3	環境のモニタリングとクラスタのシャットダウン .....	11-2
11.1.4	ブート時の CFS エラーの処理 .....	11-4
11.1.5	メンバのブート・ディスクのバックアップと修復 .....	11-4
11.1.5.1	メンバのブート・ディスクの修復例 .....	11-6
11.1.6	ブート時の cluster_root の指定 .....	11-8
11.1.7	クラスタの既存のディスクへのクラスタ・ルート・ファイ ル・システムの回復 .....	11-10
11.1.8	新しいディスクへのクラスタ・ルート・ファイル・システ ムの回復 .....	11-13
11.1.9	AdvFS の問題の処理 .....	11-19
11.1.9.1	addvol または rmvol からの警告メッセージに対する応 答 .....	11-19
11.1.9.2	デバイスの接続障害による AdvFS ドメイン・パニック の解消 .....	11-19
11.1.9.3	AdvFS ファイル・システムまたはドメインの強制アン マウント .....	11-20
11.1.9.4	ドメイン・パニックの回避 .....	11-22

11.1.10	停止したシステム上のブート・パーティションへのアクセス .....	11-22
11.1.11	ブート・ディスクがすでにマウントされているメンバのブート .....	11-23
11.1.12	クラッシュ・ダンプの生成 .....	11-23
11.1.12.1	メンバがハングしたときのクラッシュ・ダンプの生成 .....	11-24
11.1.13	ネットワーク問題の修正 .....	11-24
11.1.14	クラスタでの routed の実行 .....	11-27
11.2	クラスタ管理のヒント .....	11-27
11.2.1	/tmp の移動 .....	11-28
11.2.2	MC_CABLE コンソール・コマンドの実行 .....	11-29
11.2.3	Korn シェルにおけるメンバ固有のディレクトリへの実パスの非表示 .....	11-29
 <b>A クラスタ・イベント</b>		
 <b>B 構成変数</b>		
 <b>C clu_delete_member ログ</b>		
 <b>D LAN クラスタの管理</b>		
D.1	クラスタ LAN インターコネクトに使用する NetRAIN 仮想インタフェースの構成 .....	D-1
D.2	LAN インターコネクトのチューニング .....	D-3
D.2.1	ipmtu 値の設定によるクラスタ・インターコネクトの性能改善 .....	D-4
D.2.2	イーサネット・スイッチ・アドレス・エイジングに 15 秒を設定する .....	D-5
D.3	ネットワーク・アダプタ構成情報の取得 .....	D-5
D.4	LAN インターコネクト上の動作の監視 .....	D-6
D.5	Memory Channel から LAN への移行 .....	D-7

D.6	LAN から Memory Channel への移行 .....	D-11
D.7	高速イーサネット LAN からギガビット・イーサネット LAN への移行 .....	D-11
D.8	トラブルシューティング .....	D-13
D.8.1	新しいメンバをブートすると、ブロードキャスト・エラー が多発して既存メンバがパニックに陥る .....	D-13
D.8.2	NetRAIN 仮想インタフェースのデバイスを手動でフェイ ルオーバできない .....	D-14
D.8.3	アプリケーションをポートにマップできない .....	D-15

## 索引

### 例

2-1	sysman の出力例 .....	2-21
9-1	/etc/cfsd.conf ファイル .....	9-19

### 図

2-1	SysMan Menu の階層 .....	2-7
2-2	SysMan Menu のインタフェース .....	2-8
2-3	SysMan Station のグラフィカル・インタフェース .....	2-9
2-4	SysMan Station の最初のクラスタ・ビュー .....	2-14
2-5	SysMan Station のクラスタ・ハードウェア・ビューのサンプ ル .....	2-16
2-6	SysMan Station での有効なアクションの表示 .....	2-17
2-7	HP Insight Manager の表示 .....	2-23
2-8	構成レポート表示のサンプル .....	2-25
4-1	3 メンバ構成のクラスタ deli .....	4-10
4-2	3 メンバ構成のクラスタ deli が 1 つのメンバを失った場合 ...	4-12
4-3	2 メンバ構成のクラスタ deli にクォーラム・ディスクがない場 合 .....	4-14

4-4	2 メンバ 構成のクラスタ deli が 1 つのメンバを失ってもクォーラム・ディスクがあるために稼働し続ける場合 .....	4-15
8-1	SysMan Menu の CAA へのブランチ .....	8-23
8-2	[CAA Management] ダイアログ・ボックス .....	8-24
8-3	[Start] ダイアログ・ボックス .....	8-25
8-4	[Setup] ダイアログ・ボックス .....	8-26
8-5	SysMan Station の CAA_Applications_(active) ビュー .....	8-27
8-6	SysMan Station の CAA_Applications_(all) ビュー .....	8-28
8-7	SysMan Station の [CAA Setup] 画面 .....	8-29
9-1	SysMan Station でのハードウェア構成の表示 .....	9-7
9-2	セミ・プライベート・ストレージを持つクラスタ .....	9-10
9-3	4 ノード・クラスタ .....	9-47

## 表

1-1	クラスタ用コマンド .....	1-2
1-2	ファイル・システムおよびストレージ管理に関する相違点 ...	1-5
1-3	ネットワーク管理に関する相違点 .....	1-9
1-4	プリント管理に関する相違点 .....	1-13
1-5	セキュリティ管理に関する相違点 .....	1-14
1-6	一般のシステム管理に関する相違点 .....	1-15
1-7	サポートされていない機能 .....	1-19
2-1	クラスタ・ツールのクイック・スタート .....	2-2
2-2	利用可能な管理ツールとインタフェース .....	2-4
2-3	クラスタ管理ツール .....	2-5
2-4	SysMan Menu の起動 .....	2-12
2-5	SysMan Station の起動 .....	2-17
4-1	2 ~ 4 メンバ構成のクラスタ内でのメンバの cluster_expected_votes 設定とボートの割り当てによる影響 ..	4-21
4-2	メンバやクォーラム・ディスクに障害が発生したクラスタの クォーラム喪失の解決例 .....	4-29



4-3	クラスタを形成するのに十分なポートを持つメンバのブートによるクォーラム不足の解決例 .....	4-32
5-1	/etc/rc.config* ファイル .....	5-2
5-2	削減できないカーネル属性 .....	5-4
5-3	構成可能な TruCluster Server サブシステム .....	5-5
8-1	アプリケーション・リソースのターゲットと状態の組み合わせ .....	8-4
8-2	ネットワーク・リソースのターゲットと状態の組み合わせ ...	8-4
8-3	テープ、メディア・チェンジャ・リソースのターゲットと状態の組み合わせ .....	8-5
9-1	ストレージ・デバイス管理の参考マニュアル .....	9-1
11-1	メンバ・ブート・ディスク・パーティション .....	11-5
B-1	クラスタ構成変数 .....	B-1



---

## まえがき

本書では、HP TruCluster Server システムの日常の運用に関連する作業について説明します。クォーラムおよびポートの管理とクラスタ・ファイル・システム (CFS) について説明するとともに、デバイス要求ディスパッチャの管理方法とクラスタ内のネットワーク・サービスの構成方法についても説明します。さらに、クラスタ管理のためのヒントも提供します。

### 対象読者

本書は、TruCluster Server クラスタを構成および管理するユーザを対象としています。本書では、読者が経験を積んだ UNIX 管理者であり、ハードウェア、オペレーティング・システム、およびネットワークの構成と保守について十分な知識があることを前提としています。

### 新しい機能と変更された機能

本書 (および関連するマニュアル) でバージョン 5.1A のリリースから変更された点は次のとおりです。

- バージョン 5.1A の『クラスタ LAN インターコネクト』は、TruCluster Server ドキュメント・セットから削除され、このマニュアルの必要な情報は、本書、『クラスタ概要』、『クラスタ・ハードウェア構成ガイド』、および『クラスタ・インストール・ガイド』に組み込まれています。
- 3.14 節では、ルーティング・オプション設定について説明しています。バージョン 5.1B からは、`gated`、`routed`、または静的ルーティングが使えるようになります。`ogated` は使えなくなりました。`gated` が省略時の設定です。
- 5.8 節では、警告欄が変更され、メンバ削除後の期待ポートを調整する必要性を説明しています。
- 5.10 節では、クラスタを別の IP サブネットへ移動する方法を説明しています。

- 6.1 節が新しく追加され、ネットワーク・インタフェースを変更したときの `ifaccess.conf` ファイルの更新について説明しています。
- 7.4 節が更新され、BIND クライアントと BIND サーバとの違いが明確にされました。
- 7.6.4 項では、Automount から AutoFS へ移行する方法について説明しています。
- 8.5 節では、新しい `caa_balance` コマンドによってアプリケーション・リソースを均等に分散する方法を説明しています。
- 9.3.3 項では、新しい CFS 負荷分散 (`cfstd`) 機能を使用して、クラスタのファイル・システムを監視、解析および調整する方法を説明しています。
- 9.3.4 項では、分散ファイル・システムの `mount -o` コマンドの使い方を説明しています。
- 9.3.5 項では、一貫したファイル・システムのメタデータを保つために、クローン作成の前に新しい `freezefs` コマンドでドメインをフリーズする方法を説明しています。
- 第 10 章では、HP Tru64 UNIX と TruCluster Server の LSM 処理の相違点だけを説明しています。LSM 情報が容易に探し出せるように、すべての LSM に関する情報は Tru64 UNIX 『*Logical Storage Manager*』に集められています。
- 11.1.3 項では、メンバをシャットダウンすると、クォーラムを失うことになるのかどうか、もしそうであれば、残りのクラスタ・メンバをシャットダウンするのかどうか、を決定する `ENVMON_SHUTDOWN_SCRIPT` スクリプトを使う方法を説明しています。
- 11.1.9.3 項では、サービスされているドメインを `cfsmgr` コマンドの新しい `-U` オプションを使って強制的にアンマウントする方法を説明しています。
- 付録 D では、LAN インターコネクトを使用するクラスタの管理方法を説明しています。

## 本書の構成

本書の次のように構成されています。

第 1 章	スタンドアロンの Tru64 UNIX オペレーティング・システム・ソフトウェアの管理と TruCluster Server の管理との違いについて説明します。
第 2 章	クラスタ管理用のグラフィカル・ユーザ・インタフェース (GUI) とコマンド行ツールについて説明します。
第 3 章	ネットワーク・アプリケーションから単一システムとしてクラスタを参照するためのクラスタ別名サブシステムの使用方法について説明します。
第 4 章	クラスタの可用性を維持するために、クォーラムおよびポートを管理する方法について説明します。
第 5 章	クラスタ・メンバの構成、管理、および削除方法について説明します。
第 6 章	クラスタ内のメンバおよびクライアント・ネットワークの構成および管理方法について説明します。
第 7 章	クラスタ内のメール、プリント、およびその他のサービスの構成方法について説明します。高可用性ネットワーク・サービスの提供方法についても説明します。
第 8 章	高可用性アプリケーションの管理に関連する日常の作業について説明します。
第 9 章	クラスタ・ファイル・システムとデバイス要求ディスクパッチャの管理方法について説明します。ストレージ・デバイスの追加および削除方法と、ディスク・サーバの負荷分散方法についても説明します。
第 10 章	クラスタ内での LSM (Logical Storage Manager) ソフトウェアの使用に関する重要な相違点について説明します。
第 11 章	TruCluster Server に起こりやすい問題の調査および解決方法について説明します。
付録 A	TruCluster Server システムに固有のイベントを一覧に示します。
付録 B	TruCluster Server で提供されている構成変数を一覧に示します。
付録 C	クラスタ・メンバの削除ログ <code>/cluster/admin/clu_delete_member.log</code> の例を示します。
付録 D	LAN のクラスタ管理について説明します。

## 関連資料

クラスタの構成、インストール、および管理作業には、TruCluster Server 関連の次のマニュアルも参考になります。

- 『*TruCluster Server QuickSpecs*』 — TruCluster Server バージョン 5.1B 製品についてのわかりやすい説明が記載されています。  
『*QuickSpecs*』の最新版は、次の URL から入手することができます。  
[http://tru64unix.compaq.com/docs/pub\\_page/spds.html](http://tru64unix.compaq.com/docs/pub_page/spds.html).
- 『*クラスタ概要*』 — TruCluster Server システムの効果的な管理に必要な概念について説明しています。このマニュアルはクラスタのインストール前に参照してください。
- 『*クラスタ・リリース・ノート*』 — TruCluster Server ソフトウェア製品の新しい機能、制限事項、およびその他の重要な情報を記載しています。
- 『*クラスタ・ハードウェア構成ガイド*』 — クラスタのメンバとして追加するプロセッサの構成方法と、クラスタの共用ストレージの構成方法について説明しています。
- 『*クラスタ・インストレーション・ガイド*』 — クラスタに参加するシステムに TruCluster Server ソフトウェアをインストールする方法について説明しています。
- 『*クラスタ高可用性アプリケーション・ガイド*』 — アプリケーションを高可用性アプリケーションにする方法について説明しています。分散ロック・マネージャ (DLM) で提供されているアプリケーション・プログラミング・インタフェース (API)、クラスタ別名、および Memory Channel サブシステムについても説明しています。

TruCluster Server に関するドキュメントの最新版は、次の URL から入手することができます。

[http://www.tru64unix.compaq.com/docs/pub\\_page/cluster\\_list.html](http://www.tru64unix.compaq.com/docs/pub_page/cluster_list.html)

TruCluster Server システムの管理と Tru64 UNIX バージョン 5.1B システムの管理は良く似ています。そのため、Tru64 UNIX バージョン 5.1B オペレーティング・システム関連の次のマニュアルも参考になります。

- 『*リリース・ノート*』
- 『*システム管理ガイド*』
- 『*ネットワーク管理ガイド：接続編*』
- 『*ネットワーク管理ガイド：サービス編*』

- 『*Logical Storage Manager*』
- 『*AdvFS* 管理ガイド』
- 『システムの構成とチューニング』
- 『セキュリティ管理ガイド』
- 『プログラミング・ガイド』
- 『*Sharing Software on a Local Area Network*』
- *Advanced Printing Software* 『リリース・ノート』

ストレージの管理には、次のマニュアルが参考になります。

- POLYCENTER Advanced File System Utilities 『*Reference Manual*』
- POLYCENTER Advanced File System Utilities 『*Release Notes*』
- POLYCENTER Advanced File System Utilities 『*Installation Guide*』

## 本書の表記法

本書では次の表記法を使用します。

#	番号記号は root としてログインした場合のシステム・プロンプトを表します。
% cat	対話式の例における太字(ボールド体)は、ユーザが入力する文字を示します。
<i>file</i>	イタリック体(斜体)は、変数値、プレースホルダ、および関数の引数名を示します。
⋮	垂直の反復記号は、実際には存在する例の一部が省略されていることを示します。
cat(1)	リファレンス・ページの参照には、該当するセクション番号をカッコ内に示します。たとえば、cat(1) は、cat コマンドについての情報が、リファレンス・ページのセクション 1 に記載されていることを示します。

Return

四角で囲まれたキー名はユーザがそのキーを押すことを示します。



---

## クラスタ管理の概要

TruCluster Server クラスタの管理はスタンドアロン Tru64 UNIX システムの管理に似ています。システム管理用の 600 を超えるコマンドとユーティリティのうち、20 未満のコマンドとユーティリティがクラスタ管理専用を用意されています。これらのコマンドの大部分を使い、クラスタを作成したり、クラスタに新規メンバを追加したり、アプリケーションを高可用性アプリケーションとして構成したりします。Tru64 UNIX システムの管理に関する知識で、TruCluster Server クラスタの管理のほとんどが可能です。

本書では、クラスタの管理に特有の状況について説明します。その他の状況での管理手順については、Tru64 UNIX 『システム管理ガイド』を参照してください。

本書をさらに読み進む前に、TruCluster Server 『クラスタ概要』の内容をよく理解しておいてください。クラスタの管理には、このマニュアルの内容を理解しておく必要があります。

この章では、次の事項について説明します。

- クラスタ用のコマンドとユーティリティ (1.1 節)
- クラスタで異なる動作をするコマンドと機能 (1.2 節)

多くの状況で、TruCluster Server の次のような項目を管理する必要から、管理対象がスタンドアロン・システムではなくクラスタであることが明白になります。

- クラスタの作成と構成：クラスタの初期メンバの作成、メンバの追加と削除、クラスタ構成の照会を行う。
- CAA (Cluster Application Availability) サブシステム：高可用性アプリケーションの定義および管理を行う。
- クラスタ別名：ネットワーク上のクライアントから単一システムとしてクラスタを参照可能にする。

- クラスタのクォーラムとポート：クラスタの構成要素とメンバシップを定義する。メンバシップに従ってクラスタ・リソースへのアクセスを許可する。
- デバイス要求ディスパッチャ：クラスタ内のすべてのデバイスへの透過的かつ高可用性のアクセス手段を提供する。
- クラスタ・ファイル・システム (CFS)：ルート (/) ファイル・システムなど、クラスタ全体のすべてのファイル・システムへの一貫したアクセス手段を提供する。
- クラスタ・インターコネクト：クラスタ単位でメンバ間のプライベート通信パスのインターコネクト機能を提供する。

ただし、ユーザからクラスタが1つのコンピュータ・システムのように見えないときには、コマンド・レベルでいくつかの例外が発生します。たとえば、`wall` コマンドを実行すると、実行元のクラスタ・メンバのログインしているユーザのみにメッセージが送信されます。すべてのクラスタ・メンバのログインしているユーザすべてにメッセージを送信するには、`wall -c` コマンドを使用する必要があります。

## 1.1 クラスタ用のコマンドとユーティリティ

表 1-1 に TruCluster Server システムの管理に固有のコマンドを示します。これらのコマンドは、クラスタの該当項目の操作または照会に使用します。これらのコマンドについては、リファレンス・ページに解説があります。

表 1-1: クラスタ用コマンド

用途	コマンド	機能
クラスタのメンバの作成と構成	<code>clu_create(8)</code>	Tru64 UNIX システム上でクラスタの最初のメンバを作成する。
	<code>clu_add_member(8)</code>	クラスタにメンバを追加する。
	<code>clu_delete_member(8)</code>	クラスタからメンバを削除する。
	<code>clu_check_config(8)</code>	TruCluster Server が適切にインストールされているか、クラスタが正しく構成されているかを確認する。
	<code>clu_get_info(8)</code>	クラスタとそのメンバに関する情報を表示する。
高可用性アプリケーションの定義と管理	<code>caad(8)</code>	CAA デーモンを起動する。

### 1-2 クラスタ管理の概要

表 1-1: クラスタ用コマンド (続き)

用途	コマンド	機能
	caa_profile(8)	アプリケーションの可用性プロファイルを管理し、基本的な構文チェックを実行する。
	caa_balance(8)	アプリケーション・リソースをメンバ間で均等に分散する。
	caa_register(8)	CAA デーモンにアプリケーションを登録する。
	caa_relocate(8)	クラスタのあるメンバから高可用性アプリケーションを別のメンバに手動で再配置する。
	caa_report(8)	アプリケーション・リソースに対する可用性統計情報をレポートする。
	caa_start(8)	CAA デーモンに登録された高可用性アプリケーションを起動する。
	caa_stat(1)	CAA デーモンに登録されたアプリケーションの状態を表示する。
	caa_stop(8)	高可用性アプリケーションを停止する。
	caa_unregister(8)	高可用性アプリケーションの登録を取り消す。
クラスタ別名の管理	cluamgr(8)	クラスタ別名を作成および管理する。
	clua_active(8)	クラスタ別名がアクティブで到達可能かどうかを決定する。
クォーラムとポートの管理	clu_quorum(8)	クォーラム・ディスクを構成または削除する。クォーラム・ディスク・ポート、ノード・ポート、または期待ポートを調節する。
コンテキスト依存シンボリック・リンク (CDSL) の管理	mkcdsl(8)	CDSL を作成またはチェックする。
デバイス要求ディスパッチの管理	drdmgr(8)	分散デバイスの属性を取得または設定する。
クラスタ・ファイル・システム (CFS) の管理	cfsmgr(8)	クラスタにマウントされたファイル・システムを管理する。

表 1-1: クラスタ用コマンド (続き)

用途	コマンド	機能
Memory Channel の状態の照会	imcs(1)	Memory Channel アプリケーション・プログラミング・インタフェース (API) ライブラリ libimc の状態を報告する。
	imc_init(1)	現在のホスト上の Memory Channel API ライブラリ libimc を初期化および構成する。

## 1.2 クラスタで異なる動作をするコマンドと機能

次の表に示した Tru64 UNIX のコマンドとサブシステムは、クラスタ固有のオプションを持ち、クラスタとスタンドアロン Tru64 UNIX システムとは異なる動作をします。

一般に、プロセス管理用のコマンドはクラスタ対応ではないので、実行元のメンバの管理にしか使用できません。

表 1-2 に、ファイル・システムおよびストレージ管理用のコマンドおよびユーティリティの動作の違いを示します。

スタンドアロン Tru64 UNIX システムでは、ルート・ファイル・システム (/) は root\_domain#root です。クラスタでは、ルート・ファイル・システムは常に cluster\_root#root です。クラスタの各メンバのブート・パーティションは rootmemberID\_domain#root です。

たとえば、メンバ ID 6 のクラスタ・メンバ上では、ブート・パーティション (/cluster/members/member6/boot\_partition) は root6\_domain#root です。

表 1-2: ファイル・システムおよびストレージ管理に関する相違点

コマンド	相違点
addvol(8)	<p>スタンドアロン・システムでは、addvol を使って root_domain を拡張できない。しかしクラスタでは、addvol を使って cluster_root ドメインにボリュームを追加できる。</p> <p>rmvol コマンドを使って cluster_root ドメインからボリュームを削除することもできる。</p> <p>LSM (Logical Storage Manager) ボリュームは cluster_root ドメイン内では使用できない。addvol コマンドを使って cluster_root ドメインに LSM ボリュームを追加しようとすると、エラーが発生する。ただし、cluster_root ドメインでも、クラスタの他のファイル・システムのドメインでも、volmigrate コマンドを使用して LSM ボリュームに移行させることができる。詳細は、Tru64 UNIX 『<i>Logical Storage Manager</i>』を参照。</p>
bttape(8)	<p>bttape ユーティリティはクラスタ非対応。ファイルのバックアップと復元についての詳しい説明は、9.8 節を参照。</p>
df(1)	<p>df コマンドは、クライアントのキャッシュ内のデータを対象としない。クライアントのキャッシュ内のデータは、少なくとも 30 秒ごとにサーバと同期をとる。同期後、物理ファイル・システムはキャッシュ・データに対してストレージを割り当てる。</p>
iostat(1)	<p>iostat コマンドは、実行元のメンバに直接接続されている共用バスまたはプライベート・バス上のデバイスに関して統計情報を表示する。</p> <p>この統計情報はローカル・メンバに対して発生したトラフィックに関係する。</p>

表 1-2: ファイル・システムおよびストレージ管理に関する相違点 (続き)

コマンド	相違点
LSM 関連 voldisk(8) volencap(8) volreconfig(8) volstat(8) volmigrate(8) volunmigrate(8)	<p>LSM が制御していないディスク (つまり, autoconfig ディスク) に対して voldisk list コマンドを実行すると, メンバによっては結果が異なる場合がある。このような結果の違いは, 主に使用不能ディスク・グループに限定される。たとえば, あるメンバは使用不能ディスク・グループを示すが, 別のメンバは同じディスク・グループをまったく示さないという場合がある。また, クラスタ内で, voldisk list コマンドを実行すると, 実行元のメンバに直接接続されている非 LSM ディスクだけを示す。</p> <p>クラスタでは, あるクラスタ・メンバ上で volencap swap コマンドを実行すると, 1 つ以上のメンバのスワップ・デバイスが LSM ボリュームに置かれる。</p> <p>volreconfig コマンドは, メンバのスワップ・デバイスをカプセル化するときだけに必要である。カプセル化したいスワップ・デバイスがある場合は, そのスワップ・デバイスのある各メンバ上でこのコマンドを実行する必要がある。</p> <p>volstat コマンドは, 実行元メンバの統計情報のみを返す。</p> <p>volmigrate コマンドは AdvFS (Advanced File System) を変更して, LSM ボリュームをストレージで使えるようにする。cluster_root ドメインでも, クラスタの他のファイル・システムのドメインでも, volmigrate コマンドを使用して LSM ボリュームに移行させることができる。詳細は Tru64 UNIX 『<i>Logical Storage Manager</i>』を参照。volunmigrate は AdvFS (Advanced File System) を変更して, LSM ボリュームではなく物理ディスクをストレージで使えるようにする。</p> <p>クラスタで使用する LSM についての詳細は, 第 10 章を参照。</p>

表 1-2: ファイル・システムおよびストレージ管理に関する相違点 (続き)

コマンド	相違点
mount(8)	<p>ネットワーク・ファイル・システム (NFS) ループバック・マウントはサポートされていない。詳細は、7.6.2.4 項を参照。</p> <p>mountd 経由で実行されるその他のコマンド (たとえば umount や export) が外部クライアントから発行され、その際に省略時のクラスタ別名または /etc/exports.aliases の中で指定されている別名が使用されないと、「Program unavailable」というエラーが返される。</p>
<b>Prestoserve 関連</b> presto(8) dxpresto(8X) prestosetup(8) prestoctl_svc(8)	<p>Prestoserve はクラスタ非対応。</p>
showfsets(8)	<p>showfsets コマンドは、クライアントのキャッシュ内のデータを対象としない。クライアントのキャッシュ内のデータは、少なくとも 30 秒ごとにサーバと同期をとる。同期後、物理ファイル・システムはキャッシュ・データに対してストレージを割り当てる。</p> <p>クォータや物理ストレージの実量を超えてアクセスし、異常なデータをキャッシュすることを避けるために、ファイルセットのクォータ制限とストレージ制限を行う。</p>

表 1-2: ファイル・システムおよびストレージ管理に関する相違点 (続き)

コマンド	相違点
UNIX ファイル・システム (UFS) 関連 メモリ・ファイル・システム (MFS) 関連	<p>UFS ファイル・システムでは、接続を使った読み取りアクセスだけが可能である。メンバに障害が起こると、CFS はそのファイル・システムをサービスする新しいサーバを選ぶ。パスに障害が起こると、CFS はそのストレージに至る代わりのデバイス要求ディスパッチャを使用する。</p> <p>クラスタ・メンバは UFS ファイル・システムを読み取り/書き込みでマウントできる。そのファイル・システムにはそのメンバだけがアクセスできる。リモートからはアクセスできず、フェイルオーバーはできない。MFS ファイル・システムは、読み取り専用でマウントしたか読み取り/書き込みでマウントしたかに関係なく、それをマウントしたメンバだけがアクセスできる。MFS ファイル・システムと読み取り/書き込みの UFS ファイル・システムのサーバは、そのマウントを起動したメンバが担当する。</p>
verify(8)	<p>verify コマンドを使ってクラスタのルート・ドメインを調べることができる。ただし、-f および -d オプションは使用できない。</p> <p>詳細は、9.11.1 項を参照。</p>

表 1-3 に、ネットワーク管理用のコマンドおよびユーティリティの動作の違いを示します。



表 1-3: ネットワーク管理に関する相違点

コマンド	相違点
BIND (Berkeley Internet Name Domain) 関連 bindconfig(8) bindsetup(8) svcsetup(8)	<p>bindsetup コマンドは Tru64 UNIX バージョン 5.0 で廃止された。クラスタ内の BIND を構成するには、sysman dns コマンドまたはそれと同等の bindconfig コマンドを使用する。</p> <p>BIND クライアントの構成はクラスタ単位で行うので、クラスタのすべてのメンバは同じクライアント構成を使用する。</p> <p>クラスタの 1 つのメンバだけが BIND サーバになることができる。BIND サーバは CAA 制御下の高可用性サービスである。BIND サーバ名としてクラスタ別名が使用される。</p> <p>詳細は、7.4 節を参照。</p>
ブロードキャスト・メッセージ関連 wall(1) rwall(1)	<p>wall -c コマンドは、すべてのクラスタ・メンバのログインしているユーザすべてにメッセージを送信する。何もオプションを付けなければ、wall コマンドは、実行元のクラスタ・メンバのログインしているユーザのみにメッセージを送信する。</p> <p>rwall から省略時のクラスタ別名へのブロードキャスト・メッセージは、すべてのクラスタ・メンバのログインしているユーザすべてに送信される。</p> <p>クラスタでは、wall -c からのメッセージを受信するために、各メンバ上で clu_wall デモンが動作している。</p>
DHCP (Dynamic Host Configuration Protocol) 関連 joinc(8)	<p>クラスタは DHCP サーバになることができるが、クラスタのメンバは DHCP クライアントになることはできない。クラスタ内では joinc は実行禁止。クラスタのメンバは静的なアドレス指定を使用する必要がある。</p> <p>詳細は、7.1 節を参照。</p>

表 1-3: ネットワーク管理に関する相違点 (続き)

コマンド	相違点
dsfmgr(8)	<p>-a class オプションの使用時は, entry_type として c (クラスタ) を指定する。</p> <p>-s オプション使用時の出力には, デバイスの適用範囲として c (クラスタ) が表示される。</p> <p>オプション -o および -O (旧形式のデバイス・スペシャル・ファイルを作成) はクラスタ内では無効。</p>
メール・サービス関連 mailconfig(8) mailsetup(8) mailstats(8)	<p>メール・サービスを実行するメンバは同じメール構成を使用しなければならない。したがって, これらのメンバでは同じプロトコルを有効にする必要がある。また, これらのメンバをメール・クライアントまたはメール・サーバのいずれかに構成しなければならない。詳細は, 7.8 節を参照。</p> <p>mailstats コマンドは, 実行元のクラスタ・メンバのみに関するメールの統計情報を返す。メールの統計情報ファイル /usr/adm/sendmail/sendmail.st はメンバに固有である。つまり, クラスタの各メンバはメールに関してそれぞれ固有の統計情報ファイルを持っている。</p>
ネットワーク・ファイル・システム (NFS) 関連 nfsconfig(8) rpc.lockd(8) rpc.statd(8)	<p>NFS を構成するには, sysman nfs コマンドまたは nfsconfig コマンドを使用する。nfssetup コマンドは使用禁止 (Tru64 UNIX バージョン 5.0 でサポート終了)。</p> <p>クラスタのメンバは NFS クライアント・デーモン lockd および statd を実行できる。ただし, NFS サーバ・デーモン lockd および statd を実行できるのは, クラスタの 1 つのメンバのみである。サーバ・デーモン lockd および statd は CAA 制御下の高可用性デーモンである。</p> <p>詳細は, 7.6 節を参照。</p>

表 1-3: ネットワーク管理に関する相違点 (続き)

コマンド	相違点
<p>ネットワーク管理関連</p> <p>netconfig(8)</p> <p>netsetup(8)</p> <p>gated(8)</p> <p>routed(8)</p>	<p>クラスタ構成中にネットワークを構成した場合, <code>gated</code> をルーティング・デーモンとして構成することを推奨する。詳細は, TruCluster Server 『クラスタ・インストール・ガイド』を参照。</p> <p>TruCluster Server バージョン 5.1B より前のリリースでは, クラスタのルーティング・デーモンとして <code>gated</code> を実行する必要がある。 <code>routed</code>, <code>ogated</code>, または静的ルーティングのセットアップが使用できず, また, ルーティング・デーモンもすべて使用できない。バージョン 5.1B からは, <code>gated</code>, <code>routed</code>, または静的ルーティングが使用可能になる。 <code>ogated</code> は使用不可。 <code>gated</code> が省略時の設定である。ルーティング・オプションについては 3.14 節を参照。</p> <p><code>netsetup</code> コマンドは廃止されたので使用不可。</p>
<p>NIFF (Network Interface Failure Finder) 関連</p> <p>niffconfig(8)</p> <p>niffd(8)</p>	<p>NIFF がクラスタ内のネットワーク・インタフェースを監視するには, <code>niffd</code> (NIFF デーモン) がクラスタの各メンバ上で動作していなければならない。詳細は, 6.2 節を参照。</p>
<p>NIS (ネットワーク情報サービス) 関連</p> <p>nissetup(8)</p>	<p>NIS は高可能性アプリケーションとして動作する。NIS マスタの識別には省略時のクラスタ別名が使用される。</p> <p>詳細は, 7.2 節を参照。</p>

表 1-3: ネットワーク管理に関する相違点 (続き)

コマンド	相違点
NTP (Network Time Protocol) 関連 ntp(1)	<p>クラスタのすべてのメンバの時刻同期には NTP が使用される。</p> <p>各クラスタ・メンバはそれぞれ他のメンバの NTP ピアとして自動的に構成されるので、特別な NTP の構成は不要。</p> <p>詳細は、7.5 節を参照。</p>
routed(8)	<p>TruCluster Server バージョン 5.1B より前のリリースでは、クラスタのルーティング・デーモンとして gated を実行する必要がある。routed、ogated、または静的ルーティングのセットアップが使用できず、また、ルーティング・デーモンもすべて使用できない。バージョン 5.1B からは、gated、routed、または静的ルーティングが使用可能になる。ogated は使用不可。gated が省略時の設定である。ルーティング・オプションについては 3.14 節を参照。</p> <p>クラスタの最初のメンバを作成すると、clu_create が gated を構成する。クラスタに新規メンバを追加すると、clu_add_member が gated の構成を新規メンバに反映させる。</p> <p>ルータについての詳しい説明は、6.3 節を参照。</p>

表 1-4 に、プリント管理に関する相違点を示します。

表 1-4: プリント管理に関する相違点

コマンド	相違点
lprsetup(8) printconfig(8)	<p>プリント・サーバにするクラスタ・メンバを指定するには、クラスタ固有のプリンタ属性 (on) を使用する。プリント構成ユーティリティ (lprsetup および printconfig) を使用すれば、on 属性を簡単に設定できる。</p> <p>/etc/printcap ファイルはクラスタ内のすべてのメンバによって共用される。</p> <p>詳細は、7.3 節を参照。</p>
Advanced Printing Software	<p>クラスタに Advanced Printing Software をインストールして使用方法についての詳細は、Tru64 UNIX <i>Advanced Printing Software</i> 『システム管理/操作ガイド』を参照。</p>

表 1-5 に、セキュリティ管理に関する相違点を示します。クラスタで使用するエンハンスド・セキュリティ機能については、Tru64 UNIX 『セキュリティ管理ガイド』を参照してください。

表 1-5: セキュリティ管理に関する相違点

コマンド	相違点
auditd(8) auditconfig(8) audit_tool(8)	<p>クラスタはシングル・セキュリティ・ドメインである。クラスタの root 特権を持つユーザは、クラスタ別名かクラスタ・メンバの1つに root としてログインできる。同様に、アクセス制御リスト (ACL) およびユーザ権限/特権もクラスタ全体に対して効力がある。</p> <p>監査ログ・ファイルを除き、セキュリティ関連のファイル、ディレクトリ、およびデータベースはクラスタ全体で共用される。監査ログ・ファイルは各メンバに固有である。つまり、監査デーモン (auditd) は各メンバ上で動作し、各メンバはそれぞれ固有の監査ログ・ファイルを持っている。そのため、クラスタの1つのメンバに障害が発生しても、他のメンバに対する監査は中断されずに続行される。</p> <p>クラスタ全体に関する監査レポートを生成するには、監査ログ名の CDSL を監査リダクション・ツール (audit_tool) で指定する。特定のメンバに関する監査レポートを生成するには、そのメンバに該当するログの名前を指定する (複数指定可能)。</p> <p>エンハンスド・セキュリティ機能が必要な場合は、クラスタの作成前にエンハンスド・セキュリティ機能を構成することを強く推奨する。クラスタの作成後にエンハンスド・セキュリティ機能を構成するには、クラスタ全体をシャットダウンし、リブートする必要がある。</p>

表 1-5: セキュリティ管理に関する相違点 (続き)

コマンド	相違点
rlogin(1) rsh(1) rcp(1)	<p>クラスタからの <code>rlogin</code>, <code>rsh</code>, または <code>rcp</code> 要求では, 送信元アドレスとして省略時のクラスタ別名を使用する。したがって, クラスタ外のホストがクラスタ内のアカウントからのリモート・アクセスを常に許可する場合は, そのホストの <code>.rhosts</code> ファイルにクラスタ別名を登録する必要がある (登録形式は, <code>/etc/hosts</code> ファイルでの登録形式か, NIS またはドメイン・ネーム・システム (DNS) による解決可能な形式でよい)。</p> <p>この要件は, クラスタのメンバ間で機能する <code>rlogin</code>, <code>rsh</code>, または <code>rcp</code> にも適用される。詳細は, 5.3 節を参照。</p>

表 1-6 に, システム構成および管理用のコマンドおよびユーティリティの動作の違いを示します。

表 1-6: 一般のシステム管理に関する相違点

コマンド	相違点
DMS (Dataless Management Services)	DMS は TruCluster Server 環境ではサポートされていない。クラスタは, DMS のクライアントにもサーバにもなれない。
イベント・マネージャ (EVM) およびイベント管理関連	<p>イベントは <code>cluster_event</code> 属性を持っている。この属性に <code>true</code> が設定されたイベントは, クラスタのすべてのメンバにポストされる。<code>false</code> が設定されたイベントは, そのイベントが生成されたメンバのみにポストされる。</p> <p>クラスタ・イベントの一覧については, 付録 A を参照。</p>

表 1-6: 一般のシステム管理に関する相違点 (続き)

コマンド	相違点
halt(8) reboot(8) init(8) shutdown(8)	<p>クラスタ全体に影響する <code>halt</code> または <code>reboot</code> はない。</p> <p>コマンド <code>halt</code> および <code>reboot</code> は実行元のメンバのみに影響する。別のメンバへのファイル・システムの再配置が自動的に行われるので、コマンド <code>halt</code> , <code>reboot</code> , および <code>init</code> は、クラスタにマウントされたファイル・システムをアンマウントしないように修正された。</p> <p>クラスタは <code>shutdown -c</code> を使って停止できる。</p> <p><code>clu_quorum</code> , <code>clu_add_member</code> , <code>clu_delete_member</code> のうちのいずれかのコマンドの実行中に、<code>shutdown -c time</code> コマンドを実行すると、エラーが発生する。</p> <p>クラスタをシャットダウンして停止することはできるが、クラスタ全体をリブートすることはできない (つまり <code>shutdown -r</code> は無効)。</p> <p>クラスタの特定のメンバをシャットダウンするには、そのメンバから <code>shutdown</code> コマンドを実行する。</p> <p>詳細は、<code>shutdown(8)</code> を参照。</p>
hwmgr(8)	<p>クラスタでは、<code>-member</code> オプションに特定のメンバをホスト名で指定して、<code>hwmgr</code> コマンドを実行できる。</p> <p><code>-cluster</code> オプションを使用して、このコマンドをクラスタ全体に対して実行できる。</p> <p><code>-member</code> も <code>-cluster</code> も使用しない場合、<code>hwmgr</code> コマンドの対象は、実行元のシステムになる。</p>



表 1-6: 一般のシステム管理に関する相違点 (続き)

コマンド	相違点
プロセス制御関連 ps(1)	クラスタ全体に一意的なプロセス識別子 (PID) を割り当てるには、プロセス ID の範囲をクラスタの各メンバに割り当てる。ps コマンドは、実行元のメンバ上で動作するプロセスのみに関してレポートを生成する。
kill(1)	ゼロ (0) より大きいパラメータを指定すると、kill コマンドはそのパラメータ値と同じ PID を持つプロセスにシグナルを送る。このとき、そのプロセスがどのメンバで動作しているかは関係ない。指定されたパラメータが -1 より小さいと、プロセス・グループ ID がそのパラメータの絶対値と同じすべてのプロセス (クラスタ単位) にシグナルを送る。 クラスタ・メンバ上の init の PID が 1 でない場合でも、kill 1 はスタンドアロン・システム上で実行されたように動作して、アイドル状態のカーネルと /sbin/init を除くそのクラスタ・メンバ上のすべてのプロセスにシグナルを送る。
rcmgr(8)	/etc/rc.config* ファイルの階層化により、ローカル・エリア・ネットワーク (LAN) 内およびクラスタ内のすべてのシステムに対して構成変数を一貫して定義できる。 詳細は、5.1 節を参照。

表 1-6: 一般のシステム管理に関する相違点 (続き)

コマンド	相違点
sysman_clone(8) sysman -clone	<p>構成のクローン作成と複製はクラスタ非対応。</p> <p>クラスタ内で <code>sysman -clone</code> コマンドを使用しようとする、エラーが発生し、次のメッセージが返される。<code>Error: Cloning in a cluster environment is not supported.</code></p>
システム課金サービスおよび関連コマンド fuser(8) mailstats(8) ps(1) uptime(1) vmstat(1) w(1) who(1)	<p>これらのコマンドはクラスタ対応ではない。これらのコマンドの1つを実行すると、実行元のクラスタ・メンバに関する情報のみが返され、クラスタ全体に関する情報は返されない。</p> <p>5.15 節を参照。</p>

表 1-7 に、TruCluster Server でサポートされていない機能を示します。

表 1-7: サポートされていない機能

機能	コメント
アーカイブ関連 bttape(8)	bttape ユーティリティはクラスタ非対応。 ファイルのバックアップおよび復元についての 詳細は、9.8 節を参照。
LSM 関連 volrootmir(8) volunroot(8)	volrootmir コマンドと volunroot コマン ドはクラスタ非対応。 クラスタ内での LSM についての詳細は、 第 10 章を参照。
mount(8)	NFS ループバック・マウントはサポートさ れていない。詳細については、7.6.2.4 項を 参照。  mountd コマンドを介して実行される別のコ マンド (umount や export など) は、外部ク ライアントから発行され、省略時のクラスタ 別名または /etc/exports.aliasess の中で 指定されている別名を使用しないときは、 「Program unavailable」というエラー が返される。
Prestoserve presto(8) dxpresto(8X) prestosetup(8) prestctl_svc(8)	Prestoserve はクラスタ非対応。

表 1-7: サポートされていない機能 (続き)

機能	コメント
routed(8)	<p>TruCluster Server バージョン 5.1B より前のリリースでは、クラスタのルーティング・デーモンとして <code>gated</code> を実行する必要がある。<code>routed</code>、<code>ogated</code>、または静的ルーティングのセットアップが使用できず、ルーティング・デーモンもすべて使用できない。バージョン 5.1B からは、<code>gated</code>、<code>routed</code>、または静的ルーティングが使用可能になる。<code>ogated</code> は使用不可。<code>gated</code> が省略時の設定である。ルーティング・オプションについては 3.14 節を参照。</p> <p>最初のクラスタ・メンバを作成するときに、<code>clu_create</code> により <code>gated</code> が構成される。新しいクラスタ・メンバを追加するときに、<code>clu_add_member</code> によりその構成が新しいメンバに伝えられる。</p> <p>ルータについての詳細は、6.3 節を参照。</p>
DMS (Dataless Management Services)	<p>DMS は TruCluster Server 環境ではサポートされていない。クラスタは DMS クライアントにもサーバにもなれない。</p>
UNIX ファイル・システム (UFS)	<p>クラスタ・メンバは UFS ファイル・システムを読み取り/書き込みでマウントできる。ファイル・システムはそのメンバだけがアクセスできる。リモートからはアクセスできず、フェイルオーバーはできない。</p>
sysman_clone(8) sysman -clone	<p>構成のクローン作成と複製は非クラスタ対応。</p> <p>クラスタ内で <code>sysman -clone</code> コマンドを使用しようとする、エラーが発生して、次のメッセージが返される。Error: Cloning in a cluster environment is not supported.</p>

---

## クラスタ管理ツール

この章では、TruCluster Server システムの管理に使用できるツールについて記載します。以下の管理ツールとオプションについて説明しています。

- 単一システムとクラスタの管理の両方で利用可能な管理ツールの概要 (2.1 節)
- クラスタ固有の構成ツールと関連ユーザ・インタフェースの概要 (2.2 節)
- SysMan の概要 (2.3 節)
- クラスタ内での SysMan Menu の使用 (2.4 節)
- クラスタ内での SysMan Station の使用 (2.5 節)
- クラスタ内での SysMan Java アプレットの使用 (2.6 節)
- クラスタ内での SysMan Java PC アプリケーションの使用 (2.7 節)
- クラスタ内での SysMan コマンド行インタフェースの使用 (2.8 節)
- クラスタ内での HP Insight Manager の使用 (2.9 節)
- Tru64 UNIX 構成レポートの使用 (2.10 節)

### 2.1 利用可能な管理ツールとインタフェースの概要

Tru64 UNIX には、単一システムとクラスタの管理用に数多くの管理ツールが用意されています。クラスタは、可能な限り単一システムとして管理されます。

多くのシステムが異機種混在環境で使用されていますが、このような環境では、システム管理者が、PC から、Tru64 UNIX ワークステーションから、文字セル端末から、あるいはダイヤルアップ回線経由のノート PC から TruCluster Server システムを管理すると思われます。

こうした事実から、Tru64 UNIX と TruCluster Server には、管理タスクを行うための Web ベースのグラフィカル・インタフェースとコマンド行インタフェースが用意されています。特に、SysMan には、システムとクラスタの

管理のためのコマンド行，文字セル端末，Java，X Window，および Web ベースの Java アプレットの各インタフェースが用意されています。

SysMan は単一のアプリケーションまたはインタフェースではありません。むしろ，SysMan は，Tru64 UNIX と TruCluster Server システムの管理用のアプリケーション群です。SysMan には，SysMan Menu，SysMan Station，および SysMan コマンド行インタフェースという 3 つの主要な構成要素があります。それぞれの構成要素について，この章で説明します。

ニーズに合ったツールとユーザ・インタフェースを選択できます。おそらく従来からの Tru64 UNIX コマンド行の多彩な機能や柔軟性が最も快適と思われます。あるいは，PC からのクラスタ管理に重点を置く場合は，SysMan に対する Java スタンドアロン・グラフィカル・インタフェースを利用すれば，Windows が稼働している PC から管理タスクを実行できます。

使用できる多数のクラスタ管理ツールやインタフェースがあるので，この章では，初めに各種オプションについて説明します。各オプションの特徴や機能は，以降の各項で簡単に説明していますが，詳細は Tru64 UNIX 『システム管理ガイド』を参照してください。

一部のクラスタ操作では，グラフィカル・インタフェースを利用できないので，コマンド行インタフェースを利用する必要があります。このような操作とコマンドについては，2.2 節で説明します。

2.1.1 クラスタ・ツールのクイック・スタート

すでにクラスタ管理ツールを熟知していて，ツールを使用し始めたい場合は，表 2-1 を参照してください。この表では，要約情報だけを示しています。詳細については，この章の以降の部分で説明します。

表 2-1: クラスタ・ツールのクイック・スタート

ツール	ユーザ・インタフェース	起動方法
SysMan Menu	X Window	# /usr/sbin/sysman -menu [-display <i>display</i> ]
	文字セル	# /usr/sbin/sysman -menu
	Java アプレット	<a href="http://cluster_member_name:2301/SysMan_Home_Page">http://cluster_member_name:2301/SysMan_Home_Page</a>
	PC アプリケーション	<a href="http://cluster_member_name:2301/SysMan_Home_Page">http://cluster_member_name:2301/SysMan_Home_Page</a>

2-2 クラスタ管理ツール

表 2-1: クラスタ・ツールのクイック・スタート (続き)

ツール	ユーザ・インタフェース	起動方法
SysMan Station	X Window	# /usr/sbin/sysman -station [hostname] [-display display]
	Java アプレット	http://cluster_member_name:2301/SysMan_Home_Page
	PC アプリケーション	http://cluster_member_name:2301/SysMan_Home_Page
SysMan -CLI	コマンド行	# /usr/sbin/sysman -cli
HP Insight Manager	Web インタフェース	http://cluster_member_name:2301/
HP Insight Manager XE	Web インタフェース	http://xe_server_name:280/

## 2.1.2 HP Insight Manager の統合

SysMan を介した HP Insight Manager と Web ベースのシステム管理は、密接に結合されています。HP Insight Management のエージェントとサブエージェントにより、管理されるサブシステムすべてのデバイスとシステムの情報が得られます。HP Insight Manager の Web ベースの SNMP (Simple Network Management Protocol) エージェントを利用すれば、システムの状態に関する情報の収集と表示を行うことができます。SNMP エージェントでは、読み取り専用の情報が提供されるので、システムを管理することはできません。

HP Insight Manager には、SysMan 用の Web ベースの管理 (WBEM) フレームワークも用意されています。HP Insight Manager は、次の URL から入手して各 Tru64 UNIX システムで利用できます。

http://cluster\_member\_name:2301/

このサイトから、Web ブラウザで直接 SysMan Menu または SysMan Station を実行するか、PC クライアント・キットをダウンロードして、それぞれのアプリケーションをローカルにインストールすることができます。

## 2.1.3 HP Insight Manager XE の統合

HP Insight Manager XE は Common Cluster 管理情報ベース (MIB) を使って TruCluster Server を探し出し監視します。MIB は /usr/sbin/clu\_mibs

SNMP サブエージェントがサポートします。このサブエージェントは、クラスタ・ソフトウェアに含まれており、自動的に起動します。

clu\_mibs は TruCluster Server システム用の Extensible SNMP サブエージェント・デーモンであり、クラスタ MIB をサポートしています。このデーモンは現時点で Common Cluster MIB (/usr/share/sysman/mibs/svrClu.mib) と TruCluster Server MIB (/usr/share/sysman/mibs/truClu.mib) をサポートしています。

HP Insight Manager XE は、Web インタフェースを通してクラスタ状態の情報を収集し表示します。SNMP エージェントからは情報を読み取ることしかできず、このエージェントを使ってシステムを管理することはできません。管理作業を行う場合は、他のツールを使ってください。

2.1.4 利用可能な管理ツールとインタフェース

この項では、実行できるツールとプラットフォームおよびインタフェースとの関係について説明します。利用可能な管理ツールとインタフェースを、表 2-2 に示しています。

表 2-2: 利用可能な管理ツールとインタフェース

ツール	X Window	文字セル	Tru64 UNIX コマ ンド行	PC アプリ ケーション	Java アプ レット <sup>a</sup>	任意の プラット フォーム上 の Web ブラウザ
sysman -menu	可	可	不可	可	可	不可
sysman -station	可	不可	不可	可	可	不可
sysman -cli	不可	不可	可	不可	不可	不可
HPInsight Manager	不可	不可	不可	不可	不可	可
HP Insight Manager XE	不可	不可	不可	不可	不可	可

<sup>a</sup>ブラウザの要件は、2.6 節を参照。

どのユーザ環境を利用しても、操作のインタフェースは一貫しています。たとえば、SysMan Menu は、CDE (Common Desktop Environmen) を介した X Window アプリケーションとして文字セル端末インタフェース経由で起動



するか、Java インタフェースを介して起動するかに関係なく類似しています。ただし、インタフェース間でナビゲーション方法に違いがあります。たとえば、SysMan Menu の文字セル・インタフェースには、アイコンのようなグラフィカルな要素はありません。それに対して、X Window インタフェースは、CDE のようなウィンドウ環境で動作するようになっているので、クリック可能なボタン、ドロップダウン・リストなどを備えています。

HP Insight Manager の Web ベースの SNMP エージェントは、システムの状態に関する情報の収集と表示を行います。管理タスクを行うには、他のツールを使用してください。

## 2.2 クラスタ構成ツールと関連ユーザ・インタフェース

すべての TruCluster 管理ツールに、SysMan インタフェースが備わっているわけではありません。表 2-3 は、クラスタ固有のタスクの管理ツールを示し、どのツールが SysMan Menu から利用できないかを示しています。この表では、NA は利用不可を表します。

表 2-3: クラスタ管理ツール

コマンド	SysMan Menu での利用	機能
caa_balance(8) caa_profile(8) caa_register(8) caa_relocate(8) caa_report(8) caa_start(8) caa_stat(1) caa_stop(8) caa_unregister(8)	sysman caa	CAA (Cluster Application Availability) サブシステムで高可用性アプリケーションを管理する。
cfsmgr(8)	sysman cfsmgr	クラスタ・ファイル・システムを管理する。
cluamgr(8)  clua_active(8)	sysman clu_aliases	クラスタ別名を作成して管理する。  クラスタ別名がアクティブで到達可能かどうかを調べる。
clu_add_member(8)	NA	クラスタにメンバを追加する。

表 2-3: クラスタ管理ツール (続き)

コマンド	SysMan Menu での利用	機能
clu_create(8)	NA	Tru64 UNIX システム上に最初のクラスタ・メンバを作成する。
clu_check_config(8)	NA	TruCluster Server が正しくインストールされて、クラスタが正しく構成されているかどうかを検査する。
clu_delete_member(8)	NA	クラスタからメンバを削除する。
clu_get_info(8)	sysman hw_cluhierarchy (ほぼ同等)	クラスタとそのメンバに関する情報を取得する。
clu_quorum(8)	NA	クォーラム・ディスクの構成または削除、あるいはクォーラム・ディスクのポート、メンバ・ポート、または期待ポートの調整を行う。
drdmgr(8)	sysman drdmgr	分散デバイスを管理する。
imcs(1)	NA	Memory Channel アプリケーション・プログラミング・インタフェース (API) ライブラリ libimc の状態をレポートする。
imc_init(1)	NA	現在のホスト上で Memory Channel API ライブラリ libimc を初期化して構成する。
mkcdsl(8)	NA	CDSL の作成または検証を行う。

## 2.3 SysManの構成要素の概要

この節では、SysMan の管理オプションを紹介します。SysMan については、sysman\_intro(8) および sysman(8) を参照してください。

SysMan には、クラスタ・ファイル・システム、ストレージ、およびクラスタ別名の管理をはじめとする一般的なシステム管理タスク用の使いやすいインタフェースが用意されています。SysMan に対するインタフェース・オプションには、以下のような利点があります。

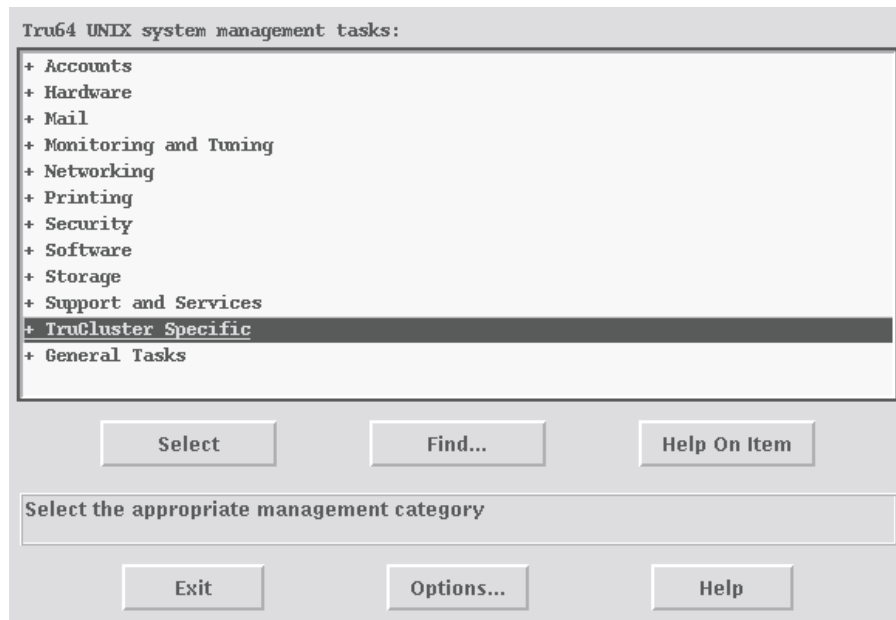
- Tru64 UNIX および Microsoft Windows の操作環境からアクセスする使い慣れたインタフェース。
- 管理のしやすさ。コマンド行構文を理解したり，手動で構成ファイルを編集したりする必要はありません。

SysMan には，SysMan Menu，SysMan Station，および SysMan コマンド行インタフェースという 3 つの主要な構成要素があります。以降の各項で，それぞれの構成要素を説明します。

### 2.3.1 SysMan Menu の紹介

図 2-1 に示すように，SysMan Menu により，単一システムやクラスタの利用可能な管理ユーティリティの大部分がメニュー・システムに組み込まれます。

図 2-1: SysMan Menu の階層

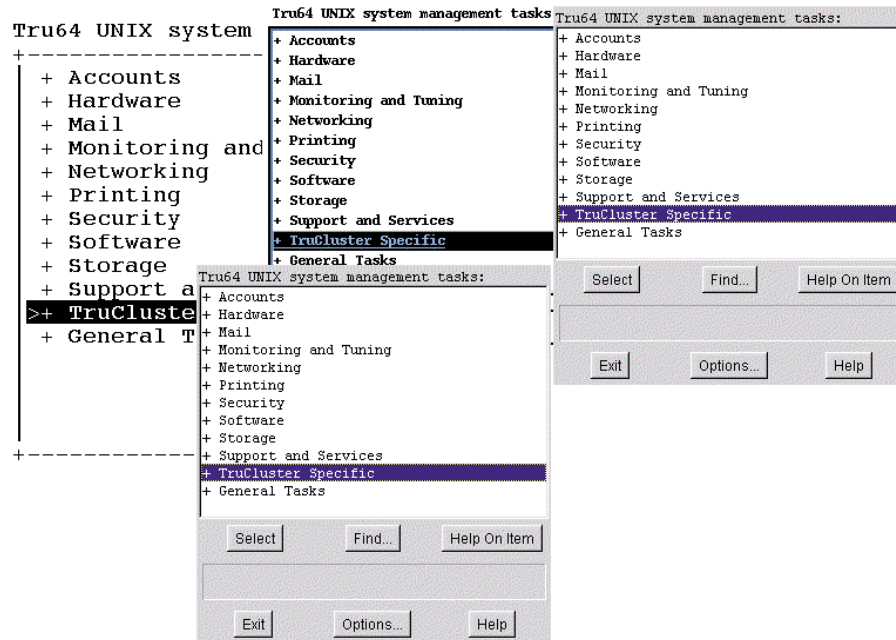


ZK-1702U-AI

SysMan Menu では，システム管理タスクのメニューは，管理カテゴリのプランチと実際のタスクのリーフという編成のツリー階層で提供されます。リーフを選択するとタスクが起動されて，タスクを実行するためのダイアログ・ボックスが表示されます。

SysMan Menu のユーザ・インタフェースは、SysMan Menu の起動方法にかかわらず機能的に同じです。たとえば、図 2-2 は、文字セル、X Window、Java、および Java アプレットのユーザ・インタフェースによる各ウィンドウを示しています。

図 2-2: SysMan Menu のインタフェース



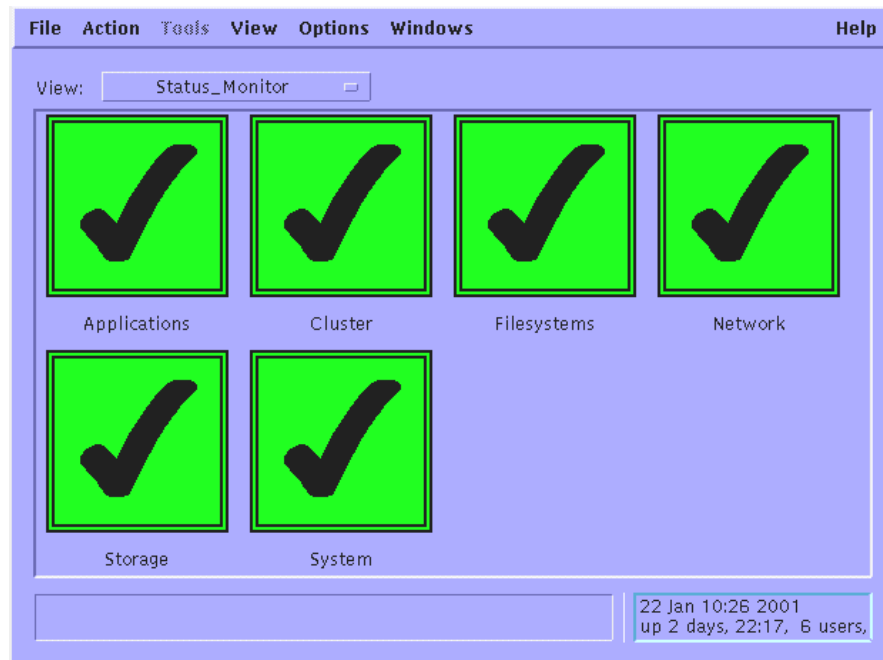
ZK-1695U-AI

### 2.3.2 SysMan Station の紹介

SysMan Station では、システム（またはクラスタ）がグラフィカルに表現されるため、ディスクなどの個々のシステム構成要素のレベルまでシステムの状態を監視できるようになります。また、ファイル・システム、AdvFS (Advanced File System) ドメインなどの論理グループの表示や監視を行ったり、独自のビューを作成することもできます。システム構成要素を表示すると、その構成要素のプロパティまたは構成要素で管理タスクを実行できるようにする起動ユーティリティに関する詳細な情報を得ることができます。SysMan Menu と異なり、SysMan Station は、グラフィック機能が必要で、文字セル・ユーザ環境からは実行できません。

図 2-3 は、SysMan Station のグラフィカル・インタフェースの例を示しています。

図 2-3: SysMan Station のグラフィカル・インタフェース



ZK-1703U-AI

SysMan Menu と同様に、SysMan Station のユーザ・インタフェースは、SysMan Station の起動方法にかかわらず機能的に同じです。

### 2.3.3 SysMan コマンド行の紹介

`sysman -cli` コマンドは、SysMan の機能に対する汎用コマンド行インタフェースを備えています。`sysman -cli` コマンドを使用すれば、SysMan のデータの表示や変更を行うことができます。また、このコマンドを使用すれば、`sysman_cli(8)` で説明しているように、辞書型の情報 (SysMan データのデータ説明、キー情報、タイプ情報など) を表示することもできます。既知の構成要素すべてを SysMan データ階層に表示するには、`sysman -cli -list components` コマンドを使用してください。

## 2.4 クラスタでの SysMan Menu の使用

この節では、クラスタで SysMan Menu を使用する方法について説明します。初めに、フォーカスと SysMan Menu に対するフォーカスの影響を解説します。

### 2.4.1 フォーカスの取得

特定の管理操作の効果が及ぶ範囲のことをその操作のフォーカスといいます。TruCluster 環境では、管理操作のフォーカスは以下の 4 とおりが考えられます。

- クラスタ単位 — 操作は、クラスタ全体に影響します。
- メンバ固有 — 操作は、指定したメンバのみに影響します。この操作には、フォーカスが必要です。
- 両方 — 操作は、クラスタ単位またはメンバ固有になる可能性があります。この操作には、フォーカスが必要です。
- なし — 操作は、フォーカスを必要とせずに、常に現在使用中のシステムが対象となります。

SysMan Menu では、管理タスクごとに選択する最適なフォーカスがわかっています。タスクでクラスタ単位の操作とメンバ固有の操作の両方がサポートされている場合は、SysMan Menu で、操作対象のクラスタ名または特定のメンバを選択できます。つまり、クラスタ名とクラスタ・メンバを選択できる場合は、操作は両方になりますが、メンバ名しか選択できない場合は、操作はメンバ固有になります。

特定の操作に対するフォーカス情報は、SysMan Menu のタイトル・バーに表示されます。たとえば、クラスタのローカル・ユーザをクラスタ単位の操作で管理するときは、タイトル・バーに以下のように表示される可能性があります (この例にある provolone はクラスタ・メンバで、deli はクラスタ別名です)。

```
Manage Local Users on provolone.zk3.dec.com managing deli
```

### 2.4.2 コマンド行でのフォーカスの指定

操作でフォーカスを指定する場合は、SysMan Menu の `-focus` オプションを使用すれば、コマンド行からフォーカスを指定できます。

コマンド行でのフォーカスの指定が shutdown コマンドに及ぼす影響を考慮してください。shutdown コマンドは、クラスタ単位またはメンバ固有で実行される場合があります。次のコマンドでクラスタ・メンバから SysMan Menu を起動した場合は、クラスタ名が shutdown オプションの最初のフォーカスになります。

```
# sysman -menu
```

ただし、次のコマンドでクラスタ・メンバから SysMan Menu を起動した場合は、provolone クラスタ・メンバが shutdown オプションの最初のフォーカスになります。

```
# sysman -menu -focus provolone
```

SysMan Menu のセッション中に新しい管理タスクを開始すると、このダイアログ・ボックスには常に、前回のタスクでのフォーカスの指定先が強調表示されます。このため、1 つのクラスタ・メンバ上で数多くの管理機能を実行する場合は、そのメンバを 1 回だけ選択する必要があります。

### 2.4.3 SysMan Menu の起動

表 2-4 に示すように、各種インタフェースから SysMan Menu を起動できます。

表 2-4: SysMan Menu の起動

ユーザ・インタフェース	起動方法
文字セル端末	<p>クラスタ・メンバ上で端末セッションを起動するか、端末ウィンドウを開いて、次のコマンドを入力する。</p> <pre># /usr/sbin/sysman -menu</pre> <p>DISPLAY 環境変数の設定、SysMan Menu コマンド行での <code>-display</code> 修飾子による直接指定、あるいはその他のメカニズムを介して X Window 表示がこの端末ウィンドウに関連付けられている場合は、SysMan Menu に対する X Window インタフェースが代わりに起動される。このような場合には、文字セル・インタフェースを強制的に使用するために、次のコマンドを使用する。</p> <pre># /usr/sbin/sysman -menu -ui cui</pre>
CDE (または他の X Window 表示)	<p>SysMan Menu は、X Window のウィンドウ環境で利用できる。SysMan Menu を起動するには、次のコマンドを入力する。</p> <pre># /usr/sbin/sysman -menu [-display displayname]</pre> <p>CDE インタフェースを使用している場合、SysMan を起動するには、root ユーザのフロント・パネルの [SysMan] サブメニュー・アイコンをクリックして [SysMan Menu] を選択する。サブメニュー・アイコンではなく [SysMan] アイコン自体をクリックした場合は、SysMan Station が直接起動される。</p> <p>また、フロント・パネルの [アプリケーション・マネージャ] アイコンをクリックしてから、[システム管理] グループの [SysMan Menu] アイコンをクリックすれば、CDE から SysMan Menu を起動することもできる。</p>



表 2-4: SysMan Menu の起動 (続き)

ユーザ・インタフェース	起動方法
コマンド行	SysMan Menu は、コマンド行からは利用できない。ただし、SysMan コマンド行インタフェース <code>sysman -cli</code> を利用すれば、コマンド行から SysMan ルーチンを実行したり、SysMan インタフェースへの入力をカスタマイズするプログラムを作成することができる。オプションとフラグの詳細は、 <code>sysman_cli(8)</code> を参照。さらに詳しくは、2.8 節を参照。
Web ベースの Java アプレット	2.6.1 項を参照。
スタンドアロン Java アプリケーション	2.7.1 項を参照。

## 2.5 クラスタでの SysMan Station の使用

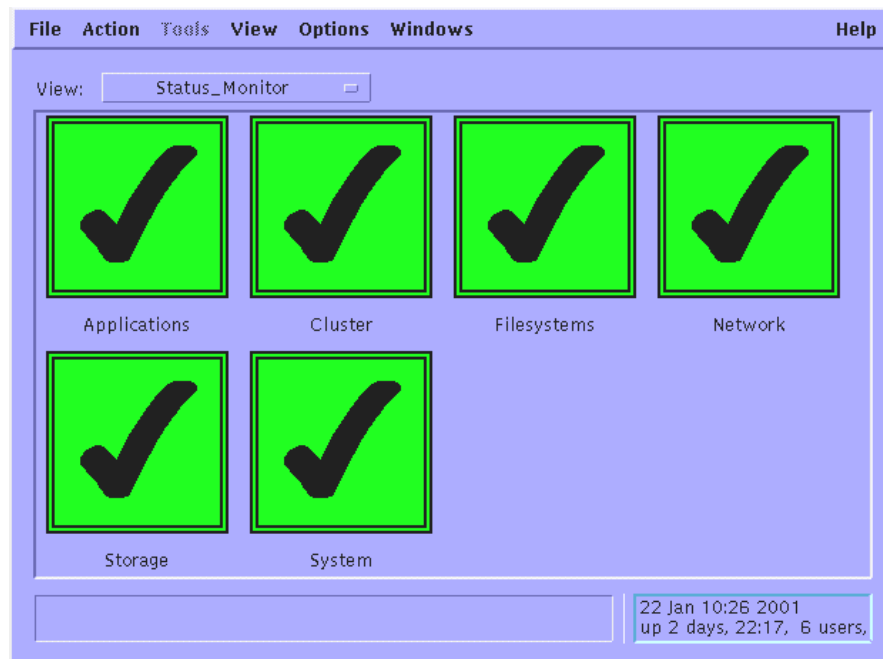
SysMan Station は、デーモン `smsd(8)` と SysMan Station のグラフィカル・ユーザ・インタフェースからなるクライアント/サーバ・アプリケーションです。SysMan Station は、単一システムまたはクラスタの監視と管理を行います。SysMan Menu を起動するか、SysMan Station からアプリケーションを直接起動することもできます。SysMan Station を使用して、以下のタスクを実行できます。

- システムまたはクラスタの状態の監視
- システムまたはクラスタに関する詳細情報の表示
- 管理タスク用の 1 つの場所の提供
- イベントの表示と問題の原因となるイベントの記録

SysMan Station を起動しておくことが便利であるとわかれば、SysMan Station をデスクトップで実行したままにしておいてください。特に、Tru64 UNIX システムの管理の経験がない場合は、SysMan Station を利用すれば、先に Tru64 UNIX コマンドの構文を習得しなくてもクラスタを管理できます。

クラスタ・メンバから SysMan Station を起動すると、モニタ・ウィンドウが表示されます (図 2-4)。

図 2-4: SysMan Station の最初のクラスタ・ビュー



ZK-1703U-AI

モニタ・ウィンドウには、以下のサブシステムの状態が表示されます。

- アプリケーション
- クラスタ
- ファイル・システム
- ネットワーク
- ストレージ
- システム

状態を示す色とパターンで、以下のような障害またはトラブル状況が示されます。

- 正常な状態は緑色で、チェック・マークが表示されます。
- トラブル状態は黄色で、感嘆符 (!) が表示されます。
- 障害発生状態は赤色で、× が表示されます。

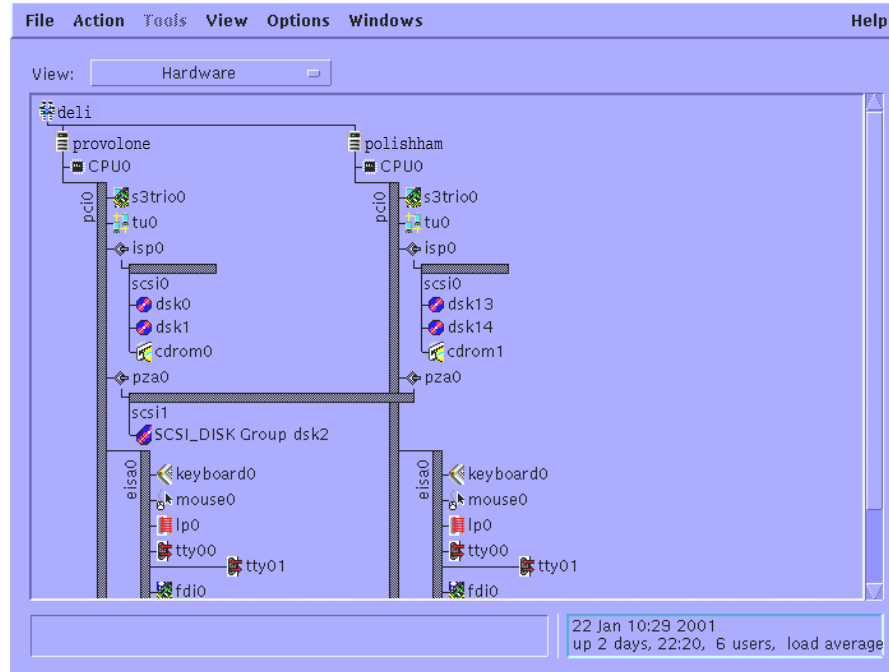
状態インジケータ，またはその下のラベルをクリックすれば，指定したサブシステムに対して通知されたイベントを表示できます。`cluster_event` 属性に `true` が設定されたイベントを除き，イベントは，イベントを生成したメンバによって識別されます。クラスタ・イベントの一覧については，付録 A を参照してください。

ドロップダウン・リストには，次のような利用可能なビューが表示されます。

- AdvFS\_FileSystems
- CAA\_Applications\_(active)
- CAA\_Applications\_(all)
- Hardware
- Mounted\_FileSystems
- Physical\_FileSystems

このいずれかのビューをクリックすれば，そのビューを表示する新しいウィンドウを開くことができます。たとえば，Hardware ビューをクリックすると，クラスタ・ハードウェアのビューが表示されます。このビューの例を図 2-5 に示しています。

図 2-5: SysMan Station のクラスタ・ハードウェア・ビューのサンプル



ZK-1700U-AI

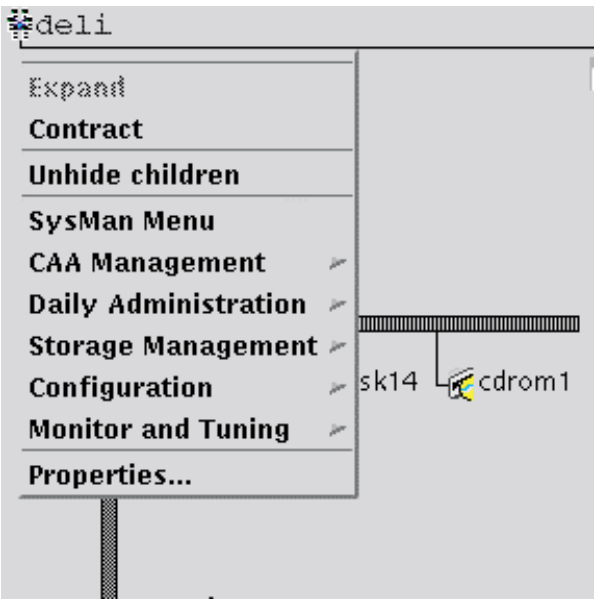
ビュー内のオブジェクトには、そのタイプに基づくアクションが設定されます。つまり、クラスタやディスク・オブジェクトのようなオブジェクトには、管理アクションが関連付けられており、アクションはオブジェクトのタイプによって変わります。たとえば、クラスタ・オブジェクトでは、以下の管理アクションを実行できます。

- SysMan Menu
- Daily Administration
- Storage Management
- Configuration
- Monitor and Tuning
- CAA Management
- Properties

一部の選択項目には、管理タスクが関連付けられていない場合があります。  
たとえば、グラフィック・カードの場合は、そのプロパティと関連イベント  
を表示できますが、そのカードを管理することはできません。

特定のオブジェクトに対して有効なアクションを調べるには、図 2-6 に示す  
ように、オブジェクトをポイントしてマウスの右ボタンをクリックして押  
したままにします。

図 2-6: SysMan Station での有効なアクションの表示



ZK-1704U-AI

### 2.5.1 SysMan Station の起動

表 2-5 に示すように、各種インタフェースから SysMan Station を起動  
できます。

表 2-5: SysMan Station の起動

ユーザ・インタフェース	起動方法
文字セル端末	SysMan Station は、ローカルまたはリモートの文字セル端末では利用できない。

表 2-5: SysMan Station の起動 (続き)

ユーザ・インタフェース	起動方法
CDE (または他の X Window 表示)	<p>SysMan Station は、X Window のウィンドウ環境で利用できる。SysMan Station を起動するには、次のコマンドを入力する。</p> <pre># /usr/sbin/sysman -station [hostname] [-display display]</pre> <p>CDE インタフェースを使用している場合、SysMan Station を起動するには、root ユーザのフロント・パネルの [SysMan] アイコンをクリックする。フロント・パネルの [アプリケーション・マネージャ] アイコンをクリックしてから、[システム管理] グループの [SysMan Station] アイコンをクリックすることもできる。</p>
コマンド行	<p>SysMan Station は、コマンド行からは利用できない。ただし、SysMan コマンド行インタフェース <code>sysman -cli</code> を利用すれば、コマンド行から SysMan ルーチンを実行したり、SysMan インタフェースへの入力をカスタマイズするプログラムを作成できる。オプションとフラグの詳細は、<code>sysman_cli(8)</code> を参照。</p> <p>さらに詳しくは、2.8 節を参照。</p>
Web ベースの Java アプレット	2.6.1 項を参照。
スタンドアロン Java アプリケーション	2.7.1 項を参照。

## 2.6 クラスタでの SysMan Java アプレットの使用

### 注意

Web ページのサービスを提供している Tru64 UNIX システムだけを管理できます。クラスタでは、これは Web ページのサービスを提供しているクラスタ・メンバだけを管理できることを表します。したがって、クラスタを管理するには、2.7 節で説明している Java PC アプリケーションを使用してください。

HP Insight Manager には、SysMan 対応の Web ベースの管理 (WBEM) フレームワークが 2 つの Java アプレット形式で用意されています。

アプレットには、Web ブラウザ内部で実行するアプレット本体、および Tru64 UNIX システム上で実行する SysMan Station デーモン `/usr/sbin/smsd` の 2 つの構成要素があります。

ブラウザを使用して正しい URL を開き、いずれかのアプレットを起動します。すると、アプレットは、ポート 2301 で部分的に HP Insight Manager http サーバを介して、Tru64 UNIX システムとやりとりします。

---

#### ブラウザの要件

---

サポートするブラウザについての情報は次の Web ページを参照してください。

`http://cluster_member_name:2301/SysMan_Home_Page`

---

Tru64 UNIX システムでは、HP Insight Manager エージェント (デーモン) は、オペレーティング・システムをインストールすると省略時の設定で構成され、システムがブートすると自動的に起動されます。

HP Insight Manager Web エージェントは、レベル 3 を実行するための状態遷移中に `/sbin/rc3.d/S50insightd` スクリプトによって初期化されます。このスクリプトは、`/usr/sbin/insightd` を実行し、エージェントが正常に起動されると、コンソールにブート時メッセージを出力します。SNMP サブエージェントの `/usr/sbin/os_mibs` と `/usr/sbin/cpq_mibs` も、レベル 3 を実行するための状態遷移中に `/sbin/rc3.d/S49snmpd` スクリプトによって起動されます。システムが正しく構成されているかどうかをテストするには、次のコマンドを入力します。

```
# ps agx | grep insight
# ps agx | grep cpq
# ps agx | grep os_mib
```

あるいは、次のコマンドを入力します。

```
# ps agx | grep -E "insight|cpq|os_mibs"
```

使用する可能性が低いために、省略時には HP Insight Manager Web Agent を使用可能にしたいくない場合は、SysMan Menu または次の `sysman imconfig` コマンドを介して、Web Agent を使用不能にすることができます。

```
# /usr/sbin/sysman imconfig
```

HP Insight Manager Web Agent を使用不能にした場合は、SysMan PC アプリケーションからオンライン・ヘルプを使用できません。

### 2.6.1 SysMan Java アプレットの起動

Web ブラウザから SysMan Java アプレットを直接実行する方法の詳細については、適合する Web ブラウザで次の場所にアクセスしてください。

```
http://cluster_member_name:2301/SysMan_Home_Page
```

ブラウザが適合する場合は、SysMan Station または SysMan Menu へのリンクをクリックすると、ブラウザ内でアプレットが起動されます。アプレットの起動に少し時間がかかるかもしれません。

アプレットが起動されると、クラスタ・メンバへの接続が確立します。root でログインしてください。

Web ブラウザから SysMan Menu や SysMan Station を実行する方法に慣れると、それらを次の URL から直接起動する方がさらに便利であることがわかります。

- SysMan Menu の起動

```
http://cluster_member_name:2301/suit_applet.html
```

- SysMan Station の起動

```
http://cluster_member_name:2301/sms_applet.html
```

## 2.7 クラスタでの SysMan Java PC アプリケーションの使用

SysMan Menu と SysMan Station はどちらも Windows PC 上で動作する Java スタンドアロン・アプリケーションとして利用できます。これらのアプリケーションを他の PC アプリケーションとまったく同様にインストールして実行します。ただし、これらのアプリケーションは Web ブラウザ・ベースではありません。

アプリケーションが Java 実行時環境 (JRE) アプリケーションとして実行する点を除けば、アプリケーションの外観や機能は、文字セル版や X Window 版と同様です。

SysMan Java アプレットと異なり、スタンドアロン Java アプリケーションは、HP Insight Manager デーモンや http サーバに依存していません。



smsd デーモン smsd(8) は、ホストからシステム管理データを収集し、その情報を SysMan Station Java クライアントに渡す役割を果たします。

### 2.7.1 PC 上での SysMan Java アプリケーションの起動

Windows PC 上でのスタンドアロン Java アプリケーションのダウンロードと実行の詳細については、Web ブラウザで次の場所にアクセスしてください。

[http://cluster\\_member\\_name:2301/SysMan\\_Home\\_Page](http://cluster_member_name:2301/SysMan_Home_Page)

「Managing Tru64 UNIX from a PC」の節で、SysMan Station Java スタンドアロン・アプリケーションのダウンロード、インストール、および実行の手順について説明しています。

## 2.8 クラスタでの SysMan コマンド行インタフェースの使用

sysman -cli コマンドには、SysMan データへの汎用コマンド行インタフェースが用意されています。sysman -cli コマンドを使用すれば、SysMan データの表示または変更を行うことができます。また、このコマンドを使用すれば、sysman\_cli(8) で説明しているように、辞書型の情報 (SysMan データのデータ記述、キー情報、タイプ情報など) を表示することもできます。

フォーカスを指定するには、-focus オプションを使用します。フォーカスとは特定の管理タスクの効果の及ぶ範囲のことで、これによりクラスタの範囲が、クラスタ・メンバ全体または特定のクラスタ・メンバと見なされます。

既知の構成要素すべてを SysMan データ階層に表示するには、sysman -cli -list component コマンドを使用します。

sysman -cli コマンドの例を例 2-1 に示しています。このコマンドでは、amember という名前のクラスタ・メンバの clua 構成要素の属性が表示されます。

#### 例 2-1: sysman の出力例

---

```
# sysman -cli -focus amember -list attributes -comp clua
Component: clua
  Group: cluster-aliases
    Attribute(s):
      aliasname
      memberlist
  Group: clua-info
```

### 例 2-1: sysman の出力例 (続き)

---

```
Attribute(s):
  memberid
  aliasname
  membername
  selw
  selp
  rpri
  joined
  virtual
Group: componentid
Attribute(s):
  manufacturer
  product
  version
  serialnumber
  installation
  verify
Group: digitalmanagementmodes
Attribute(s):
  deferredcommit
  cdfgroups
```

---

## 2.9 クラスタでの HP Insight Manager の使用

HP Insight Manager を使用すれば、現在の Web ブラウザを使用して、広範囲な Tru64 UNIX 構成情報を表示できます。Tru64 UNIX システム上の Web ブラウザまたは Windows PC 上の Web ブラウザを使用できます。どちらを使用するかはユーザ次第です。

Tru64 UNIX に実装されるように、HP Insight Manager は、Tru64 UNIX システムのポート 2301 で受信待ちをするプライベート http サーバと Tru64 UNIX SNMP サブエージェントの組み合わせを利用して、クラスタ・メンバの構成情報を表示する Web ベースのインタフェースになっています。つまり、SNMP サブエージェント `/usr/sbin/os_mibs` と `/usr/sbin/cpq_mibs` では、属性の取得はできますが、属性の設定はできません。

HP Insight Manager Web エージェントは、レベル 3 を実行するための状態遷移中に初期化されます。初期化スクリプトは `/sbin/rc3.d/S50insightd` にあります。このスクリプトは、`/usr/sbin/insightd` を実行し、エージェントが正常に起動されると、コンソールにブート時メッセージを出力します。

SNMP サブエージェントの /usr/sbin/os\_mibs と /usr/sbin/cpq\_mibs も、レベル 3 を実行するための状態遷移中に /sbin/rc3.d/S49snmpd スクリプトによって起動されます。

HP Insight Management エージェントは、クラスタ対応ではありませんが、指定したクラスタ・メンバに関する有用なデバイス情報や状態情報を提供します。特に、HP Insight Management を利用すれば、経験の浅いサポート要員でも Tru64 UNIX コマンド行インタフェースを使用せずに、システムやデバイスの情報 (特定のディスク・デバイスの容量、シリアル番号など) を収集できることがわかります。

HP Insight Manager の表示サンプルを 図 2-7 に示しています。

図 2-7: HP Insight Manager の表示

The screenshot displays the HP Insight Manager web interface. On the left is a navigation sidebar with sections: 'COMPAQ INSIGHT MANAGEMENT AGENTS Version 4.21' (with links for Agent Help, Summary, Device Home, and Options), 'Condition Legend' (Unknown, OK), 'CONFIGURATION' (with links for System Info, System Board, Software Version Info), and 'MASS STORAGE' (with links for File System Space Used and Floppy Drives). The main content area is titled 'CONFIGURATION System Information' and shows 'General Information' and 'Description' for the device 'zk3.zk3.dec.com : AlphaServer 800 5/500'. The 'General Information' section lists Product, Operating System, Machine ID, and Expansion Bus. The 'Description' section lists System Name and a detailed description including the revision and date.

General Information	
Product:	AlphaServer 800 5/500
Operating System:	Compaq Tru64 UNIX Rev. 148
Machine ID (System Board):	N/A
Expansion Bus:	ISA/PCI

Description	
System Name:	zk3.zk3.dec.com
Description:	zk3.zk3.dec.com AlphaServer 800 5/500 Compaq Tru64 UNIX (Rev. 148); Wed Feb 23 11:45:21 EST

ZK-1701U-AI

HP Insight Manager のブラウザの要件については、insight\_manager(5) を参照してください。HP Insight Manager では、Java、JavaScript、および cookie を有効にする必要があります。

## 2.9.1 HP Insight Manager の起動

HP Insight Manager を起動するには、管理したいクラスタ・メンバ上で次の URL を開いて、HP Insight Management Agents セクションにナビゲートします。

```
http://cluster_member_name:2301
```

ナビゲーション・フレームには、データを取得できるすべての下位構成要素と関連データ項目が一覧表示されます。また、ネットワーク・インタフェース・カード (NIC) のようなハードウェアに関する状態データや、CPU 使用率のような一般的なシステム状態に関するデータも表示されます。このフレームのコンテンツは、HP Insight Manager で利用できるデバイス・データによって変わります。一般的なカテゴリは、次のとおりです。

- 構成
- 大容量ストレージ
- NIC
- 使用率
- 回復

## 2.10 Tru64 UNIX 構成レポートの使用

HP Insight Manager に用意されている機能のほかに、Web ブラウザを使用してクラスタ・メンバ上でシステム検査を実行できます。このシステム検査では、`sys_check` コマンドが実行されますが、そのためには、コマンド行から `sys_check` を実行する場合と同じ特権が必要です。

新しいレポートを生成する場合は、ブラウザから SysMan Menu Java アプレットが起動されます。アプレットが正しく動作するように、2.6 節で記述されている互換性のあるブラウザを使用する必要があります。

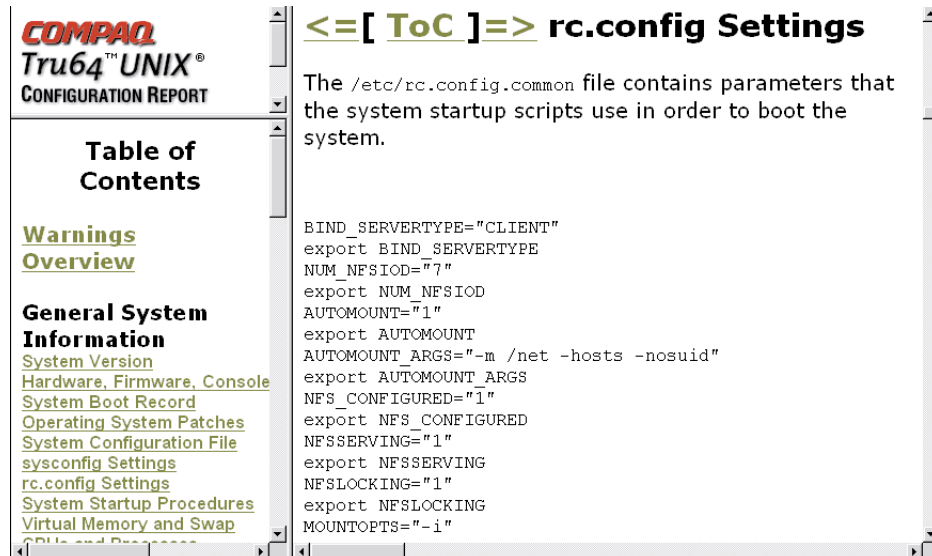
任意のブラウザを使用して、生成されたレポートを表示することができます。

構成レポートを起動するには、管理したいクラスタ・メンバ上で次の URL を開きます。

```
http://cluster_member_name:2301/ConfigReport
```

構成レポート表示のサンプルを図 2-8 に示しています。

図 2-8: 構成レポート表示のサンプル



ZK-1696U-AI

新しいシステム構成レポートを生成するには、Web ページの最下部の [Create New Report] をクリックします。この操作の結果、ブラウザ内の SysMan Menu Java アプレットが起動されます。このアプレットにより、レポートを作成する前に参照したい情報の種類を指定できます。

ブラウザからこの新しいレポートを表示するには、必ず [Export To Web] オプションを選択します。



---

## クラスタ別名サブシステムの管理

クラスタの管理者は、クラスタ別名の数、各クラスタ別名のメンバシップ、およびクラスタ別名の各メンバの属性を制御します。また、`/etc/clua_services` ファイル内でポートに割り当てられたクラスタ別名属性を制御します。

この章では次の項目について説明します。

- クラスタ別名の特徴についての要約 (3.1 節)
- クラスタ別名の構成と動作を制御するファイル (3.2 節)
- 一般的な設計に関する考慮事項 (3.3 節)
- 別名の作成準備 (3.4 節)
- クラスタ別名の指定とクラスタ別名への参加 (3.5 節)
- クラスタ別名属性とサービス・ポート属性の修正 (3.6 節)
- クラスタ別名のモニタ (3.7 節)
- クラスタ別名からのメンバの削除 (3.8 節)
- クラスタ別名の削除 (3.9 節)
- アプリケーションの負荷分散 (3.10 節)
- クラスタ単位のポート・スペースの拡張 (3.11 節)
- クラスタ別名の vMAC サポートの有効化 (3.12 節)
- ルーティング構成のガイドライン (3.13 節)
- ルーティング・デーモンと静的ルーティング・オプション (3.14 節)
- クラスタ別名と NFS (3.15 節)
- クラスタ別名と CAA (Cluster Application Availability) (3.16 節)

クラスタ別名の構成には `cluamgr` コマンドと SysMan Menu の両方を使用できます。

- `cluamgr` コマンド行インタフェースは、実行元のクラスタ・メンバを対象に、クラスタ別名関連のパラメータを構成します。パラメータはただちに有効になりますが、そのメンバの `clu_alias.config` ファイルにもコマンド行を追加しておかないと、リブートした後、無効になってしまいます。
- SysMan Menu グラフィカル・ユーザ・インタフェース (GUI) は、クラスタのすべてのメンバを対象に、クラスタ別名関連の静的なパラメータを構成します。静的パラメータはメンバの `clu_alias.config` ファイルに書き込まれますが、次にブートし直すまでは有効になりません (新しいパラメータをただちに有効にしたい場合には、SysMan Menu がメンバの `clu_alias.config` ファイルに書き込んだコマンド行を手動で実行してください)。

### 3.1 クラスタ別名の特長についての要約

TruCluster Server 『クラスタ概要』のクラスタ別名に関する章で、クラスタ別名の概念について説明しています。クラスタ別名属性またはサービス属性を修正する前に、この章を参照してください。

次に、クラスタ別名の重要な特長についてまとめます。

- 1 つのクラスタ内では、さまざまなメンバの集合から複数のクラスタ別名を作成できます。
- クラスタには省略時のクラスタ別名が 1 つあります。省略時のクラスタ別名としてクラスタ名が使用されます。
- クラスタ別名はドメイン・ネーム・システム (DNS) 名ではなく IP アドレスによって定義されます。クラスタ別名の IP アドレスとして、共通サブネットまたは仮想サブネット上の IP アドレスを使用できます。
- クラスタのメンバは、クラスタ別名へのルートを公開するには、クラスタ別名を指定する必要があります。また、クラスタ別名宛の接続要求またはパケットを受信するには、クラスタ別名に参加しなければなりません。
- クラスタ別名の管理はメンバ単位で行われます。1 つのメンバ上で `cluamgr` コマンドを入力すると、このコマンドはそのメンバのみに影響します。



- クラスタ・メンバだけがクラスタ別名に参加できます。クラスタ別名と、クラスタ・メンバ上の特定のネットワーク・インタフェース・カード間には、1 対 1 の対応はありません。また、クラスタ別名と、1 つ以上のクラスタ・メンバ上の特定のアプリケーション間にも、1 対 1 の対応はありません。クラスタ別名は、クラスタの 1 つ以上のメンバが共通 IP アドレスに応答できるようにするための汎用メカニズムです。クラスタ別名と CAA (Cluster Application Availability) サブシステムとの違いについての説明は 3.16 節を参照してください。
- `/etc/clu_alias.config` ファイルは、メンバ固有のクラスタ別名構成ファイルを指すコンテキスト依存シンボリック・リンク (CDSL) です。ファイル内のコマンドは、ブート時に実行されます。各メンバのこのファイルには、次の処理をする `cluamgr` コマンドが記述されています。
  - 省略時のクラスタ別名を指定して参加する。`clu_create` および `clu_add_member` コマンドを使って、新しいメンバの `clu_alias.config` ファイルに次の行を追加する。  

```
/usr/sbin/cluamgr -a selw=3,selp=1,join,alias=DEFAULTALIAS
```

 (クラスタ別名サブシステムが、`DEFAULTALIAS` キーワードをクラスタの省略時のクラスタ別名に自動的に対応させる。)
  - そのメンバがルートを公開するか参加するクラスタ別名があれば、その名前を指定する。
  - クラスタ別名属性を設定する。たとえば、ルーティング時の選択重みとルータ優先順位を設定する。

ブート時、クラスタの各メンバはそれぞれ固有の `clu_alias.config` を読み取るので、クラスタ別名の定義とメンバシップはリブートしても保持されます。このファイルの編集は手作業でも行うことができますが、SysMan Menu を使用して行うことを推奨します。SysMan Menu を使用して行った編集は、次のブート時まで有効になりません。新しい値を直ちに有効にするには、`cluamgr` コマンド行を手動で実行します。

- 名前が `/etc/exports.aliaes` ファイルの中に指定されているクラスタ別名のメンバは、その別名に宛てられたネットワーク・ファイル・システム (NFS) の要求を受け付けます。これにより、省略時のクラスタ別名とは異なる別名を NFS サーバとして使用できるようになります。詳細は、3.15 節を参照してください。

- TruCluster Server バージョン 5.1B より前のリリースでは、ルーティング・デーモンとして `gated` をクラスタで実行する必要がありました。`ogated`、`routed` が使用できず、また、ルーティング・デーモンを使わない静的ルーティングのセットアップもできませんでした。バージョン 5.1B からは、`gated`、`routed`、または静的ルーティングが使用できます。`ogated` は引き続き使用できません。ルーティング・オプションについては、3.14 節で説明しています。

省略時の構成では、クラスタ別名サブシステムは `gated` とルーティング情報プロトコル (RIP) を使用して、ホスト・ルートに別名アドレスを公開します。`cluamgr` に `nogated` オプションを指定すると、この処理を無効にすることができます。例としては、`routed -q` や `gated` をサイトごとにカスタマイズする構成ファイルで使用したり、または静的ルーティングを使用したりできます。

`gated` がルーティング・デーモンとして使用されると、別名デーモンの `aliasd` は、必要に応じて、クラスタ・メンバの `/etc/gated.conf.membern` ファイルにホスト・ルート・エントリを追加します。別名デーモンはメンバの `gated.conf` ファイルを変更することはありません。

- クラスタ別名経由でアクセスされるサービスが使用するネットワーク・ポートは、`in_single` または `in_multi` を使って定義します。この定義は、サービスが同時に複数のクラスタ・メンバ上で実行されるかどうかには無関係です。この定義に従って、クラスタ別名サブシステムは次のように動作します。
  - サービスを `in_single` として指定すると、クラスタ別名の 1 つのメンバのみが、そのサービス宛の接続要求とパケットを受信する。そのメンバが使用できなくなった場合、クラスタ別名サブシステムは、そのサービス宛のすべての接続要求とパケットの受信先として、クラスタ別名の別のメンバを選択する。
  - サービスを `in_multi` として指定すると、クラスタ別名サブシステムは、そのサービス宛の接続要求とパケットをそのクラスタ別名の正式メンバすべてに送信する。

省略時の構成では、クラスタ別名サブシステムは、すべてのサービスを `in_single` とみなします。クラスタ別名サブシステムがサービスを `in_multi` とみなすようにするには、`/etc/clua_services` ファイルを編集するか、`clua_registerservice()`、`clusvc_getcommport()`、

または `clusvc_getresvcommport()` 関数を呼び出して、そのサービスを `in_multi` として登録しなければなりません。

- 各クラスタ別名の識別に次の属性が使用されます。

クラスタ単位の属性:

- IP アドレスとサブネット・マスク: クラスタ別名を識別する。

メンバ単位の属性:

- ルータ優先順位: 共通サブネット上でこの別名に対するプロキシ ARP (Address Resolution Protocol) 要求にどのクラスタ別名のメンバが応答するかを決定する。
- 選択優先順位: クラスタ別名内でメンバに優先順位を付けて論理的にグループ化する。選択優先順位を使用することで、通常時に別名のどのメンバが要求を処理するかを制御することができる。選択優先順位の最も高いメンバが動作していれば、それより選択優先順位の低いメンバには要求が割り当てられない。選択優先順位は、別名メンバのフェイルオーバー順位を決める方法であると考えられることもできる。
- 選択重み: `in_multi` が指定されている場合に、クラスタ別名のメンバ間の静的な負荷分散を制御する。これは、別名のどのメンバに最もよく接続を受け持たせるかを制御する単純な方法である。選択重みとは、接続を選択優先順位の同じ次の別名メンバへ回すまでにそのメンバが受け持つ (平均的な) 接続数を示す。

## 3.2 構成ファイル

クラスタ別名およびサービスを管理するには、次の構成ファイルを使用します。

`/sbin/init.d/clu_alias`

これは、クラスタ別名サブシステムのブート時の起動スクリプトです。

`/etc/clu_alias.config`

これは、`/sbin/init.d/clu_alias` スクリプトから呼び出されるメンバ固有の `clu_alias.config` ファイルへの CDSL です。各メンバの `clu_alias.config` ファイルには、ブート時に `cluamgr` コマンドを実行してそのメンバのクラスタ別名 (省略時のクラスタ別名を含む) を構成するとともに、そのクラスタ別名へ参加する指定が記述されています。

/etc/clua\_services

これは、クラスタで提供されクラスタ別名でアクセスされるインターネット・サービス用のポート番号、プロトコル、および接続属性を定義するファイルです。cluamgr コマンドは、このファイルをブート時に読み取り、各サービスをそれぞれの属性とともに登録します。

clua\_services ファイルを修正する場合は、クラスタの各メンバ上で cluamgr -f を実行します。詳細は、clua\_services(4) および cluamgr(8) を参照してください。

/etc/exports.aliases

メンバの中に NFS 要求を受け付けるメンバがいるクラスタ別名を定義するファイルです (1 行に 1 つずつ定義します)。省略時の設定では、省略時のクラスタ別名だけが NFS の要求を受け付けることができます。NFS サーバとして他の別名を追加指定する場合は、/etc/exports.aliases ファイルを使用します。

/etc/gated.conf.membern

省略時の設定では、クラスタの各メンバのクラスタ別名デーモン (aliasd) は、メンバ固有の /etc/gated.conf.membern ファイルを作成します。このデーモンはその後 gated を起動するとき、gated の構成ファイルとしてこのファイルを使用します。メンバの構成ファイル /etc/gated.conf は使用しません。

cluamgr -r stop を使ってクラスタのメンバによるクラスタ別名へのルーティングを停止した場合、クラスタ別名デーモンは gated を再起動するとき、gated の構成ファイルとしてそのメンバの gated.conf を使用します。

### 3.3 一般的な設計に関する考慮事項

クラスタでどのようにクラスタ別名を構成するかを設計するときには、次の事項を検討します。

- クラスタがクライアント (たとえばメール・ハブ、アプリケーション・サーバ、NFS サーバなど) にどのようなサービスを提供するか。
- クライアントからの要求に効果的に応答するには何個のクラスタ別名が必要か。

省略時のクラスタ別名のみで十分な場合もあります。しばらくの間、省略時のクラスタ別名のみを使用して見て、その後、自分の構成にとってクラスタ別名を追加する必要があるかどうかを決定します。

- `/etc/services` 内に列挙されている一般的なインターネット・サービスがクラスタにある場合は、省略時のクラスタ別名で十分です。
- 省略時の設定では、クラスタがネットワーク・ファイル・システム (NFS) サーバとして設定されている場合、外部クライアントはクラスタによってエクスポートされたファイル・システムをマウントするときに、省略時のクラスタ別名を NFS サーバ名として使用しなければなりません。ただし、クラスタ別名を新しく作成してそれを NFS サーバとして使用することもできます。この機能は、本書の 3.15 節、『クラスタ概要』、および `exports.aliases(4)` で説明しています。
- クラスタが多くのシステム・リソースを使用するサードパーティのアプリケーションを実行している場合は、そのアプリケーションに対するアクセス制御と負荷分散を行うために、クラスタ別名の追加が必要になります。

Web サーバとして使用されるクラスタで、2 つのクラスタ別名を使ってアクセス制御と冗長構成を実現する例を、3.10 節に示しています。

- クラスタの外部ネットワーク・トポロジに基づいてルーティングのストラテジを決定する。使用可能なルーティング・デーモン・オプションについては、3.14 節で説明。
- クラスタのどのメンバがどのクラスタ別名に属するか。

作成する別名に一部のクラスタ・メンバが参加しない場合に、その別名を経由してアクセスすると、その別名の少なくとも 1 つのメンバから対応するサービスを受けられることを確認してください。次に例を挙げます。

- 別名を作成して NFS サーバとして使用する場合、そのメンバがすべて、エクスポートされるファイル・システムのあるストレージに直接接続されていることを確認してください。
- CAA で制御するアプリケーションにクラスタ別名を経由してアクセスさせる場合、CAA の配置ポリシーが、そのクラスタ別名のメンバで

はないクラスタ・メンバでサービスを開始しないようになっていることを確認してください。

- 各メンバは自分が指定するか参加する各クラスタ別名にどの別名属性を割り当てるか。

最初は属性の省略時の設定を使用して (`rpri=1,selw=1,selp=1`) , 後から属性を変更することもできます。

- サービス属性を追加する場合は, `/etc/clua_services` 内のインターネット・サービスのエントリにどの属性を関連付けるか (このファイルは, インターネット・サービスのストック・セットに対する省略時の別名ポート設定を提供する)。
- クラスタ別名の IP アドレスは, 既存の共通サブネット上に存在するか, 仮想サブネット上に存在するか, あるいはその両方に存在するか。

共通サブネット上に存在する場合: クラスタが接続されている既存のサブネットからクラスタ別名の IP アドレスを選択します。

---

#### 注意

共通サブネット内のクラスタ別名に対してはプロキシ・アドレス解決プロトコル (ARP) が使用されるので, ローカル・エリア・ネットワーク (LAN) でプロキシ ARP をブロックするルータまたはスイッチが使用されている場合, そのクラスタ別名はローカル以外のセグメントからは見えません。したがって, 共通サブネットの構成では, クラスタ別名のクライアントを接続するルータまたはスイッチの構成で, プロキシ ARP のブロックを有効にしてはなりません。

仮想サブネット上に存在する場合: 仮想サブネット内のクラスタ別名のホスト・ルートは, クラスタ別名ソフトウェアによって自動的に構成されます。クラスタ・メンバは, メンバの指定または参加に際して `virtual` 属性を追加する場合, その仮想サブネットへ至るネットワーク・ルートも公開します。

---

#### 注意

仮想サブネットには, 実際のシステムを置くことはできません。

`nogated` オプションを指定すると、仮想サブネット上に存在する別名へのルートは公開されません。クライアント・システムが静的ルートをこの別名に対して構成していなければ、この別名は到達可能とはなりません。

---

サブネットの種類を選択するときは主に、既存のサブネット(つまり共通サブネット)上にクラスタ別名に使用できる IP アドレスが十分にあるかどうかを検討します。既存のサブネット上にすぐに使用できる IP アドレスがない場合は、仮想サブネットの作成を考慮するとよいでしょう。ただし、クラスタが複数のサブネットに接続されている場合に、仮想サブネットを作成すると、クラスタに接続されているすべてのサブネットから同じアドレスを使ってその仮想サブネットへ到達できるようになる、という利点については、あまり考慮する必要がありません。この利点は、実質よりもスタイルの問題です。クラスタ別名用の IP アドレスを確保するためにはどちらの種類のサブネットを使用しても、実質的に大きな違いはありません。サイトで最適な方法を選択してください。

### 3.4 クラスタ別名の作成準備

`cluamgr` コマンドを使用して、クラスタ別名の指定およびクラスタ別名へ参加する前に、次の手順を実行します。

1. 各クラスタ別名については、サイトが使用する `hosts` テーブル(たとえば `/etc/hosts` , BIND (Berkeley Internet Name Domain) , または NIS (Network Information Service)) でクラスタ別名の IP アドレスがホスト名に関連付けられていることを確認します。

---

#### 注意

---

クラスタからのパスワード保護なしのログインとリモート・シェルを許可するようにクライアント上の `.rhosts` ファイルを修正する場合は、クラスタの各メンバのホスト名ではなく省略時のクラスタ別名を、ホスト名として使用します。これによってクラスタからのログイン要求では、送信元アドレスとして省略時のクラスタ別名が使用されます。

---

2. クラスタ別名の IP アドレスが仮想サブネット上に存在する場合、ローカルのルータには該当するサブネットを登録します (仮想サブネット内には実際のシステムを置くことができないことに注意してください)。
3. クライアントからの NFS 要求に応答するために、省略時のクラスタ別名以外に別名が必要な場合は、その名前を `/etc/exports.aliases` に追加します。
4. 固定ポートが割り当てられたサービスについては、  
`/etc/clua_services` 内でそのインターネット・サービスのエントリを調べます。新しい別名でアクセスされる追加のサービスがあれば、  
`/etc/clua_services` にそのサービスのエントリを追加します。

### 3.5 クラスタ別名の指定と参加

`cluamgr` コマンドは、クラスタ別名の指定、参加、および管理用のコマンド行インタフェースです。クラスタのメンバ上でクラスタ別名を指定すると、そのメンバはクラスタ別名を認識し、その別名へのルートを開示できるようになります。クラスタのメンバ上でクラスタ別名への参加を指定すると、そのメンバはそのその別名宛の接続要求またはパケットの受信ができるようになります。

クラスタ別名属性すべてに省略時の値を使用するクラスタ別名を指定するには (しかし、参加はしない)、次のコマンドを使用するのが最も簡単です。

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

このクラスタ・メンバで `cluamgr -r start` コマンドを使用して別名ルーティングを再起動すると、システムはその別名へのルートを開示して、ネットワークからパケットを取得し、それらを別名に参加している別名のメンバに転送することができます。このメンバで `cluamgr -s alias` コマンドを実行すると、この別名に対して `<ENABLED>` という状態が表示されます。

すべての属性に省略時の値を使用するクラスタ別名を指定し、同時にその別名への参加を指定するには、次のコマンドを使用するのが最も簡単です。

```
# cluamgr -a alias=alias,join
```

このクラスタ・メンバで `cluamgr -r start` コマンドを使用して別名ルーティングを再起動すると、システムは別名へのルートを開示して、ネットワークからパケットを取得し、それらを別名に参加している別名のメンバに転送し、その別名宛での接続要求またはパケットを受信できます。



このメンバで `cluamgr -s alias` コマンドを実行すると、この別名の `<JOINED,ENABLED>` という状態が表示されます。

クラスタ別名を指定する場合とクラスタ別名に参加する場合は、次の手順を実行します。

1. 別名のホスト名と IP アドレスを取得します。
2. SysMan Menu を使ってクラスタ別名を追加します。クラスタ別名属性に省略時の値を使用しないときは、適切な値に変更します。たとえば、`selp` または `selw` の値を変更します。

SysMan Menu は、メンバの `clu_alias.config` ファイルにコマンド行を書き込むだけです。メンバの `clu_alias.config` ファイルに別名を書き込むだけなので、その別名はすぐには有効とならず、次のブート後に初めて有効となります。

次の例は、クラスタの 1 つのメンバの `clu_alias.config` ファイルにある `cluamgr` コマンド行を示したものです。別名の IP アドレスはすべて共通サブネット内にあります。

```
/usr/sbin/cluamgr -a alias=DEFAULTALIAS,rpri=1,selw=3,selp=1,join
/usr/sbin/cluamgr -a alias=clua_ftp,join,selw=1,selp=1,rpri=1,virtual=f
/usr/sbin/cluamgr -a alias=printall,selw=1,selp=1,rpri=1,virtual=f
```

3. メンバ上で該当する `cluamgr` コマンドを手動で実行して、別名を指定するかまたは別名に参加した後、別名ルーティングを再起動します。たとえば、次のように実行します。

```
# cluamgr -a alias=clua_ftp,join,selw=1,selp=1,rpri=1
# cluamgr -a alias=printall,selw=1,selp=1,rpri=1
# cluamgr -r start
```

上記の例では、`virtual` 属性の省略値が `f` であるため、2 つの別名に対して `virtual=f` を明示的には指定していません。`cluamgr -r start` コマンドはカーネル・ルーティング情報を更新して、このメンバ上のクラスタ別名サブシステムが新しい別名に対してルーティングできるようにします。

前述したように、別名へ参加する場合に別名属性として省略値を使用するときは、次のコマンドで十分です。

```
# cluamgr -a alias=alias_name,join
```

次の例は、仮想ネットワーク内でクラスタ別名の指定および別名へ参加する方法を示しています。この構成方法は、共通サブネット内のクラスタ別名の構成方法とほとんど同じです。

```
# cluamgr -a alias=virtestalias,join,virtual,mask=255.255.255.0
```

この `cluamgr` コマンドを構成する要素は次のとおりです。

**cluamgr -a alias=virtestalias**

別名の指定。`cluamgr -r start` コマンドを実行後、クラスタ・メンバが別名へのホスト・ルートの公開を開始する。

**join**

別名に参加。メンバは別名宛のパケットを受信できる。

**virtual**

`virtestalias` アドレスが属する仮想ネットワークへのネットワーク・ルートを公開する。

**mask=255.255.255.0**

サブネット・マスクを明示的に定義する。サブネット・マスクを指定しない場合、このメンバの別名デモンは、仮想サブネットが公開されるこのシステムの最初のインタフェースのネットワーク・マスクを使用する。

クラスタのメンバが仮想サブネットのネットワーク・ルートを公開しないようにする場合には、仮想サブネット内の別名に対して `virtual` または `virtual=t` を指定しないでください。たとえば、あるクラスタ・メンバで次のコマンドを実行すると、そのクラスタ・メンバは `virtestalias` 別名を指定し参加しますが、ネットワーク・ルートは公開しません。

```
# cluamgr -a alias=virtestalias,join
```

仮想サブネット内の別名の構成についての詳細は、`cluamgr(8)` を参照してください。

## 3.6 クラスタ別名属性とサービス・ポート属性の変更

クラスタ別名属性を変更するために、クラスタの任意のメンバ上で `cluamgr` コマンドをいつでも実行できます。たとえば、クラスタ別名 `clua_ftp` のメ

ンバに対してその別名の選択重みを変更するには、そのメンバで次のコマンドを入力します。

```
# cluamgr -a alias=clua_ftp,selw=2
```

この変更がリブート後も有効であるようにするには、このメンバの `clu_alias.config` ファイルでこの別名のエントリを変更します。

クラスタ単位の `/etc/clua_services` に登録されているインターネット・サービスに関連付けられているサービス・ポート属性を変更するには、次の手順を実行します。

1. `clua_services(4)` の情報を使用して、`/etc/clua_services` 内のエントリを変更します。
2. このファイルを `cluamgr` に再度読み取らすために、クラスタの各メンバ上で次のコマンドを入力します。

```
# cluamgr -f
```

---

#### 注意

`clua_services` ファイルを再度読み込んでも、現在実行中のサービスには影響しません。構成ファイルを再度読み込んだ後、サービスを停止して再起動する必要があります。

たとえば、`telnet` サービスは `/etc/inetd.conf` から `inetd` が起動します。`clua_services` にある `telnet` のサービス属性を変更した場合は、`cluamgr -f` を実行してから、`inetd` を停止した後再起動して変更が有効になるようにしなければなりません。そうしなければ、次にリブートするまで変更が有効になりません。

---

## 3.7 クラスタ別名のモニタ

このクラスタ・メンバが認識しているクラスタ別名の状態を表示するには、たとえば次のように `cluamgr -s all` コマンドを使用します。

```
# cluamgr -s all
```

```
Status of Cluster Alias: deli.zk3.dec.com
```

```
netmask: 0
aliasid: 1
flags: 7<ENABLED,DEFAULT,IP_V4>
connections rcvd from net: 72
connections forwarded: 14
```

```
connections rcvd within cluster: 52
data packets received from network: 4083
data packets forwarded within cluster: 2439
datagrams received from network: 28
datagrams forwarded within cluster: 0
datagrams received within cluster: 28
fragments received from network: 0
fragments forwarded within cluster: 0
fragments received within cluster: 0
Member Attributes:
memberid: 1, selw=3, selp=1, rpri=1 flags=11<JOINED,ENABLED>
memberid: 2, selw=2, selp=1, rpri=1 flags=11<JOINED,ENABLED>
```

---

### 注意

---

netstat -i を実行してもクラスタ別名は表示されません。

---

共通サブネット内のクラスタ別名の場合、各メンバ上で arp -a を実行して、どのメンバがクラスタ別名へのルーティングを行うかをチェックできます。たとえば、クラスタ別名と permanent published を検索するには、次のように実行します。

```
# arp -a | grep permanent
deli (16.140.112.209) at 00-00-f8-24-a9-30 permanent published
```

## 3.8 クラスタ別名からのメンバの削除

参加先のクラスタ別名からクラスタ・メンバを削除するには、そのメンバ上で次のコマンドを入力します。

```
# cluamgr -a alias=alias,leave
```

ただし、別名へのルートを公開するように構成されているメンバは、削除された後もこの別名へのルートを公開できますが、この別名宛の接続要求とパケットは受信できなくなります。ただし、クラスタ別名が構成されると、クラスタは、その別名の最後のメンバが別名から削除された後でも、その別名宛てのパケットを拒否しません。クラスタ別名の削除についての詳細は、3.9 節を参照してください。

## 3.9 クラスタ別名の削除

メンバの別名のカーネル・リストからクラスタ別名を削除する cluamgr コマンドはありません。その別名を認識しているすべてのクラスタ・メンバが別名から削除されても (cluamgr -a alias=alias,leave)、クラスタは引き続きその別名宛ての接続要求を受け付けます。

クラスタ・メンバが有効にされるか (`cluamgr -a alias=alias`) または参加して (`cluamgr -a alias=alias,join`)、別名のルーティングが再起動される (`cluamgr -r start`) と、`aliasd` はその別名のエントリをメンバの `gated.conf.membern` ファイルに追加し、別名はそのメンバの別名のカーネル・リストに追加されます。

クラスタ・メンバがブートすると、カーネルはそのメンバの `/etc/clu_alias.config` ファイル内の情報からクラスタ別名のリストを構築します。`aliasd` デーモンは、`clu_alias.config` ファイルからではなく、カーネル内の別名リストからその情報を取得します。したがって、ファイルから別名エントリを削除して、別名デーモンを再起動するか、または別名ルーティングを再起動しても、別名は削除されません。

クラスタ別名を削除するには、次の手順に従います。

---

#### 注意

---

省略時のクラスタ別名を削除することはできません。クラスタの名前を変更する必要がある場合には、本書で説明されている手順を使用します。

---

1. `/etc/hosts` ファイルからクラスタ別名のエントリを削除します (技術的には必要ありませんが、エントリを削除またはコメント・アウトすると管理上わかりやすくなります)。
2. 各クラスタ・メンバで、`cluamgr -s alias` コマンドを実行して、この別名を認識しているクラスタ・メンバを判断します。
3. この別名を認識している各クラスタ・メンバで、そのメンバの `/etc/clu_alias.config` ファイルからその別名に関連するエントリをすべて削除します (エントリを削除しておくと、システムのリブート時にその別名が確実に再構成されなくなります)。
4. 1度に1つずつ、削除する別名のメンバだった各クラスタ・メンバをリブートします。その別名を認識していたすべてのクラスタ・メンバがリブートされると、別名は削除されます。

## 3.10 アプリケーションの負荷分散

この節では、クラスタ別名のメンバ間でのアプリケーションの負荷分散について説明します。ネットワーク・インタフェースの負荷分散に役立つルーティング・デーモン構成オプションについての情報は、3.14 節を参照してください。

---

### 注意

---

負荷分散の概念は、`/etc/clua_services` ファイル内で `in_multi` サービス属性が割り当てられているサービスだけに適用されます。クラスタ別名サブシステムは、そのサービスが `in_multi` として明示的に定義されていない限り、`in_single` サービスとして扱います。`in_single` サービスに対するパケットと要求はすべて別名のメンバへ送られますが、同時に 2 つ以上の別名メンバへ送られることはありません。

クラスタ別名サブシステムは、クラスタの各メンバの性能を監視せず、`in_multi` のサービスの負荷分散も自動的に行いません。接続要求の分配を制御するには、クラスタ別名の各メンバに選択優先順位と選択重みを割り当てます。これらの属性の値はいつでも手作業で修正できます。

選択優先順位属性 `selp=n` は、別名のメンバが新しい接続要求を受信する順番を決定します。`n` の値は 1 以上 100 以下の整数でなければなりません。別名の選択優先順位を使用することで、別名内に階層を作成することができます。

たとえば、4 つのクラスタ・メンバがクラスタ別名に参加していて、別名に対し `selp` を次の値に設定しているとします。

- メンバ A および B は `selp=5`
- メンバ C および D は `selp=4`

`selp=5` のメンバのどれかが要求に応答できれば、要求が `selp=4` のメンバに送られることはありません。つまり、メンバ A または B が要求を処理できる場合、C と D がこの別名に宛てられたパケットや要求を受信することはありません。

この例では、選択優先順位を使用して、クラスタ別名のメンバ間にフェイルオーバーの階層が作成されます。このタイプの階層を作成する 1 つの理由

は、メンバ A と B が、クライアントがこのクラスタ別名でアクセスするストレージに直接接続されているためです。したがって、A と B にこの別名宛てのクライアント要求をサービスさせます。ほとんどの場合、C と D はクライアント要求を受信しませんが、A と B のどちらも応答できない場合には、引き受けることができます。

メンバがクラスタ別名に割り当てる選択重み  $selw=n$  は、そのメンバに送信される要求の平均数 (アプリケーション単位) に変換されます。変換された後、要求は同じ選択優先順位の次の別名メンバに送信されます。 $n$  の値は、1 以上 100 以下の整数でなければなりません ( $selp$  の値は、要求またはメッセージの受信に適したメンバの順番を決定し、 $selw$  の値は、適切なメンバになったのち、そのメンバが受信する要求またはメッセージの数を決定します)。

たとえば、1 つのクラスタ別名に 4 つのクラスタ・メンバが参加しているものとします。4 つのクラスタ・メンバすべてが別名に対して同じ  $selp$  値を持ち、 $selw$  を次の値に設定しているとします。

- メンバ A および B は  $selw=3$
- メンバ C および D は  $selw=2$

別名のすべてのメンバが同じ選択優先順位を持っているので、ラウンド・ロビン・アルゴリズムに従ってメンバ・リスト内からこれらのメンバが選択され、 $selw$  に従って各メンバに順番に要求が分配されます (クラスタ別名サブシステムは、どのサービス・ポートで別名のどのクラスタ・メンバが待機しているかを追跡し、接続要求およびパケットを分散する方法を決める際にこの情報を使用します)。特定のサービス (たとえば、telnet) に対してこの別名に宛てられたクライアント要求について、メンバ A は 3 つの要求を受信し、メンバ B は 3 つの要求を受信し、メンバ C は 2 つの要求を受信する、という具合に続きます。

クラスタ別名のメンバに選択重みを割り当てるときは、クラスタ別名経由でアクセスされるアプリケーションのリソースに最もよく一致するリソースを持つメンバに、他のメンバよりも高い選択重みを割り当てます。

---

#### 注意

---

省略時のクラスタ別名は  $selw$  として 3 を持っています。たとえば、Secure Shell (SSH) を使用するようないくつかのアプリケーションは、単一の接続を確立するために 2 つの接続を使用します。このため、SSH を使用する省略時のクラスタ別名への接続要

求は、あるシステムは2つの接続を取得し、別のシステムは1つの接続を取得するように分散されますが、実際のところ、分散の不均衡はありません。接続要求は各クラスタ・メンバに割り当てられた選択重みに従って分散されています。

省略時のクラスタ・メンバ間でSSHを分散させるように接続するためには、省略時のクラスタ別名のすべてのクラスタ・メンバでselwを任意の偶数に設定します。この変更を永続的なものにするため、次の行を各メンバの/etc/clu\_alias.configファイルに追加します。

```
/usr/sbin/cluamgr -a selw=4,selp=1,join,alias=DEFAULTALIAS
```

シェル・スクリプトに精通している管理者であれば、スクリプトを記述して、クラスタ・メンバの性能の監視と、その情報に基づく選択重みの増減を自動化することもできます。この場合の性能は、監視対象のメンバ上のアプリケーションに関連する要素すべてによって決まります。

今度は、次のような状況を仮定します。管理者は、ファイルのアーカイブ保存を主目的とするWebサイトとして、4メンバ構成のクラスタを構成します。ユーザは、このサイトに接続し、大きなサイズのファイルをダウンロードします。クラスタは、共通ネットワークに接続されている4つのメンバで構成されます。クラスタ内では、メンバAおよびBが1つのディスク・セットを共用し、メンバCおよびDが別のディスク・セットを共用します。メンバAおよびBのネットワーク・インタフェースは、大容量データ転送（たとえばftp転送）用に調整されています。メンバCおよびDのネットワーク・インタフェースは、短いタイムアウトと待ち時間（たとえばWebからの接続）用に調整されています。

この状況で、cluftpとcluhttpという2つのクラスタ別名を定義します。クラスタの4つのメンバはすべて両方のクラスタ別名に参加しますが、値は異なります。

メンバAおよびBの/etc/clu\_alias.configファイル内に次のような行があります。

```
/usr/sbin/cluamgr -a alias=cluftp,selw=1,selp=10,join  
/usr/sbin/cluamgr -a alias=cluhttp,selw=1,selp=5,join
```

メンバCおよびDの/etc/clu\_alias.configファイル内に次のような行があります。



```
/usr/sbin/cluamgr -a alias=clu_ftp,selw=1,selp=5,join  
/usr/sbin/cluamgr -a alias=clu_http,selw=1,selp=10,join
```

その結果、少なくともメンバ A と B の一方が稼働していれば、その稼働中のメンバがすべての ftp 要求に応答します。同様に、少なくともメンバ C と D の一方が稼働していれば、その稼働中のメンバがすべての http 要求に応答します。ただし、4 つのメンバすべてが両方のクラスタ別名に属しているため、どちらかの別名で 2 つの一次サーバがダウンしても、残りの別名メンバがクライアントからの要求に応答します (クォーラムが維持されていることが前提です)。

### 3.11 クラスタ単位のポート・スペースの拡張

サービス用にクラスタ全体で使用可能な一時 (動的) ポートの数は、サブシステム属性 `inet`、`ipport_userreserved_min` (省略時の値は 1024)、および `ipport_userreserved` (省略時の値は 5000) によって決まります。

ポート・スペースがクラスタのすべてのメンバ間で共用されるので、複数のメンバが属しているクラスタでは、使用可能なポートの競合が起こる可能性があります。クラスタに 3 つ以上のメンバがある場合は、`ipport_userreserved` に最大許容値 (65535) を設定することをお勧めします (`ipport_userreserved = 65535` にしても何ら悪影響はありません)。

`ipport_userreserved` に最大値を設定するには、次の手順に従います。

1. クラスタの任意のメンバ上で、クラスタ単位の `/etc/sysconfigtab.cluster` ファイルに次の行を追加して、次のリブートで `ipport_userreserved` が 65535 に設定されるように構成します。  

```
inet:  
  ipport_userreserved=65535
```
2. クラスタの各メンバ上で `sysconfig` を実行して、`ipport_userreserved` の現在の値を変更します。  

```
# sysconfig -r inet ipport_userreserved=65535
```

### 3.12 クラスタ別名の vMAC サポートの有効化

vMAC サポートを有効にすると、クラスタの 1 つのメンバがクラスタ別名のプロキシ ARP マスタになり、クラスタ別名の vMAC アドレスを生成します。vMAC アドレスは、接頭部 (省略時は AA:01) と 16 進形式のクラスタ別名の IP アドレスから構成されます。たとえば、クラスタ別名の IP

アドレスが 16.140.112.209 であるとする、省略時の vMAC アドレスは AA:01:10:8C:70:D1: になります。

省略時の vMAC 接頭部:	AA:01
クラスタ別名 IP アドレス:	16.140.112.209
IP アドレス (16 進形式):	10.8C.70.D1
このアドレスの vMAC:	AA:01:10:8C:70:D1

クラスタの別のメンバがこのクラスタ別名のプロキシ ARP マスタになると、vMAC アドレスはクラスタ別名とともに移動されるので、一貫した vMAC アドレスが各クラスタ別名の共通サブネット内に通知されます。

vMAC サポートを構成するときは、クラスタのすべてのメンバを同じ構成にします。この理由から、`/etc/rc.config.common` 内の vMAC 構成変数を設定する必要があります。

省略時の設定では、vMAC サポートは無効に設定されています。vMAC サポートを有効にするには、`rcmgr` を使って `/etc/rc.config.common` に適切なエントリを追加します。

```
# rcmgr -c set VMAC_ENABLED yes
```

反対に vMAC サポートを無効にするには、次のように入力します。

```
# rcmgr -c set VMAC_ENABLED no
```

vMAC アドレスの省略時の接頭部 AA:01 を変更するには、次のように入力します。

```
# rcmgr -c set VMAC_PREFIX xx:xx
```

各クラスタ・メンバの vMAC サポートを手動で有効にするには `cluamgr` のルーティング・オプション `vmac` を使用し、無効にするには `novmac` を使用します。たとえば、クラスタ・メンバで vMAC サポートを手動で有効にするには、次のように入力します。

```
# cluamgr -r vmac
```

各クラスタ・メンバで vMAC サポートを手動で無効にするには、次のように入力します。

```
# cluamgr -r novmac
```

クラスタのすべてのメンバの vMAC の構成は同一にする必要があるため、vMAC サポートを有効にするときは、次の手順の実行をお勧めします。

1. クラスタの任意のメンバ上で、次のように入力します。

```
# rcmgr -c set VMAC_ENABLED yes
```

これにより、ブート時に vMAC サポートが自動的に有効になります。ただし、この変数の設定はメンバがリブートするときのみ影響を与えるため、現在実行中のクラスタでは vMAC は有効になりません。

2. 現在実行中のクラスタで vMAC サポートを手動で有効にするには、各クラスタのメンバで次のコマンドを入力します。

```
# cluamgr -r vmac
```

各クラスタ・メンバの `/etc/clu_alias.config` ファイルに `cluamgr -r vmac` コマンドを追加する必要はありません。各メンバ上で `cluamgr -r vmac` コマンドを手動で実行すると、vMAC サポートが有効になります。共用の `/etc/rc.config.common` ファイルで `VMAC_ENABLED` を `yes` に設定すると、すべてのクラスタ・メンバで、ブート時に vMAC のサポートが自動的に有効になります。

クラスタ・メンバが、alt ドライバ・バージョンが V2.07 より小さい DEGPA のギガビット・イーサネットのアダプタを使用して外部のネットワークに接続され、vMAC が有効である場合には、ネットワークのインタフェースが `promiscuous` モード (`promisc`) に設定されていないと、メンバはクラスタの別名宛の外部からのパケットに応答しません (alt ドライバが V2.07 以降の DEGPA アダプタは、vMAC アドレスが構成されると、自動的に `promiscuous` モードを有効にします)。

次のコマンドを使って `promiscuous` モードを設定します。

```
# ifconfig alt0 promisc
```

ブート時にすべての DEGPA アダプタに対して `promiscuous` モードを自動的に有効にするには、各メンバの `/etc/inet.local` ファイルに次の行を追加します。

```
for i in `ls /usr/sbin/ifconfig -l` ; do
    case $i in
        alt*) ifconfig $i promisc ;;
    esac
done
```

『クラスタ概要』では VMAC の概要を説明しています。

### 3.13 ルーティング構成のガイドライン

クラスタ別名操作を行う場合は、クラスタに接続されたすべてのサブネットに、機能するルータが含まれている必要があります。この条件が満たされていれば、手動でルーティングの構成を行わなくてもクラスタ別名で接続できます。接続されているサブネットにルータがない場合は、一部手動でルーティングの構成を行う必要があります。この理由は、クラスタ・メンバ上のクラスタ別名デーモンでは、考えられるすべてのルーティング・トポロジに対する正しい経路の特定や検証を行うことができないためです。

クラスタに接続されているサブネット内のルータを構成できない場合、たとえば、1つのクラスタ・メンバが、非ルータしか含まれていない独立した LAN に接続されている場合は、そのサブネットに接続されていない各クラスタ・メンバ上で、そのサブネットへのネットワーク経路を手動で構成しなければなりません。ルータのないサブネットに接続されているメンバごとに、該当するサブネットへのネットワーク経路をそのメンバのクラスタ・インターコネクト・インタフェースに追加してください。

---

#### 注意

---

同じ LAN 上にあるクラスタであれば、同じ仮想サブネットを使用できます。

このようにできるのは、同じ LAN 上にあるルータであれば、どのルータでもクラスタ別名ごとにその個別ホスト・ルートを確認して正しいクラスタへパケットを送信できるからです。LAN の外に対しては、公開されているネットワーク・ルートを用いて仮想サブネットを公開するので、その仮想サブネット内のクラスタ別名アドレスへ宛てたパケットは、その LAN のルータに送られます。要約すると、(1) ホスト・ルートが生成され、(2) クラスタが同じ LAN を共有していれば、各クラスタに対して別々の仮想サブネットを使用する必要はない、ということです。

ただし、クラスタがマルチホームの場合は、複数のクラスタに対して同じ仮想サブネットを使用する方が複雑です。たとえば、2つのクラスタがあって、一方が LAN1 と LAN2 に、また、もう一方が LAN 1 と LAN 3 にそれぞれ接続されている場合、両方のクラスタに同じ仮想サブネットを使用すると、LAN 2 と LAN 3 に送

られてくるパケットに対してうまく動作しません。マルチホームの場合は、LAN へ同じように接続している必要があります。

### 3.14 ルーティング・デーモンと静的ルーティングのオプション

バージョン 5.1B より前には、クラスタ別名サブシステムが `routed` や、静的ルートと効果的に共存していなかったため、クラスタ別名ルーティングを正しく処理するために、ルーティング・デーモンとして `gated` を使用する必要がありました。このリリースでは、正しく動作するために `gated` を必要としない `aliasd` デーモンが提供されます。

サポートされているルーティング・デーモンを以下で説明します。

- `gated` と `routed` ルーティング・デーモンがクラスタでサポートされています。また、静的ルーティングがサポートされています (ルーティング・デーモンを必要としません)。

`aliasd` は `gated` 用に最適化されているので、`gated` が省略時の優先ルーティング・デーモンです。ただし、必ず必要ではなく、クラスタ・メンバでルーティングを構成する唯一の方法でもありません。例として、すべてのメンバが静的ルーティングを使用するクラスタを構成することも、あるメンバだけに `routed` を実行させることも、ルーティング・デーモンと静的ルーティングの組み合わせを使用することもできます。

ただし、`ogated` の使用に関する既存の制限は、引き続き適用されます。クラスタで `ogated` をルーティング・デーモンとして使用しないでください。

#### 注意

クラスタ・メンバは、同一のルーティング構成を持つ必要はありません。一般的には、すべてのクラスタ・メンバを同一に構成の方がより簡便ですが、経験を積んだクラスタの管理者であれば、メンバごとに別のルーティング・タスクを実行させるような構成を選んでよい場合もあります。たとえば、メンバの `/etc/rc.config` ファイル内に `CLUAMGR_ROUTE_ARGS="nogated"` と指定したり、完全に関連付けられた `/etc/routes` ファイルを使うこともでき

ます。または、メンバが `nogated` と `routed -q` を使って動作することもできます。

- 別名デーモンは動的ルーティングか静的ルーティングのどちらかで、クラスタ・インターコネクトを介してクラスタ別名の IP アドレスをフェイルオーバーします。たとえば、インタフェースが障害になった場合、`aliasd` は、クラスタの別のメンバに別名のトラフィックを再ルーティングします。クラスタ・インターコネクトが機能している限り、クラスタ別名トラフィックは発着信できます。
- サブネット単位の複数インタフェース (ネットワーク負荷分散)。 `gated` では、この構成がサポートされませんが、静的ルーティングがサポートされているので、管理者はネットワークの負荷分散のために静的ルーティング (`nogated`) を使えます。

省略時では、クラスタ別名サブシステムは `gated` とカスタマイズ `gated` 構成ファイル (`/etc/gated.conf.member<n>`) を使い、さらに別名アドレスのホスト・ルートを公開するために RIP を使います。この動作を無効にするには、`cluamgr` の `nogated` オプションを使い、メンバ上で `cluamgr -r nogated` コマンドを実行するか、メンバの `/etc/rc.config` ファイルで `CLUAMGR_ROUTE_ARGS="nogated"` を設定します。

クラスタにとって、3 つの一般的なルーティング構成のシナリオがあります。

- 省略時の構成: `aliasd` が `gated` を制御する動的ルーティング。
  - 各メンバの `/etc/rc.config` ファイルに次の内容を設定する。

```
GATED="yes"
```

`CLUAMGR_ROUTE_ARGS` がメンバの `/etc/rc.config` ファイルにある場合には、それをコメント・アウトするか、またはヌル文字に設定します。
  - 必要であれば、静的ルートをそのメンバの `/etc/routes` ファイルに定義する。

## 注意

`/etc/routes` ファイルの静的ルートのインストールは、ルーティング・デーモンを開始する前に行い、ルーティング・デーモンによって管理されます。

この構成では、`aliasd` は `gated` のオペレーションを制御します。ホスト・ルートは RIP を経由して伝播され、`aliasd` は、ネットワーク・インタフェース・カード (NIC) の障害のために孤立したノードの代わりに、`gated` にクラスタから外に向かうパスを習得させることができます。`aliasd` デーモンは、`gated` を停止して再起動することができ、`gated` からの干渉なしでルーティング・テーブルを変更することができます。

- 動的ルーティング (`gated` および `aliasd` が独立して動作する)。

クラスタ別名サブシステムと `aliasd` はメンバの `gated` の実行とは依存関係はありません。管理者は `gated` と各メンバの `gated` 構成ファイル `/etc/gated.conf` を総合的に管理します。IP 転送を可能にし、クラスタ・メンバを本格的なルータとして構成したい管理者にとってこの方法は有益です。

- このポリシーに従うメンバでは `/etc/rc.config` ファイルに次の内容を設定する。

```
GATED="yes"
CLUAMGR_ROUTE_ARGS="nogated"
ROUTER="yes" # if this member will be a full-fledged router
```

- 必要であれば、静的ルートをそのメンバの `/etc/routes` に設定する。

このシナリオでは、`aliasd` は `gated` を使用してホスト・ルートを伝播することはできず、`gated` に、クラスタ・ノードがクラスタから外に向かうパスの習得の手助けをさせることはできません。`gated` デーモンは自身のルーティング・テーブルを保持しています。

- 静的ルーティング (1 つ以上のクラスタ・メンバがルーティング・デーモンを実行しない)。

- 静的ルーティングを使う各メンバでは `/etc/rc.config` ファイルに次の内容を設定する。

```
GATED="no"
CLUAMGR_ROUTE_ARGS="nogated"
```

```
ROUTED="no"
ROUTED_FLAGS=""
```

- メンバの `/etc/routes` ファイルに静的ルートを定義する。

この構成は、ルーティング・デーモンを実行したくないサイトで便利です。これは、ローカルの管理ポリシーで RIP が禁止されていたり、サブネット内で複数の NIC を実行したい場合 (gated はこれをサポートしない) にあてはまります。少なくとも 1 つの省略時のルートを設定する必要があります。aliasd はルーティング・テーブルを制御し、NIC に障害が発生した場合に備えて、この省略時のルートを再構成することができます。これができない場合、ノードは孤立してしまいます。

### 3.15 クラスタ別名と NFS

クラスタを NFS サーバとして構成すると、NFS クライアントの出す要求は、省略時のクラスタ別名か、または `/etc/exports.aliases` の中に指定されている別名のいずれかに送信しなければなりません。クラスタ・メンバを個別に指定して NFS のマウント要求を送ると、その要求は拒否されます。

出荷時には、NFS クライアントの使える別名が省略時のクラスタ別名のみになるよう構成されていますが、独自のクラスタ別名も追加できます。`/etc/exports.aliases` ファイルにクラスタ別名の名前を追加すると、その別名のメンバが NFS 要求を受け付けるようになります。この機能は、クラスタ内の一部のメンバが、エクスポートされたファイル・システムのあるストレージに直接接続していない場合に便利です。そのような場合、直接接続しているシステムのみをメンバとする別名を作成すれば、NFS の要求のサービスに必要な内部ホップの数を削減できます。

『クラスタ概要』で説明されているように、NFS 要求を受け付けて処理する別名のメンバは、エクスポートされているファイル・システムのあるストレージに直接接続していなければなりません。また、その別名のメンバでない他のクラスタ・メンバがこのストレージに直接接続している場合、エクスポートされているファイル・システムの要求をそのシステムが受け付けて処理しないようにする必要があります。このファイル・システムの要求を受け付けて処理できるのは、そのファイル・システムのアクセスに使用する別名のメンバに限られます。この制限に合わせる方法はいくつかありますが、そのひとつは、`cfgmgr` を使用してファイル・システムを別名のメンバに手動で再配置するという方法です。また、ブート時に起動するスクリプトを作成して、どのメンバがファイル・システムを受け付けて処理



するかを自動的に調べ、そのファイル・システムを必要に応じてその別名メンバへ再配置する、という方法もあります。

『クラスタ概要』の中に、NFS とクラスタ別名サブシステムが NFS に関連する TCP および UDP のトラフィックを処理する方法が説明されているので、NFS サーバとして使用する別名を追加する前に、その説明も参照してください。また、`exports.aliases(4)` と `/etc/exports.aliases` ファイルの先頭にあるコメントも参照してください。

### 3.16 クラスタ別名と CAA (Cluster Application Availability)

ここでは、クラスタ別名サブシステムと CAA (Cluster Application Availability) サブシステムの違いについて簡単に説明します。

2 つのサブシステムには関係がまったくなく、相互に独立しています。CAA サブシステムは、アプリケーションの起動、リソースのモニタ、およびフェイルオーバーを扱う、アプリケーション制御ツールです。一方、クラスタ別名サブシステムは、クラスタ別名に宛てられた接続要求およびパケットをルーティングするためのルーティング・ツールです。これらのツールは相互に機能を補い合っています。具体的には、次に説明するように、CAA がアプリケーションの動作場所を決定し、クラスタ別名サブシステムがその場所へ到達する方法を決定します。

- CAA サブシステムは、一度に 1 つのクラスタ・メンバ上で動作するアプリケーションを対象に機能するよう設計されています。CAA サブシステムを使用すれば、必要なリソースをアプリケーションに関連付けておくことで、リソースを使用可能にしてからそのアプリケーションを起動するようにできます。また、アプリケーションを別のクラスタ・メンバ上にフェイルオーバーし、そのメンバ上でそのアプリケーションを自動的に再起動させることもできます。
- クラスタ別名サブシステムは受信した接続要求およびパケットをクラスタ内の複数のメンバに分配できるので、アプリケーションをクラスタ内の複数のメンバで動作させる場合に非常に有用です。このサブシステムは、クラスタ別名へ至るルートを公開し、その別名のメンバに接続要求とパケットを送信します。

混乱しやすいのが、シングル・インスタンス・アプリケーションという用語です。CAA サブシステムでは、この用語は、一度に 1 つのクラスタ・メンバ上でのみ動作するアプリケーションを意味します。しかし、クラスタ別名サ

ブシステムでは、アプリケーションが `in_single` と指定されていると、そのアプリケーションに対応したポート上で受信待ちしているクラスタ別名のメンバの数に関係なく、そのアプリケーションの1つのインスタンスにのみ接続要求とパケットが送信されます。つまり、クラスタ別名サブシステムは、アプリケーションがクラスタのすべてのメンバ上で動作しているか1つのメンバ上で動作しているかに関係なく、ポート上で受信待機中のクラスタ別名のメンバから任意に1つのメンバを選択して、そのメンバにすべての要求を送信します。そのメンバが応答できなくなると、クラスタ別名サブシステムは残りのメンバの1つに要求を送信します。

サービスを `in_single` にするか `in_multi` にするかは、`/etc/clua_services` で指定できます。`/etc/clua_services` に登録して CAA サブシステムに制御させるサービスに対しては、通常、`in_single` を指定する必要があります。ただし、そのサービスに対して `in_multi` を指定しても、次の理由から、そのサービスは正常に動作します。

- CAA サブシステムの制御するアプリケーションは同時にはただ1つのクラスタ・メンバ上でしか動作しない。したがって、ポート上で動作状態になるリスナは1つのみである。
- 要求またはパケットが到着したとき、クラスタ別名サブシステムがクラスタ別名のメンバをすべてチェックするが、リスナになっているメンバが1つだけなので、すべての要求とパケットがそのメンバに送信される。
- リスナが応答できなくなると、クラスタ別名サブシステムは別のリスナを見つけられないので、CAA デーモンが別のメンバ上でアプリケーションを起動するまで、すべてのパケットを廃棄するか、エラーを返す。別のメンバがリスナになると、クラスタ別名サブシステムは、その新しいポートにすべてのパケットを送信する。

省略時のクラスタ別名にはクラスタのすべてのメンバが属しています。これとは別に、クラスタの中から一部のメンバを集めて、任意のクラスタ別名を作成することができます。また、アプリケーションを起動または再起動するときに、CAA サブシステムの使用するメンバを制限することもできます (`favored` または `restricted` 配置ポリシ)。

クラスタ別名を作成した後、ユーザにこのクラスタ別名経由で CAA 制御下のアプリケーションへアクセスさせる場合は、アプリケーションの CAA 配置ポリシで選択されるメンバとそのクラスタ別名のメンバとが一致していることを確認する必要があります。一致していないと、そのクラスタ別

名に属さないメンバ上でアプリケーションが起動されることがあります。この状況では、アプリケーションがクラスタ・メンバ上で実行されているにもかかわらず、クラスタ別名サブシステムはそのクラスタ・メンバへパケットを送信できません。

クラスタ別名およびサービス属性と CAA サブシステムとがどのように関わり合うかを、次の 4 つの例を使って示します。

#### シナリオ 1

この例では、アプリケーションは CAA の制御下にありません。クラスタ別名サブシステムが、どのクラスタ別名にどのクラスタ・メンバが参加しているかを認識しているものとします。今、クライアントから、ターゲットのホスト名としてクラスタ別名を指定したサービス要求が出されたとします。クラスタ別名サブシステムは、次の手順に従って、クラスタ別名の 1 つのメンバに要求を送信します。

- 要求されたサービスが `clua_services` に登録されている場合は、そこに指定されているポート属性の値を調べる。たとえば `in_multi` と `in_single` のどちらか、または `in_noalias` と `in_nolocal` のどちらかなど。この例では、サービスが `in_multi` として指定されているものとする。
- 各メンバが別名に割り当てた選択優先順位 (`selp`) を調べる。
- 各メンバが別名に割り当てた選択重み (`selw`) を調べる。別名サブシステムは調べた `selp` および `selw` の値を用いて、どの別名メンバにパケットと接続要求を受信する資格があるかを決定する。
- その資格のあるメンバが、アプリケーションに対応したポート上で受信待ちをしているかどうかを調べる。
  - 資格のあるメンバが受信を待っていれば、そのメンバに接続要求またはパケットを送信する。
  - 資格のあるメンバが受信を待っていなければ、同じクラスタ別名内で `selp` と `selw` の条件を満たす次の別名メンバを調べる。

#### シナリオ 2

今度は、アプリケーションが CAA サブシステムに制御されているが、さらに状況を複雑にするために、`clua_services` 内でアプリケーションが

`in_multi` サービスとして誤って指定されているものとします。この場合は次のようになります。

- クラスタ別名サブシステムが接続要求またはパケットを受信する。
- 資格のあるクラスタ別名のメンバのうち、受信を待っているリスナは 1 つだけである (CAA が 1 つのクラスタ・メンバ上でだけアプリケーションを実行させているため)。
- こうした状況で、クラスタ別名サブシステムは接続要求またはパケットの送信先が 1 つのメンバのみであると判断し、CAA がアプリケーションを実行しているメンバにその要求またはパケットを送信する (実質的には `in_multi` が無視される)。

### シナリオ 3

このシナリオでは、アプリケーションが CAA サブシステムの制御下になく、複数のクラスタ・メンバ上で動作しているものとします。また、このアプリケーションのすべてのインスタンスは、同じ既知のポートにバインドされ、そのポート上で受信を待っているものとします。ただし、このサービスの `clua_services` 内のエントリは `in_multi` として指定されていないものとします。この場合、クラスタ別名サブシステムはポートを `in_single` とみなします。この場合は、次のようになります。

- クラスタ別名サブシステムが接続要求またはパケットを受信する。
- ポートは `in_single` になっている。
- クラスタ別名サブシステムは、接続要求またはパケットを受信するために、資格のある別名メンバを任意に 1 つ選択する。
- クラスタ別名サブシステムは、そのメンバのダウン、アプリケーションのクラッシュ、またはその他の理由でこのメンバ上に動作状態のリスナがなくなるまで、そのメンバにのみ接続要求またはパケットを送信する。

### シナリオ 4 (推奨しない)

最後に、CAA サブシステムとクラスタ別名サブシステムが適切に補い合っ  
て働かないシナリオを示します。

- クラスタ・メンバ A および B がクラスタ別名に参加している。
- CAA がアプリケーションを制御しているが、そのアプリケーションの配置ポリシーは `restricted` になっており、クラスタ・ノード A または

ノード C のいずれかで実行できる。ユーザは、クラスタ別名でサービスにアクセスする。

- アプリケーションはノード A で実行されている。ノード A に障害が発生すると、CAA はアプリケーションをノード C に再配置する。
- しかし、アプリケーションがノード C で実行されていても、ユーザは別名を通してアプリケーションにアクセスすることができない。



---

## クラスタのメンバシップ管理

クラスタ化されたシステムでは、ディスクやファイルのような、さまざまなデータおよびシステム・リソースを共有します。リソースの一貫性の維持にはメンバの連携が必要であり、そのためにはメンバシップに関する明確な基準が必要です。この基準を満たさないシステムに対しては、クラスタへの参加を許可しないようにします。

この章では、次の項目について説明します。

- 接続マネージャの機能の概要 (4.1 節)
- クォーラム、ポート、およびクラスタのメンバシップ (4.2 節)
- 接続マネージャによるクォーラムの計算方法 (4.3 節)
- 3 メンバ構成のクラスタの使用例 (4.4 節)
- クォーラム・ディスクを使用するタイミングと方法 (4.5 節)
- `clu_quorum` コマンドによるクラスタのクォーラム情報の表示方法 (4.6 節)
- さまざまなポート設定による接続マネージャの動作例 (4.7 節)
- 接続マネージャのモニタ方法 (4.8 節)
- 接続マネージャのパニック (4.9 節)
- 不適切な期待ポートおよびノード・ポート設定に関する問題解決 (4.10 節)

### 4.1 接続マネージャの概要

接続マネージャは、クラスタのメンバが相互に通信できるかどうかを監視し、クラスタのメンバシップに関する規則を強制する分散型のカーネル構成要素です。接続マネージャは次のような機能を実行します。

- クラスタの形成、クラスタへのメンバの追加、およびクラスタからのメンバの削除を行う。

- クラスタのどのメンバが稼働中であるか追跡する。
- クラスタのすべてのメンバ上でクラスタのメンバシップ・リストの一貫性を維持する。
- EVM (イベント・マネージャ) のイベントを使ってメンバシップの変更を随時通知する。
- クラスタ分断を検出して対処する。

接続マネージャのインスタンスは、クラスタの各メンバ上で動作します。これらのインスタンスは、クラスタのメンバシップ・リストのような情報を共有することによって、メンバ間の相互接続を維持します。接続マネージャは、三相コミット・プロトコルを使って、すべてのメンバから見えるクラスタのビューの一貫性を維持します。

## 4.2 クォーラムとボートの概要

接続マネージャは、投票メカニズムを使用することによって、通信障害の発生時でもデータの一貫性を保証します。このメカニズムでは、ボート (投票数) が過半数に達したときに限り、クラスタ内でのプロセス動作と入出力操作を許可します。このようにクラスタ内に過半数のボートが存在する状態を、クラスタがクォーラム (定足数) を維持しているといいます。

接続マネージャはクォーラムを計算し、その結果に基づいて、クラスタ・メンバとしてのシステムの追加および残留を許可します。このような投票メカニズムは、期待ボート、現在のボート、ノード・ボート、クォーラム・ディスク・ボートなど多くの要素に依存します。ここでは、これらの要素の概念について説明します。

### 4.2.1 システムをクラスタのメンバにする方法

クラスタのメンバシップを管理できるのは接続マネージャだけです。コマンド `clu_create` または `clu_add_member` を使ってクラスタのメンバになるようにノードを構成しても、そのノードはすぐにはクラスタのメンバになりません。メンバになるのは、クラスタ化カーネルによってリブートされ、接続マネージャによってクラスタの形成またはクラスタへの参加を許可されたからです。クラスタのメンバと、クラスタのメンバになるように構成されたノードとの違いは、クォーラムとボートについて考えるとき常に重要です。



ノードがクラスタを形成したか、クラスタに参加した後、そのノードは接続マネージャによって無期限に (ただし `clu_delete_member` を使ってクラスタから削除されるまで) クラスタのメンバとみなされます。希に、クラスタ内でのハードウェアの故障または切断などによる通信の切断によって、既存クラスタが複数のクラスタに分断されることがあります。この状況をクラスタ分断といいます。クラスタ分断が発生すると、ノードは所属先のクラスタを特定できなくなる可能性があります。ただし 4.3 節で説明するように、接続マネージャは、これらのクラスタの 1 つしか稼働させません。

## 4.2.2 期待ポート

期待ポートは、構成済みのポートがすべて使用可能なときのポート数で、接続マネージャが使用します。つまり、期待ポートはクラスタ内で構成済みのノード・ポート (4.2.4 項) とクォーラム・ディスク・ポート (4.2.5 項) との合計ということになります (クォーラム・ディスク・ポートはクォーラム・ディスクが構成されている場合のみ加算します)。各メンバは、それぞれ自身の情報をクラスタに通知します。すべてのメンバの期待ポートの数は、一致する必要があります。

接続マネージャは、クラスタのブート・メンバのノードの期待ポートを参照して、クラスタ全体の期待ポートを決定します。このクラスタ全体の期待ポートのことをクラスタの期待ポートといいます。接続マネージャは、クラスタの期待ポートの値を使って、クラスタがクォーラムを維持するために必要なポート数 (4.3 節を参照) を決定します。

クラスタの期待ポートの現在の値を表示するには、コマンド `clu_quorum` または `clu_get_info -full` を使用します。

新規の投票メンバまたはクォーラム・ディスクがクラスタ内で構成されると、スクリプト `clu_create` および `clu_add_member` が各メンバの期待ポートを自動的に調節します。クラスタからメンバが削除されると、`clu_delete_member` コマンドが期待ポートを自動的に減らします。同様に、クォーラム・ディスクの追加または削除が行われたり、メンバに対してノード・ポートの割り当てや削除が行われたりすると、`clu_quorum` コマンドが各メンバの期待ポートを調節します。これらのコマンドによって確実に、クラスタ内のすべてのメンバの期待ポートの値が一致し、さらにそのポートの値は、すべてのノード・ポートとクォーラム・ディスク・ポート (クォーラム・ディスクが構成されている場合) との合計に一致します。

メンバの期待ボートは、そのメンバ固有の `etc/sysconfigtab` ファイル内にある `clubase` サブシステムのカーネル属性 `cluster_expected_votes` によって初期化されます。メンバの期待ボートを表示するには、`clu_quorum` コマンドを使用します。

メンバの期待ボートを変更するには、`clu_quorum -e` コマンドを使用しなければなりません。これによって確実に、すべてのメンバの期待ボートが適切かつ同一になります。カーネル属性 `cluster_expected_votes` を直接変更することはできません。

### 4.2.3 現在のボート

期待ボートの数がクラスタ内で構成されたボートの数と一致する場合、現在のボートは、現在オンラインのメンバと構成済みクォーラム・ディスクが投じたボートの合計です。つまり、現在のボートはクラスタ内で実際に見えるボートの数です。

### 4.2.4 ノード・ボート

ノード・ボートは、クラスタの1つのメンバがクォーラムに投じるボートの定数です。メンバには1または0(ゼロ) ノード・ボートを割り当てることができます。1 ボートを持つメンバは、クラスタの投票メンバとみなされます。0(ゼロ) ボートを持つメンバは非投票メンバとみなされます。

---

#### 注意

---

シングルユーザ・モードへの移行はメンバの投票状態に影響しません。投票メンバは、シャットダウンされてシングルユーザ・モードでリブートされても、引き続き投票メンバです。つまり接続マネージャは、シャットダウンされてシングルユーザ・モードでリブートされたメンバを、引き続きクラスタのメンバとみなします。

---

投票メンバはクラスタを形成できます。非投票メンバは既存クラスタへの参加しかできません。

通常は、クラスタの構成中にメンバにボートを割り当てます。たとえば、`clu_create` による初期メンバの作成時か、`clu_add_member` による新規メンバの追加時に、ボートの割り当てを行います。省略時の設定により、`clu_create` の実行時に、最初のメンバに1 ボートが割り当てられま

す。clu\_add\_member の実行時には、省略時の設定により、期待ポートが 1 の場合は新規メンバに 0 ポートが割り当てられ、期待ポートが 2 以上の場合は新規メンバに 1 ポートが割り当てられます (clu\_create および clu\_add\_member は、クラスタ内で新規ポートが構成されると、期待ポートを自動的に増やします)。clu\_quorum -m コマンドを使用すれば、クラスタのメンバに割り当てられたノード・ポートの数を後で調節できます。

メンバのノード・ポートは、そのメンバ固有の etc/sysconfigtab ファイル内にある clubase サブシステムのカーネル属性 cluster\_node\_votes によって初期化されます。メンバのノード・ポートを表示するには、コマンド clu\_quorum または clu\_get\_info -full を使用します。詳細については、4.6 節を参照してください。

メンバのノード・ポートを変更するには、clu\_quorum コマンドを使用しなければなりません。カーネル属性 cluster\_node\_votes を直接変更することはできません。

#### 4.2.5 クォーラム・ディスク・ポート

クラスタ構成では、4.5 節の説明に従ってクォーラム・ディスクを構成することにより、クラスタの可用性を高めることができます。クォーラム・ディスク・ポートは、クォーラム・ディスクがクォーラムに投じるポートの定数です。クォーラム・ディスクには 1 または 0 ポートを割り当てることができます。

通常、clu\_create によるクラスタの作成時に、クォーラム・ディスクを構成して、ポートを割り当てます。クラスタの作成時にクォーラム・ディスクを定義する場合、省略時の設定により、クォーラム・ディスクには 1 ポートが割り当てられます。

クォーラム・ディスク・ポートは、各メンバの etc/sysconfigtab ファイル内にある clubase サブシステムのカーネル属性 cluster\_qdisk\_votes によって初期化されます。クォーラム・ディスク・ポートを表示するには、clu\_quorum コマンドまたは clu\_get\_info コマンドを使用します。

クォーラム・ディスク・ポートを変更するには、clu\_quorum コマンドを使用しなければなりません。カーネル属性 cluster\_qdisk\_votes を直接変更することはできません。

クォーラム・ディスクを構成すると、そのポートはクラスタの形成で特殊な役割を果たします。これは、接続マネージャが強制する次のような規則があるためです。

- ブート・ノードは、クォーラムを満たすまでクラスタを形成できない。
- ブート・ノードは、クォーラム・ディスクの所有権とそのポートを要求するために、クラスタのメンバでなければならない。

つまり、ブート・ノードがクォーラムを満たそうとしてクォーラム・ディスク・ポートを要求すると、これらの規則によってジレンマに陥ります。ブート・ノードはクラスタを形成できません。

このジレンマから抜け出すために、接続マネージャは、ブート・ノードがクォーラムにクォーラム・ディスク・ポートを一時的に投じることができるようにします。これによって、ブート・ノードはクォーラムを満たして、クラスタを形成できます。ブート・ノードは、クラスタが形成されると、クォーラム・ディスクの所有権を要求します。この時点で、そのクォーラム・ディスクのポートは仮のポートから正式なポートになります。

### 4.3 クラスタのクォーラム・ポートの計算の概要

接続マネージャは、クォーラム・アルゴリズムを使って、あるノードがクラスタに参加し、クラスタ全体のリソースに安全にアクセスして、有用な作業を実行できるかどうかを調べます。このアルゴリズムは動的です。つまり、クラスタのクォーラム・ポートの計算はクラスタのイベントによって開始され、その計算結果はクラスタの存在期間に渡って変化します。

クォーラム・アルゴリズムは次のような仕組みになっています。

1. 接続マネージャは、クラスタのクォーラム・ポートの計算に使用するクラスタ・メンバの集合を選択します。この集合には相互通信可能なメンバのみが含まれます。構成されているがブートされていないノード、ダウンしているメンバ、ハードウェアの障害 (クラスタのインターコネクト・ケーブルの切断や Memory Channel アダプタの故障) によって到達不能になったメンバなどは含まれません。
2. クラスタの形成時、ノードがブートしてクラスタに参加するたびに、接続マネージャは次の値から最大値を選択し、その値を使ってクラスタの期待ポートを計算します。

- 手順 1 で選択したメンバの集合から取得したメンバの期待ポートのうちの最大値
- 手順 1 で選択したメンバの集合から取得したノード・ポートとクォーラム・ディスク・ポート (クォーラム・ディスクが構成されている場合) との合計
- 前回のクラスタの期待ポート値

たとえば、クォーラム・ディスクがない 3 メンバ構成のクラスタがあるとして、すべてのメンバは稼働中であり、相互に接続されています。各メンバのノード・ポートは 1 に設定され、期待ポートは 3 に設定されています。クラスタの期待ポートは現在 3 です。

その後、4 番目の投票メンバがクラスタに参加するとします。この新しいメンバがブートしてクラスタに参加した場合、接続マネージャはクラスタの期待ポートを計算して、4 に設定します。この値はクラスタ内のノード・ポートの合計です。

クラスタの期待ポートの現在の値を表示するには、コマンド `clu_quorum` または `clu_get_info -full` を使用します。

3. 接続マネージャは、クラスタの期待ポートを再計算するたびに (または `clu_quorum -e` コマンドでクラスタの期待ポートをリセットするたびに)、クォーラム・ポートを計算します。

クォーラム・ポートは、クラスタの期待ポートの値に基づいて動的に計算されるクラスタ単位の値です。この値によって、ノードに対し、クラスタの形成、クラスタへの参加、またはクラスタへの残留を許可するかどうかが決まります。接続マネージャは、次の数式を使ってクラスタのクォーラム・ポートを計算します。

クォーラム・ポート = (クラスタの期待ポート + 2) / 2 (小数点以下切り捨て)

たとえば、前述した手順の 3 メンバ構成のクラスタの場合、クラスタの期待ポートが 3 なので、クォーラム・ポートは、 $((3+2) / 2)$  で計算して小数点以下を切り捨てた結果、2 になります。4 番目のメンバが正常に追加された場合、クォーラム・ポートは、 $((4+2) / 2)$  で計算して小数点以下を切り捨てた結果、3 になります。

#### 注意

期待ポートは (結果的にクォーラム・ポートも) クラスタ構成に基づきます。どのノードが稼働しているかダウンしてい

るかには関係ありません。メンバがシャットダウンされたか、他の何らかの理由でダウンしたとき、接続マネージャはクォーラム・ポートの値を減らしません。メンバが削除されたか、`clu_quorum -e` コマンドが実行されたときだけ、接続マネージャは稼働中のクラスタのクォーラム・ポート値を減らします。

4. クラスタのメンバが参照可能なポート数の変化を検出すると (ノードがクラスタに参加したか、既存メンバがクラスタから削除されたか、通信エラーが報告されたことが原因)、そのメンバは常に現在のポートとクォーラム・ポートとを比較します。

この後、メンバは次の条件に基づいたアクションをとります。

- 現在のポートの値がクォーラム・ポートの値以上の場合、そのメンバは動作を続行するか、動作を再開します (中断されていた場合)。
- 現在のポートの値がクォーラム・ポートの値未満の場合、そのメンバはすべてのプロセス動作、クラスタ全体のストレージへの入出力操作、およびクラスタの外部ネットワークに対する操作を中断します。十分な数のポートが追加され (つまり十分な数のメンバがクラスタに参加するか、通信上の問題が解決され)、現在のポートの値がクォーラム・ポートの値以上になると、中断されていた動作や操作は再開されます。

イベント・メッセージにはクォーラムが失われたことがクラスタ単位のイベントとして書き込まれることがありますが、現在のポートとクォーラム・ポートとの比較はメンバ単位で行われます。クラスタのいずれかのメンバがクォーラムを失うと、そのメンバの入出力操作はすべて中断され、クラスタ・インターコネクト以外のすべてのネットワーク・インタフェースが停止状態になります。クラスタ全体のリソースへのアクセスを必要とするコマンドはすべて、そのメンバ上では機能しなくなります。その結果、そのメンバはハングアップしているように見えることがあります。

そのメンバがどのようにクォーラムを失ったかにもよりますが、クォーラムを失ったメンバがクォーラムを満たせるだけのポートを別のメンバに割り当て、そのメンバをブートし、クォーラムを回復させることによって、この状況を解決できる場合があります。ただし、クラスタのすべてのメンバがクォーラムを失った場合は、それらのメンバがクォーラムを満たせるだけの

ポートを新規メンバに割り当て、その新規メンバをブートするか、クラスタ全体をリブートするか、あるいは 4.10 節の手順を実行するしかありません。

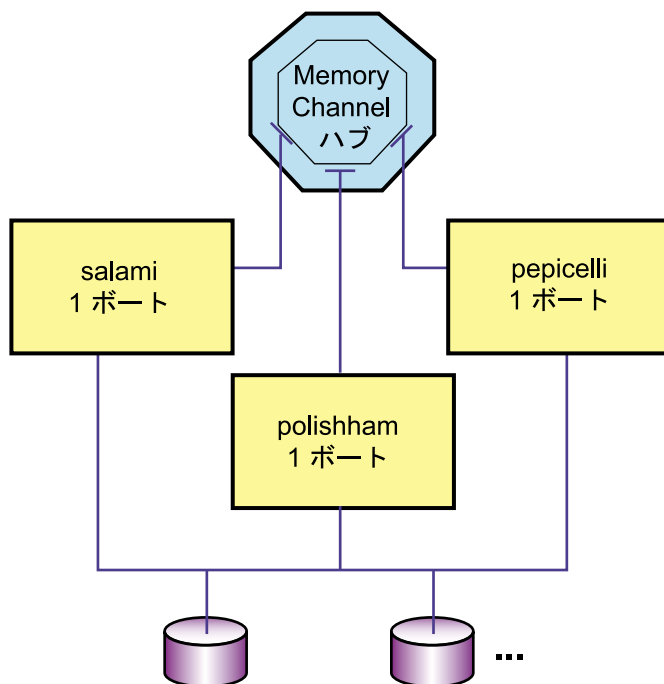
## 4.4 接続マネージャの動作例

接続マネージャがクラスタの形成を許可するのは、ブートされたノードによってポートが投じられ(クォーラム・ディスクが構成されている場合はクォーラム・ディスク・ポートも投じられ)、それらのポートの合計がクラスタのクォーラムに達したときです。

図 4-1 に示すような 3 メンバ構成のクラスタ `deli` があるとします。すべてのメンバが稼働中であり使用可能なとき、各メンバは 1 ノード・ポートを投じます。その結果、クラスタの期待ポートは 3 になり、クォーラム・ポートの計算結果は 2 になります。つまり、クラスタ `deli` は、1 つのメンバに障害が発生しても稼働し続けることができます。

図 4-1: 3 メンバ構成のクラスタ deli

deli クラスタ  
期待ポート = 3  
クォーラム・ディスクなし  
クォーラム = 2 [小数点以下切り捨て((3+2)/2)]



ZK-1567U-AI

ノード salami が最初にブートされると、コンソールには次のメッセージが表示されます。

```
CNX MGR: Node salami id 3 incarn 0xbde0f attempting to form or join cluster
deli
CNX MGR: insufficient votes to form cluster: have 1 need 2
CNX MGR: insufficient votes to form cluster: have 1 need 2
.
.
.
```

ノード polishham がブートされると、このノードのポートと salami のポートとの合計がクォーラム (2) に達し、クラスタの形成が許可されます。その結果、次の CNX MGR メッセージが表示されます。

```
.
.
.
CNX MGR: Cluster deli incarnation 0x1921b has been formed
Founding node id is 2 csid is 0x10001
```

#### 4-10 クラスタのメンバシップ管理



```
CNX MGR: membership configuration index: 1 (1 additions, 0 removals)
CNX MGR: quorum (re)gained, (re)starting cluster operations.
CNX MGR: Node salami 3 incarn 0xbde0f csid 0x10002 has been added to the
cluster
CNX MGR: Node polishham 2 incarn 0x15141 csid 0x10001 has been added to the
cluster
```

形成されたクラスタにノード pepicelli が参加すると、ノード pepicelli のブート・ログには、前記のメッセージと似たメッセージが表示されますが、クラスタ形成のメッセージの代わりに、次のメッセージが表示されます。

```
CNX MGR: Join operation complete
CNX MGR: membership configuration index: 2 (2 additions, 0 removals)
CNX MGR: Node pepicelli 1 incarn 0x26510f csid 0x10003 has been added to the
cluster
```

もちろん、ノード pepicelli は、他の 2 つのノードと同時にブートされるときは、それら 2 つのノードと同様、クラスタの形成に参加し、その場合は、クラスタ形成のメッセージが表示されます。

その後、pepicelli がシャットダウンされた場合は、図 4-2 に示すように、メンバ salami および polishham はそれぞれ自身の現在の期待ポート (2) と、クォーラム・ポート (2) とを比較します。その結果、現在のポートがクォーラム・ポートと等しいので、これら 2 つのメンバはクラスタに残留することができ、ノード pepicelli のシャットダウン後も稼働し続けることができます。次に、1 つのメンバを失った場合の一連の動作を記録したログ・メッセージを示します。

```
memory channel - removing node 2
rm_remove_node: removal took 0x0 ticks
ccomsub: Successfully reconfigured for member 2 down
ics_RM_membership_change: Node 3 in RM slot 2 has gone down
CNX MGR: communication error detected for node 3
CNX MGR: delay 1 secs 0 usecs
.
.
.
CNX MGR: Reconfig operation complete
CNX MGR: membership configuration index: 13 (2 additions, 1 removals)
CNX MGR: Node pepicelli 3 incarn 0x21d60 csid 0x10001 has been removed
from the cluster
```

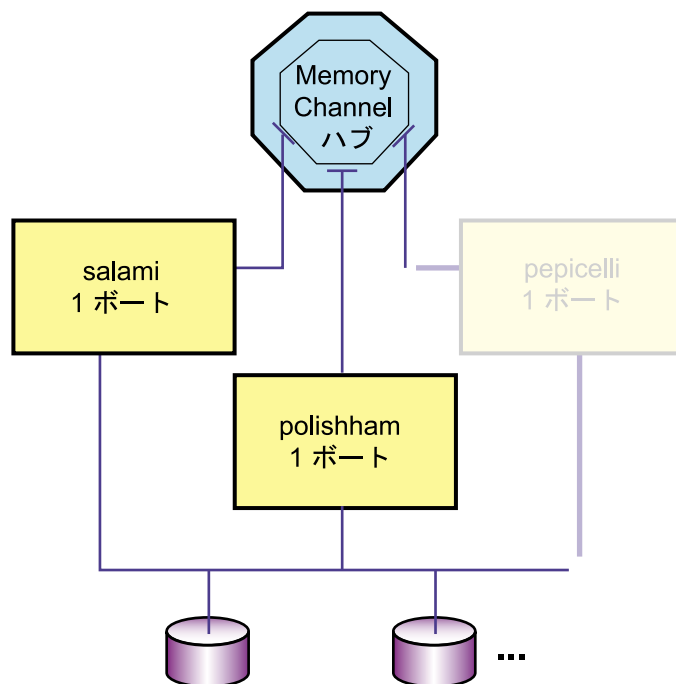
図 4-2: 3 メンバ構成のクラスタ deli が 1 つのメンバを失った場合

**deli** クラスタ

期待ポート = 3

クォーラム・ディスクなし

クォーラム = 2 [小数点以下切り捨て  $((3+2)/2)$ ]



ZK-1568U-AI

ただし、このクラスタは、さらに別のメンバが失われると稼働できなくなります。メンバ polishham がシャットダウンされると、図 4-3 の図と 4.5 節の説明のような状況になります。クラスタ deli はクォーラムを失い、動作を中断し、次のメッセージが表示されます。

```
memory channel - removing node 4
rm_remove_node: removal took 0x0 ticks
ccomsub: Successfully reconfigured for member 4 down
ics_RM_membership_change: Node 2 in RM slot 4 has gone down
CNX MGR: communication error detected for node 2
CNX MGR: delay 1 secs 0 usecs
CNX MGR: quorum lost, suspending cluster operations.
.
.
```

## 4.5 クォーラム・ディスクの使用

2 メンバ構成のクラスタで、各メンバのノード・ポートが 1 であり、期待ポートが 2 である場合、1 つのメンバが失われると、クラスタはクォーラムを失い、すべてのアプリケーションが中断されます。この種のクラスタ構成の可用性は高くありません。

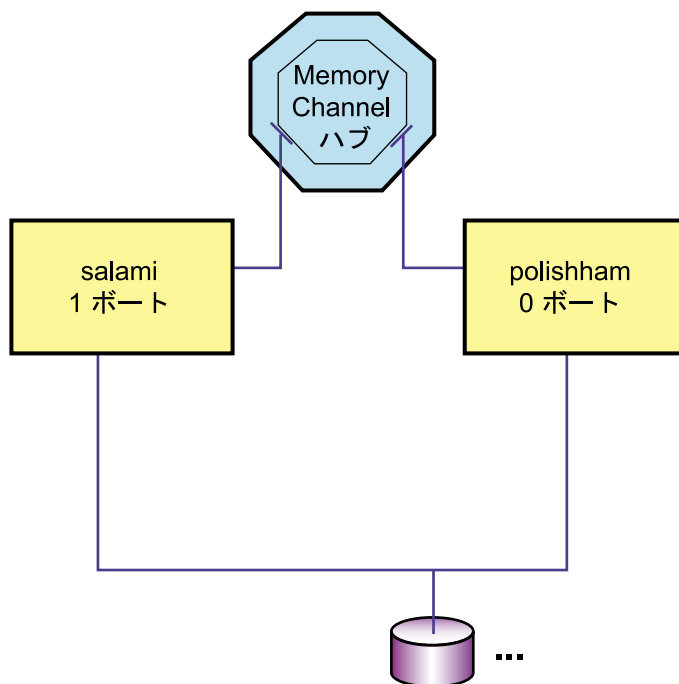
2 メンバ構成のクラスタで、1 番目のメンバのノード・ポートが 1 であり、2 番目のメンバのノード・ポートが 0 であり、期待ポートが 1 の場合、前の構成よりも可用性は高まりますが、あまり良い構成とは言えません。このクラスタは、2 番目のメンバ (非投票メンバ) を失っても稼働し続けますが、1 番目のメンバ (投票メンバ) を失うと稼働できなくなります。

このようなクラスタ構成の可用性を高めるには、共用バス上のディスクをクォーラム・ディスクとして指定します。クォーラム・ディスクは、仮想クラスタ・メンバとして動作し、その目的は期待ポートに 1 ポートを加算することです。2 メンバ構成のクラスタでクォーラム・ディスクを構成すると、クォーラム・ディスクまたは 1 つのメンバに障害が発生しても、クラスタは稼働し続けることができます。

たとえば、図 4-3 に示すように、クォーラム・ディスクがない 2 メンバ構成のクラスタ `deli` があるとします。

図 4-3: 2 メンバ構成のクラスタ **deli** にクォーラム・ディスクがない場合

**deli** クラスタ  
 期待ポート = 1  
 クォーラム・ディスクなし  
 クォーラム = 1 [小数点以下切り捨て((1+2)/2)]



ZK-1569U-AI

メンバ **salami** は 1 ノード・ポートを投じ、メンバ **polishham** は 0 ポートを投じます。したがって、クラスタの期待ポートは 1 です。接続マネージャは次のようにクォーラム・ポートを計算します。

$$\begin{aligned} \text{クォーラム・ポート} &= (\text{クラスタの期待ポート} + 2) / 2 && \text{(小数点以下切り捨て)} \\ &= (1 + 2) / 2 \\ &= 1 \end{aligned}$$

メンバ **salami** の障害またはシャットダウンにより、メンバ **polishham** はクォーラムを失い、クラスタの動作は中断されます。

しかし、クラスタ内にクォーラム・ディスク (クラスタの期待ポートに 1 ポートを加算) を構成し、メンバ **polishham** にも 1 ポートを割り当てた場合、期待ポートは 3 になり、クォーラム・ポートは 2 になります。この場合のクォーラム・ポートの計算は次のようになります。

$$\begin{aligned}
 \text{クォーラム・ポート} &= (\text{クラスタの期待ポート} + 2) / 2 && (\text{小数点以下切り捨て}) \\
 &= (3 + 2) / 2 \\
 &= 2
 \end{aligned}$$

このクラスタ構成では、いずれか一方のメンバまたはクォーラム・ディスクをクラスタから削除する場合でも、クラスタがクォーラムを失わないだけの数の現在のポートが残ります。このように、図 4-4 に示すクラスタは稼働し続けることができます。

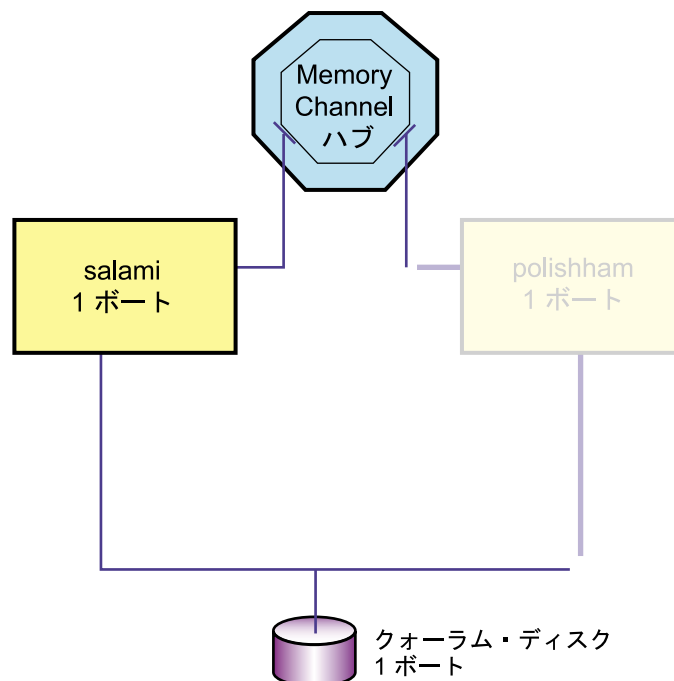
図 4-4: 2 メンバ 構成のクラスタ **deli** が 1 つのメンバを失ってもクォーラム・ディスクがあるために稼働し続ける場合

**deli** クラスタ

期待ポート = 3

1 ポートのクォーラム・ディスク

クォーラム = 2 [小数点以下切り捨て((3+2)/2)]



ZK-1575U-AI

`clu_create` ユーティリティを使用すれば、クラスタの作成時に、クォーラム・ディスクを指定し、そのディスクにポートを割り当てることができます。また、`clu_quorum` ユーティリティを使用すれば、クラスタの存在期間内のその他の任意のタイミングでも、クォーラム・ディスクを追加できま

す。たとえば、2メンバ構成のクラスタで `clu_delete_member` を実行した結果、クラスタの可用性が下がったときなどが挙げられます。

クォーラム・ディスクを構成するには、`clu_quorum -d add` コマンドを使用します。たとえば次のコマンドは、`/dev/disk/dsk11` をクォーラム・ディスクとして定義し、そのディスクに 1 ボートを割り当てます。

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

Info: Disk available but has no label: dsk11
      Initializing cnx partition on quorum disk : dsk11h

Successful quorum disk creation.
# clu_quorum

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

次に、クォーラム・ディスクの使用上の制限を示します。

- クラスタ内には 1 つのクォーラム・ディスクしか構成できません。
- クォーラム・ディスクは、クラスタのすべてのメンバが直接接続される共用バス上に置くことを推奨します。共用バス上にない場合、クォーラム・ディスクに直接接続されないメンバは、クォーラム・ディスクに直接接続されるメンバよりも先に、クォーラムを失う可能性があります。
- クォーラム・ディスクにはデータが格納されてはなりません。クォーラム・ディスクの初期化時、ディスク内の既存データは `clu_quorum` コマンドによって上書きされます。また、メンバに障害が発生すると、稼働中のクラスタからクォーラム・ディスクに格納されたデータ (またはファイル・システムのメタデータ) の一貫性は失われます。

メンバのブート・ディスク、およびクラスタ全体のルート (/) があるディスクは、クォーラム・ディスクとして使用できません。

- クォーラム・ディスクの容量は非常に小さくても問題ありません。クラスタ・サブシステム用に必要なのは 1 MB だけです。
- クォーラム・ディスクには 1 ボートを割り当てても、ボートを割り当てなくてもかまいません。通常は、クォーラム・ディスクにはボートを割り当てておきます。ただし、テスト構成または一時的な構成では、ボートを割り当てない場合もあります。たとえば、1 メンバ構成

のクラスタで 1 ボートを持つクォーラム・ディスクが別の障害ポイントとなる場合です。

- クォーラム・ディスク上で LSM (Logical Storage Manager) を使用することはできません。

概念としては、クラスタの障害のあるメンバ (ボート) とないメンバが同数である場合、クォーラム・ディスクで提供されるボートはタイブレーカとして働きます。タイブレーカのボートによって、クォーラムを維持してクラスタ操作が継続されます。この点についてクォーラム・ディスクのボートは、ボートと違いはありません。たとえば、2 メンバ・クラスタの 3 番目のボート・メンバとするか、4 メンバ・クラスタの 5 番目のボート・メンバとすることができます。これは、全共用ストレージへ直接接続されていない非ボート・メンバが数多くある、大規模なクラスタを計画するときに重要な考えです。

ファイル・サーバとして動作する 2 つの大規模なメンバを持つクラスタを考えてみてください。これらのメンバは、重要なクラスタのファイル・システムとアプリケーション・データベースへ直接接続されるため、クラスタの操作に対して重要であり、それぞれ 1 ボートが割り当てられます。このクラスタのその他のメンバはクライアント要求を処理し、サーバへ送信します。これらのメンバは、共用ストレージへ直接接続されていないため、クラスタ操作への重要性は低く、ボートは割り当てられません。ただし、このクラスタには 2 つしかボートがないため、タイブレーカのボートを設定するまでどちらのファイル・サーバ (メンバ) も動作を停止することができません。

この場合、タイブレーカのボートをどのように構成しようとするでしょうか。ボートを持つクォーラム・ディスクを設定するのは、あまりよい選択ではありません。この設定でのクォーラム・ディスクは 2 つのファイル・サーバのメンバのみに直接接続されています。クライアントの処理メンバは、結果的にクォーラム用のボートに数えることができません。クォーラム・ディスクまたは 1 つのファイル・サーバのメンバに障害が発生すると、クライアントの処理メンバはクォーラムを失い、サーバへのクライアント要求の送信を停止します。これは、たとえクォーラムを維持し続けても、サーバ・メンバの操作に影響を与えます。タイブレーカのボートをこの種の設定に使用するための最良の解決策は、クライアントの処理メンバの 1 つにボートを割り当てることです。クラスタ全体では、1 つのボートが失われても、操作を継続できます。

クォーラム・ディスクを追加し、クォーラムを維持するためにそのポートが必要になると、`clu_quorum` から次のようなメッセージが表示されます。

```
Adding the quorum disk could cause a temporary loss
of quorum until the disk becomes trusted.
Do you want to continue with this operation? [yes]:
```

通常は、この質問に "yes" で答えます。`clu_quorum` コマンドは 20 秒ほどでクォーラム・ディスクの信頼性を判断します。クォーラム・ディスクを信頼できるものにするためには、メンバがそのディスクに直接接続していること、メンバがディスクとの間で読み書きできること、そして、メンバが所有権を要求しているかまたは所有権を要求しているメンバと同じクラスタのメンバになっていることが必要です。

既存のクォーラム・ディスクのポートを調整しようとしたときにメンバがディスクを信頼できると判断していなければ (`cnx` サブシステムの `qdisk_trusted` 属性の値がゼロになっている)、`clu_quorum` コマンドから次のメッセージが表示されます。

```
The quorum disk does not currently appear to be trusted.
Adjusting the votes on the quorum disk could cause quorum loss.
Do you want to continue with this operation? [no]:
```

クォーラム・ディスクが現在信頼されていない場合、前述の条件を満たすために何らかの処置を施さない限り、信頼性は得られません。この場合は、質問に "no" で応答し、別の方法でクラスタへポートを追加してください。

#### 4.5.1 障害が発生したクォーラム・ディスクの交換

クラスタの稼働中にクォーラム・ディスクに障害が発生したが、クラスタがクォーラムを失わない場合は、次の手順でディスクを交換できます。

1. ディスクがクラスタから切断されていることを確認します。
2. `clu_quorum` コマンドを実行し、クォーラム・ディスク・ポートの実行時の値をメモしておきます。
3. `clu_quorum -f -d remove` コマンドを使って、クラスタからクォーラム・ディスクを削除します。
4. ディスクを交換します。各クラスタ・メンバ上で `hwmgr -scan scsi` コマンドを入力します。



---

## 注意

---

`hwmgr -scan scsi` コマンドは、すべてのクラスタ・メンバ上で実行する必要があります。

---

新規ディスクがすべてのメンバによって認識されるまで、しばらく待ちます。

5. `hwmgr -view devices -cluster` コマンドを使って、新規ディスクのデバイス・スペシャル・ファイルの名前 (dsk 名) を調べます。この名前は、障害が発生したクォーラム・ディスクの名前とは異なります。任意に、`dsfmgr -n` コマンドを使って、新規ディスクのデバイス・スペシャル・ファイルの名前を障害が発生したディスクの名前に変更することもできます。
6. `clu_quorum -f -d add` コマンドを使って、新規ディスクをクォーラム・ディスクとして構成します。新規ディスクには、手順 2 でメモしておいたポートと同数のポートが割り当てられていることを確認します。

クラスタの稼働中にクォーラム・ディスクに障害が発生して、クラスタがクォーラムを失い、クラスタの動作が中断された場合は、4.10.1 項の手順に従って、1 つのクラスタ・メンバを停止し、対話形式でリブートして、クラスタのクォーラムを回復しなければなりません。その後、前述の手順を実行できます。

## 4.6 clu\_quorum コマンドによるクラスタのクォーラム情報の表示

`clu_quorum` コマンドをオプションなしで実行するか、`-f` または `-v` (あるいはその両方) を付けて実行すると、クラスタの現在のクォーラム・ディスク、メンバのノード・ポート、およびクラスタの期待ポートに関する情報が表示されます。この情報は次の項目から構成されます。

- クラスタの共通クォーラム・データ。たとえば、構成済みのクォーラム・ディスクのデバイス名、およびクラスタ単位の `/etc/sysconfigtab.cluster` からのクォーラム・データが表示されます。
- 各メンバの動作中のカーネルおよび `/etc/sysconfigtab` ファイルからのメンバ固有のクォーラム・データと、メンバの状態 (UP また

は DOWN)。省略時の設定では、DOWN 状態のメンバに関するクォーラム・データは表示されません。ただし、DOWN 状態のメンバのブート・パーティションが `clu_quorum` コマンドの実行元のメンバにアクセス可能な限り、`-f` オプションを使って、DOWN 状態のメンバのクォーラム・データの値を表示できます。

`clu_quorum` による表示項目については、`clu_quorum(8)` を参照してください。

## 4.7 クラスタ・ボートの割り当て例

表 4-1 に、クラスタ・メンバ上の属性 `cluster_expected_votes` および `cluster_node_votes` のさまざまな設定によって、クラスタの形成にどのような影響が出るかを示します。さらに、これらの属性のどの設定の組み合わせによって、クラスタ全体が稼働不能になるか、またはクラスタの可用性が最大限に高まるかも示します。この表では、2 メンバ、3 メンバ、および 4 メンバ構成のクラスタを対象にします。

次に、この表内の見出し下の列の内容と表記について説明します。

- 「メンバの期待ポート」の列には、メンバの `/etc/sysconfigtab` ファイル内にある `clubase` スタンザの `cluster_expected_votes` 属性に設定したメンバの期待ポートを示します。
- 「M1」, 「M2」, 「M3」, および「M4」の列には、メンバに割り当てたポートを示します。
- 「クォーラム・ディスク」の列には、クォーラム・ディスク (構成されている場合) に割り当てたポートを示します。
- 「---」という表記は、そのノードがクラスタ内で構成されていないことを示します。

表 4-1: 2～4 メンバ構成のクラスタ内でのメンバの `cluster_expected_votes` 設定とポートの割り当てによる影響

メンバの期待ポート	M1	M2	M3	M4	クォーラム・ディスク	影響
1	1	0	---	---	0	クラスタを形成できるのは M1 が存在するときだけである。クラスタは M2 の障害発生時は稼働し続けるが、M1 の障害発生時には動作が中断される。クォーラム・ディスクを使用しないとき、この構成は 2 メンバ構成のクラスタには一般的である。M2 にポートを追加し、この構成にクォーラム・ディスクを追加してみるとよい。
2	1	1	---	---	0	クラスタを形成できるのは、両方のメンバが存在するときだけである。クラスタは一方のメンバの障害発生時に動作が中断される（4.4 節で説明したように、この構成は上記の構成よりも可用性が低い）。この構成にクォーラム・ディスクを追加してみるとよい。4.5 節を参照。
3	1	1	---	---	1	クォーラム・ディスクが構成され、このディスクに 1 ポートが割り当てられたので、クラスタはいずれか一方のメンバまたはクォーラム・ディスクの障害発生時でも稼働し続けることができる。2 メンバ構成のクラスタとしてはこの構成を推奨。
1	1	0	0	---	0	クラスタはメンバ M2 および M3 の障害発生時でも稼働し続けるが、M1 の障害発生時は動作が中断される。
2	1	1	0	---	0	クラスタが稼働し続けるには M1 および M2 の両方が稼働中である必要がある。M3 の障害発生時には、クラスタの動作は中断される。
3	1	1	1	---	0	クラスタはいずれか 1 つのメンバの障害発生時でも稼働し続けることができる。3 メンバ構成のクラスタとしてはこの構成を推奨。

表 4-1: 2 ~ 4 メンバ構成のクラスタ内でのメンバの `cluster_expected_votes` 設定とボートの割り当てによる影響 (続き)

メンバの期待ボート	M1	M2	M3	M4	クォーラム・ディスク	影響
4	1	1	1	---	1	クォーラム・ボートは 3 になるので、1 ボートを持つクォーラム・ディスクがあっても、この構成は上記の構成よりも可用性は低くなる。実際、クォーラム・ディスクに障害が発生した場合 (発生率は 0 に近いが)、いずれか 1 つのメンバの障害発生時には、クラスタの動作は中断される。 <sup>a</sup>
4	1	1	1	1	0	クラスタは、いずれか 1 つのメンバの障害発生時でも稼働し続けることができる。この構成にクォーラム・ディスクを追加してみるとよい。4.5 節を参照。
5	1	1	1	1	1	クラスタは、いずれか 1 つまたは 2 つのメンバの障害発生時、またはクォーラム・ディスクの障害発生時でも稼働し続けることができる。4 メンバ構成のクラスタとしてはこの構成を推奨。

<sup>a</sup>このような状況で考えられる 1 つの方法は、各メンバに 1 ボートを与え、1 ボートを持つクォーラム・ディスクを構成することである。期待ボートは 4 で、クォーラムは 3 になる。クォーラム・ディスクに障害が発生した場合は、`clu_quorum -d adjust 0` コマンドを使用してクォーラム・ディスクを削除する。または、メンバに障害が発生した場合は、`clu_quorum -m failed-cluster-member 0` コマンドを使用してメンバ・ボートを削除する。この結果、クラスタは (期待ボートが 3 でクォーラムが 2 になり)、別の障害が起こっても稼働し続けることができる。

## 4.8 接続マネージャのモニタ

接続マネージャは、次の 4 種類のイベントを EVM (イベント・マネージャ) のイベント・メッセージによって管理者に通知します。

- クラスタにノードが参加した。
- クラスタからノードが削除された。
- クォーラム・ディスクが使用不能になった (エラー、削除などが原因)。
- クォーラム・ディスクが再度使用可能になった。

これらの各イベントは、コンソール・メッセージによっても通知されます。

接続マネージャは、メンバのブート中やクラスタ・トランザクションの実行中に、さまざまな通知メッセージをコンソールに表示します。

クラスタ・トランザクションは、クラスタのすべてのメンバ上でクラスタ全体の状態をアトミックに変更するためのメカニズムです。つまり、すべてのメンバが新規の値を採用するか、どのメンバも新規の値を採用しないかのいずれかの状態になります。最も多く発生するクラスタ・トランザクションは、メンバシップ関連のトランザクションです。たとえば、クラスタの形成時やクラスタに対するメンバの追加または削除時に発生します。ある種の保守作業、たとえばクォーラム・ディスクの追加または削除、クラスタの期待ポートの変更、メンバのノード・ポートの変更を行うときにも、クラスタ・トランザクションは発生します。

クラスタ・トランザクションはグローバルに (クラスタ単位で) 発生するイベントですが、接続マネージャは、ローカルで発生したイベントに対応するメッセージもメンバのコンソールに表示します。たとえば、そのノードと別のノード (またはクォーラム・ディスク) との間の接続性の変化、クォーラムの獲得、クォーラムの喪失などを通知します。

## 4.9 接続マネージャのパニック

接続マネージャはクラスタのメンバを連続的に監視しています。クラスタの分断によって既存のクラスタが 2 つ以上のクラスタに分割されている場合は、ノードが自分たちをいずれかのクラスタのメンバであると認識することがまれにあります。4.3 節で説明したように、接続マネージャが動作を許可するクラスタは、多くても 1 つです。

クラスタ分断の発生時にデータの一貫性を維持するために、接続マネージャは 1 つのメンバをパニック状態にします。パニック文字列はクラスタ分断の検出時の状況を示します。このようなパニックは、接続マネージャの問題が原因で発生するのではなく、悪状況に対応して発生します。たとえば、徹底した対処策をとらないとデータの一貫性を維持できないような状況です。クラスタ分断を解消するには、1 つ (または複数) のメンバをリブートして、クラスタに再参加させるしか対処策はありません。

接続マネージャがクラスタの 1 つのメンバをパニック状態にするのは、次のような状況が発生したときです。

- クォーラム・ディスクが 2 つのクラスタに接続された。

```
CNX QDISK: configuration error. Qdisk in use by cluster of different name.  
CNX QDISK: configuration error. Qdisk written by cluster of different name.
```

- クラスタ分断後に複数のクラスタ間でクォーラム・ディスクの所有権の競合が発生した。

この状況が発生したメンバは、引き続きクォーラム・ディスクの所有権を要求するか、パニック状態に移行してこの所有権を別のクラスタに譲ります。

```
CNX QDISK: Yielding to foreign owner with quorum.  
CNX QDISK: Yielding to foreign owner with provisional quorum.  
CNX QDISK: Yielding to foreign owner without quorum.
```

- あるクラスタのメンバであるノード上の接続マネージャが、別のクラスタのメンバであるノード (または同じクラスタからクラスタ分断によって発生した別のクラスタのメンバであるノード) を検出した。

この状況が発生したノードは、クォーラムの状態に応じて、パニック状態に移行するように別のノードに指示するか、自身がパニック状態に移行します。

```
CNX MGR: restart requested to resynchronize with cluster with quorum.  
CNX MGR: restart requested to resynchronize with cluster
```

- パニック状態に移行したノードがクラスタを検出し、リポートしてそのクラスタに参加しようとした。

```
CNX MGR: rcnx_status: restart requested to resynchronize with cluster  
with quorum.  
CNX MGR: rcnx_status: restart requested to resynchronize with cluster
```

- 通信上の問題のため再構成中にノードがクラスタから削除された。

```
CNX MGR: this node removed from cluster
```

## 4.10 不適切な期待ポートおよびノード・ポート設定に関するトラブルシューティング

クラスタがクォーラムを維持している限り、`clu_quorum` コマンドを使って、クラスタ内のノード・ポート、期待ポート、およびクォーラム・ディスク・ポートを調節できます。このコマンドに `-f` オプションを使用すれば、現在ダウンしているメンバにも変更を反映できます。

ただし、クラスタのいずれかのメンバがクォーラムを失った場合は、そのメンバの入出力操作はすべて中断され、クラスタ・インターコネクト以外のすべてのネットワーク・インタフェースが停止状態になります。クラスタ全体のリソースへのアクセスが必要なコマンド (`clu_quorum` など) もすべて、そのメンバ上では機能しなくなります。この場合は、クォーラムを失ったメン

バがクォーラムを満たせるだけのポートを別のメンバに割り当て、そのメンバをクラスタに再参加させて、クォーラムを回復させるか、あるいはクラスタ・メンバを停止して、リブートしなければなりません。

クォーラムの喪失でハングアップしているクラスタや、ポート不足のために形成不能のクラスタでは、ポート構成の調節が必要な場合があります。以降の各項では、クラスタに関するいくつかの問題とそれらの解決に使用できるメカニズムについて説明します。

#### 4.10.1 クラスタのメンバまたはクォーラム・ディスクに障害が発生しクラスタがクォーラムを失った後にクラスタに参加させる

1 つまたは複数のメンバ (またはクォーラム・ディスク) にハードウェアの問題が発生し、クラスタはそれらのメンバを失いました。問題が発生したメンバはリブート不能です。それらのメンバを失ったことで、クラスタはクォーラムを失いました。残りの稼働中のメンバの期待ポートまたはノード・ポートの設定では、メンバが減った現在のクラスタの可用性を維持できません。クラスタはクォーラムを失い、ハングアップしています。

この種のクォーラム喪失状況は、クラスタ全体をシャットダウンしなくても解決できます。手順としては、1 つのクラスタ・メンバを停止し、クラスタに参加できるようにそのメンバをリブートして、クォーラムを回復します。このメンバをブートした後に、`clu_quorum` コマンドを使用して、本来の問題を修正する必要があります。

---

##### 注意

---

1 つのクラスタ・メンバだけが稼働中であるか、またはクォーラム・ディスクに障害が発生した場合は、4.10.2 項で説明する手順を使用し、十分なポートを持つクラスタ・メンバをブートして、クラスタを形成します。

---

1 つまたは複数のメンバあるいはクォーラム・ディスクに障害が発生したためにクォーラムを失ったクラスタのクォーラムを回復するには、次の手順に従います。

1. Halt ボタンを使用して、クラスタの 1 つのメンバを停止します。
2. 停止したメンバを対話形式でリブートします。リブート中、ブート元のカーネルの名前を要求されたら、カーネル名と `clubase` の

`cluster_adjust_expected_votes` 属性値 0 (ゼロ) の両方を指定します。この属性に 0 を設定すると、接続マネージャは、クラスタ内で現在使用可能なメンバとクォーラム・ディスクの総数と同数の期待ポートを設定します。

---

注意

---

`cluster_adjust_expected_votes` トランザクションは、ブートするノードがクラスタに参加した後でのみ実行されます。したがって、この方法は既存のクラスタがクォーラムの喪失でハングした場合にのみ有効です。期待ポートが高すぎてクラスタを形成できなければ、`cluster_adjust_expected_votes` が実行できず、ブートしたメンバがハングします。この場合、4.10.2 項で説明しているいずれかの方法を使用して、クラスタからメンバをブートしなければなりません。

---

次に、対話形式のリブートの例を示します。

```
>>> boot -fl "ia"
(boot dkb200.2.0.7.0 -flags ia)
block 0 of dkb200.2.0.7.0 is a valid boot block
reading 18 blocks from dkb200.2.0.7.0
bootstrap code read in
base = 200000, image_start = 0, image_bytes = 2400
initializing HWRPB at 2000
initializing page table at fff0000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

:

Enter kernel_name [option_1 ... option_n]
Press Return to boot default kernel
'vmunix':vmunix
clubase:cluster_adjust_expected_votes=0 Return
```

ブートを再開すると、そのメンバはクラスタに参加でき、新規の期待ポート値が接続マネージャによってその他のメンバに通知されます。その結果、クォーラムが回復されます。



---

## 警告

---

`cluster_adjust_expected_votes` の設定は、現在稼働中のクラスタの期待ポート設定のみを変更し、クラスタ全体が稼働している場合にかぎり使用されます。  
`/etc/sysconfigtab` ファイル内に格納されている値は変更されません。この時点で、クラスタ内のノード・ポート、期待ポート、およびクォーラム・ディスクを明示的に再構成しない場合、クラスタの次回リブート時には、ブート・メンバはクォーラムを満たさず、クラスタを形成できません。この理由から、必要に応じ、次の手順に従って、クラスタ内のこのメンバとその他のメンバのノード・ポートと期待ポートの値を修正してください。

- 故障しているハードウェアが修理されるかまたは取り換えられるまで、表 4-2 を参照し、適切な `clu_quorum` コマンドを使用して、クラスタのポート構成を一時的に修正します。通常は、クラスタが起動して安定すると、`clu_quorum` コマンドを使用して、本来の問題を修正します。たとえば、次のようにします。

- ハードウェアに問題があるメンバのノード・ポートを減らす。

```
# clu_quorum -f -m member-ID lower_node_votes_value
```

このメンバのブート・ディスクにアクセスできない場合（たとえばブート・ディスクがメンバのプライベート・バス上にある場合）、このコマンドはエラーを返すことがあります。この理由からこのコマンドの実行が失敗する場合は、`clu_quorum -f -e` コマンドを使って、期待ポートを適切な値に調節します。

- すべてのメンバの期待ポートから、ハードウェアの問題によって投票不能になったが、そのポートを削除できないメンバ分のポートを差し引く。

```
# clu_quorum -f -e lower_expected_votes_value
```

`clu_quorum -f` コマンドがダウンしているメンバの `/etc/sysconfigtab` ファイルにアクセスできない場合は、適切なメッセージが表示されて失敗します。これは、通常、ダウンしているメンバのブート・ディスクが、そのメンバのプライベート・バス上にある場合に起こります。そのようなメンバが原因となっているクォー

ラムの問題を解決するには、そのメンバを対話形式でブートして、`cluster_expected_votes` をそのメンバがクラスタに参加できる値に設定します。そのメンバがクラスタに参加すると、`clu_quorum` コマンドを使用して、この項で説明したように、ポートの設定を修正します。

表 4-2 に、クォーラム・ディスクのある 4 メンバ構成のクラスタと、クォーラム・ディスクのない 5 メンバ構成のクラスタで、クォーラムを回復する例を示します。この表で、NC は、クラスタでメンバまたはクォーラム・ディスクが構成されていないことを示します。

表 4-2: メンバやクォーラム・ディスクに障害が発生したクラスタのクォーラム喪失の解決例

M1	M2	M3	M4	M5	クォーラム・ディスク	手順
稼働, 1 ポート	稼働, 1 ポート	障害, 1 ポート	障害, 1 ポート	NC	障害	<p>1. <code>clubase:adjust_expected_votes=0</code> で M1 または M2 を対話型でブートする。</p> <p>2. <code>clu_quorum -f -m</code> コマンドを使用して, M3 と M4 から ノード・ポートを削除する。</p> <p>3. <code>clu_quorum -f -d remove</code> コマンドを使用して, クォーラム・ディスクを削除する。</p> <p>4. 故障したハードウェアを修理または交換する。2 メンバ・クラスタで, 障害に耐えなければならない場合に最も緊急に必要なのは, 投票クォーラム・ディスクである。<code>clu_quorum -f -d add</code> コマンドを使用して, 新しいクォーラム・ディスクを追加する。クォーラム・ディスクがクラスタ全体で認識されるようにするには, すべてのクラスタ・メンバ上で <code>hwmgr -scan scsi</code> コマンドを実行する必要がある。</p> <p>クォーラム・ディスクを追加できない場合は, <code>clu_quorum -f -m</code> コマンドを使用して M1 または M2 からポートを削除する。障害の発生したメンバが長時間利用できない場合は, <code>clu_delete_member</code> コマンドを使用してそのメンバをクラスタから削除する。</p>

表 4-2: メンバやクォーラム・ディスクに障害が発生したクラスタのクォーラム喪失の解決例 (続き)

M1	M2	M3	M4	M5	クォーラム・ディスク	手順
稼働, 1 ポート	稼働, 1 ポート	障害, 1 ポート	障害, 1 ポート	障害, 1 ポート	NC	<p>1. clubase:adjust_expected_votes=0 で M1 または M2 を対話型でブートする。</p> <p>2. clu_quorum -f -m コマンドを使用して, M3, M4, および M5 から ノード・ポート を削除する。</p> <p>3. 故障したハードウェアを修理または交換する。2 メンバ・クラスタで障害に耐えなければならない場合に最も緊急に必要となるのは, 投票クォーラム・ディスクである。clu_quorum -f -d add コマンドを使用して, 新しいクォーラム・ディスクを追加する。クォーラム・ディスクがクラスタ全体で認識されるようにするには, すべてのクラスタ・メンバ上で hwmgr -scan scsi コマンドを実行する必要がある。</p> <p>障害の発生したメンバが長時間利用できない場合は, clu_delete_member コマンドを使用してそのメンバをクラスタから削除する。</p>

4.10.2 メンバがブートとクラスタ形成に必要な数のポートを持っていない場合のクラスタ形成

クラスタを形成できません。すべてのメンバをブートしようとする、クラスタを形成できず、各メンバはハングアップします。これらのメンバには、クォーラムに達するのに必要なポートが不足しています。ハードウェアの故障がよく起こる小規模のクラスタも、最後に稼働していた投票メンバがクォーラムを失うような構成になる場合があります。

次の手順を実行すると、クラスタを形成するのに十分なポートを持つ1つのクラスタ・メンバをブートすることができます。その後、ノード・ポートを調節して、残りのメンバをクラスタにブートします。

1. 各クラスタ・メンバを停止します。
2. 表 4-3 を参照して、そのクラスタに特有なクォーラムを失った状況を解決するため、ブート時に調整する必要のあるカーネル属性を決定します。
3. クラスタの1つの投票メンバを対話形式でブートします。ブート中、ブート元のカーネルの名前を要求されたら、カーネル名と推奨するカーネル属性設定の両方を指定します。たとえば、メンバとクォーラム・ディスクの両方に障害が発生した2メンバ構成のクラスタ(2つのノード・ポートと1つのクォーラム・ディスク)では、`clubase:cluster_expected_votes=1` `clubase:cluster_qdisk_votes=0` を入力します。

次に、対話形式のリブートの例を示します。

```
>>> boot -fl "ia"
(boot dkb200.2.0.7.0 -flags ia)
block 0 of dkb200.2.0.7.0 is a valid boot block
reading 18 blocks from dkb200.2.0.7.0
bootstrap code read in
base = 200000, image_start = 0, image_bytes = 2400
initializing HWRPB at 2000
initializing page table at fff0000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

:
:

Enter kernel_name [option_1 ... option_n]
Press Return to boot default kernel
'vmunix':vmunix
clubase:cluster_expected_votes=1 clubase:cluster_qdisk_votes=0 Return
```

ブートを再開すると、このメンバはクラスタを形成できます。

4. 故障したハードウェアを修理するか交換するまで、表 4-3 を参照し、適切な `clu_quorum` コマンドを使用して、クラスタのポートの構成を一時的に修正します。利用できないクォーラム・ディスクが問題の原因となっている場合は、そのディスクが利用可能かどうかと、ポートを持っているかどうかを確認します。必要であれば、クォーラム・ディスクを交換します(4.5.1 項を参照)。そうでない場合、他のメンバをブートできないことがあります。
5. 残りのメンバをリブートします。

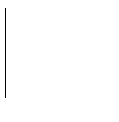
表 4-3 に、クラスタを形成するのに十分なボートを持つクラスタ・メンバをブートして、クォーラム不足を解決する方法の例を示します。この表で、NC は、メンバまたはクォーラム・ディスクがクラスタで構成されていないことを示します。

表 4-3: クラスタを形成するのに十分なボートを持つメンバのブートによるクォーラム不足の解決例

M1	M2	M3	クォーラム・ディスク	手順
稼働, 1 ボート	稼働, 0 ボート	NC	障害, 1 ボート	1. clubase:cluster_node_votes=1 で M2 を対話型でブートする。 2. clu_quorum -f -d remove コマンドを使用して、クォーラム・ディスクを削除する。 3. clu_quorum -f -d add コマンドを使用して、故障したクォーラム・ディスクを交換する。この結果、2 つのノード・ボートと 1 つのクォーラム・ディスク・ボートを持つ 2 メンバ構成のクラスタとなる (ディスクまたは 1 メンバの障害に耐える構成)。クォーラム・ディスクを交換できない場合は、clu_quorum -f -m コマンドを使用して、1 つのメンバのボートを削除する。この結果、非投票メンバの障害に耐える構成になる。

表 4-3: クラスタを形成するのに十分なポートを持つメンバのブートによるクォーラム不足の解決例 (続き)

M1	M2	M3	クォーラム・ディスク	手順
稼働, 1 ポート	障害, 1 ポート	NC	障害, 1 ポート	<ol style="list-style-type: none"> <li>clubase:cluster_expected_votes=1 および clubase:cluster_qdisk_votes=0 で M1 を対話型でブートする。</li> <li>clu_quorum -f -d remove コマンドを使用して, クォーラム・ディスクを削除する。</li> <li>clu_quorum -f -m 2 0 コマンドを使用して, M2 のポートを削除する。</li> <li>故障したハードウェアを修理または交換する。投票クォーラム・ディスクのある 2 番目の投票メンバを直ちに入手できない場合は, 暫定的な解決方法として, ポートのない 2 番目のメンバを追加する。この結果, 非投票メンバの障害に耐える構成となる。</li> </ol>
稼働, 1 ポート	障害, 1 ポート	障害, 1 ポート	NC	<ol style="list-style-type: none"> <li>clubase:cluster_expected_votes=1 で M1 を対話型でブートする。</li> <li>適切な clu_quorum -f -m コマンドを使用して, M2 および M3 からノードを削除する。</li> <li>故障したハードウェアを修理または交換する。投票クォーラム・ディスクのある 2 番目の投票メンバを直ちに入手できない場合は, 暫定的な解決方法として, ポートのない 2 番目のメンバを追加する。この結果, 非投票メンバの障害に耐える構成となる。</li> </ol>





# 5

## クラスタ・メンバの管理

この章では次の項目について説明します。

- 構成変数の管理 ( 5.1 節)
- カーネル属性の管理 ( 5.2 節)
- クラスタへのリモート・アクセスの管理 ( 5.3 節)
- クラスタのシャットダウン ( 5.4 節)
- クラスタ・メンバのシャットダウンと起動 ( 5.5 節)
- クラスタのシャットダウンとシングルユーザ・モードへの移行 ( 5.6 節)
- クラスタ・メンバのリブート ( 5.7 節)
- クラスタからのメンバの削除 ( 5.8 節)
- クラスタ・メンバの削除とスタンドアロン・システムとしての復元 ( 5.9 節)
- クラスタ名または IP アドレスの変更 ( 5.11 節)
- 別の IP サブネットへのクラスタの移動 ( 5.10 節)
- メンバ名, IP アドレス, またはクラスタ・インターコネクト・アドレスの変更 ( 5.12 節)
- ソフトウェア・ライセンスの管理 ( 5.13 節)
- レイヤード・アプリケーションのインストールと削除 ( 5.14 節)
- 課金サービスの管理 ( 5.15 節)

クラスタ・メンバの管理に関連する次のエントリについては, TruCluster Server 『クラスタ・インストレーション・ガイド』を参照してください。

- クラスタへの新規メンバの追加
- クラスタ・メンバの再インストール

- ソフトウェアのライセンス登録

Tru64 UNIX と TruCluster Server の可用性および保守性に関する構成と管理については、Tru64 UNIX の『*Managing Online Addition and Removal*』を参照してください。このマニュアルでは、可用性の高いシステムを構成して管理するためのガイドラインを、システム構成要素の OLAR (Online Addition and Replacement) 管理を中心に説明しています。

注意

Tru64 UNIX の『*Managing Online Addition and Removal*』で説明されているように、システムに固有なポリシーは `/etc/olar.config` ファイルで定義し、クラスタ単位のポリシーは `/etc/olar.config.common` ファイルで定義します。システムの `/etc/olar.config` ファイルで行った設定は、そのシステムに関する限り、`/etc/olar.config.common` ファイルに記述したクラスタ単位のポリシーより優先します。

## 5.1 構成変数の管理

`/etc/rc.config*` ファイルの階層化により、ローカル・エリア・ネットワーク (LAN) 内およびクラスタ内のすべてのシステムの構成変数を一元管理できます。表 5-1 に、各種構成ファイルの有効範囲を示します。

表 5-1: `/etc/rc.config*` ファイル

ファイル	有効範囲
<code>/etc/rc.config</code>	メンバ固有の変数を保存している。 <code>/etc/rc.config</code> はテキスト依存シンボリック・リンク (CDSL) である。クラスタの各メンバにはそれぞれ固有の <code>/etc/rc.config</code> ファイルがある。 <code>/etc/rc.config</code> 内の構成変数は <code>/etc/rc.config.common</code> と <code>/etc/rc.config.site</code> 内の構成変数を上書きする。

### 5-2 クラスタ・メンバの管理

表 5-1: /etc/rc.config\* ファイル (続き)

ファイル	有効範囲
/etc/rc.config.common	<p>クラスタ単位の変数を保存している。これらの変数はクラスタのすべてのメンバに適用される。</p> <p>/etc/rc.config.common 内の構成変数は、 /etc/rc.config.site 内の構成変数を上書きするが、 /etc/rc.config 内の構成変数によって上書きされる。</p>
/etc/rc.config.site	<p>サイト単位の変数を保存している。これらの変数は LAN 上のすべてのマシンの変数と一致する。</p> <p>このファイル内の値は、/etc/rc.config.common または /etc/rc.config 内の対応する値によって上書きされる。</p> <p>省略時の設定では、/etc/rc.config.site は作成されない。サイト単位の変数を設定する場合は、ファイルを /etc/rc.config.site という名前で作成して、サイト単位の構成変数を設定し、参加している各システムにそのファイルをコピーする。</p> <p>その後、各システム上の /etc/rc.config 内で、 /etc/rc.config.common を実行する行の直前に、次のコードを追加する。</p> <pre># Read in the cluster sitewide attributes # before overriding them with the # clusterwide and member-specific values. # ./etc/rc.config.site</pre> <p>詳細は、rcmgr(8) を参照。</p>

rcmgr コマンドは標準の検索順 (最初に /etc/rc.config、次に /etc/rc.config.common、最後に /etc/rc.config.site) でこれらの変数にアクセスし、指定された値を検出または設定します。

特定のメンバの実行時構成変数を取得または設定するには、-h オプションを使用します。これによって、このコマンドの実行対象は /etc/rc.config (メンバ固有の構成ファイルの CDSL) になります。

このコマンドの実行対象をクラスタ全体にするには、-c オプションを使用します。これによって、このコマンドの実行対象は /etc/rc.config.common (クラスタ単位の構成ファイル) になります。

-h も -c も指定しない場合は、`/etc/rc.config` 内のメンバ固有の値が使用されます。

メンバ固有の構成変数については、付録 B を参照してください。

## 5.2 カーネル属性の管理

クラスタの各メンバは、独自にカーネルを実行するので、それぞれ独自に `/etc/sysconfigtab` ファイルを持っています。このファイルには、メンバ固有の静的な属性設定が格納されます。クラスタ単位の `/etc/sysconfigtab.cluster` ファイルがありますが、その用途は `/etc/rc.config.common` とは異なり、TruCluster Server 製品に付属するユーティリティに限られています。

ここでは、各 TruCluster Server サブシステムで提供されているカーネル属性のリストの一部を示します。

次のコマンドを使用して、指定したサブシステムのこのような属性の現在の設定を表示します。

```
# sysconfig -q subsystem-name attribute-list
```

すべてのサブシステムのリストと状態を得るには、次のコマンドを使用します。

```
# sysconfig -s
```

ここで示したクラスタ関連のカーネル属性のほかに、2 つのカーネル属性がクラスタのインストール中に設定されます。表 5-2 に、このようなカーネル属性を示しています。これらの属性に設定されている値は、増やすことはできますが、減らすことはできません。

表 5-2: 削減できないカーネル属性

属性	値 (削減不可)
vm_page_free_min	30
vm_page_free_reserved	20

表 5-3 は、各 TruCluster Server 構成要素に対応するサブシステム名を示しています。

表 5-3: 構成可能な TruCluster Server サブシステム

サブシステム名	構成要素	詳細
cfs	クラスタ・ファイル・システム (CFS)	sys_attrs_cfs(5)
clua	クラスタ別名	sys_attrs_clua(5)
clubase	クラスタ・ベース	sys_attrs_clubase(5)
cms	クラスタ・マウント・サービス	sys_attrs_cms(5)
cnx	接続マネージャ	sys_attrs_cnx(5)
dln	分散ロック・マネージャ	sys_attrs_dln(5)
drd	デバイス要求ディスパッチャ	sys_attrs_drd(5)
hwcc	ハードウェア・コンポーネント・クラスタ	sys_attrs_hwcc(5)
icsnet	ノード間通信サービスのネットワーク・サービス	sys_attrs_icsnet(5)
ics_hl	上位レベルのノード間通信システム (ICS)	sys_attrs_ics_hl(5)
mcs	Memory Channel アプリケーション・プログラミング・インタフェース (API)	sys_attrs_mcs(5)
rm	Memory Channel	sys_attrs_rm(5)
token	CFS トークン・サブシステム	sys_attrs_token(5)

カーネル・サブシステムの性能を調整するには、次のいずれかの方法で `/etc/sysconfigtab` ファイル内の属性をいくつか設定します。

- `/etc/sysconfigtab` ファイル内の *subsystem name* スタンザ・エントリの追加または編集を行い、属性の値を変更して、次のシステム・ブート時に新しい値が有効になるようにします。
- 次のコマンドを使用して、再設定できる属性の値を変更し、その新しい値が実行時に直ちに有効になるようにします。

```
# sysconfig -r subsystem-name attribute-list
```

次のシステム・ブート時まで変更を保持するには、`/etc/sysconfigtab` ファイルも編集しなければなりません。たとえば、`drd-print-info` 属性の値を 1 に変更するには、次のコマンドを入力します。

```
# sysconfig -r drd drd-print-info=1
drd-print-info: reconfigured
```

Tru64 UNIX 『システム管理ガイド』で説明しているように、構成マネージャのフレームワークを使用すれば、属性を変更する以外に、別のホスト上のクラスタ・カーネル・サブシステムを管理することもできます。この作業を行うには、リモート・クライアント・システムの `/etc/cfgmgr.auth` ファイル内にホスト名を設定し、次の例に示すように、`/sbin/sysconfig` コマンドに `-h` オプションを指定します。

```
# sysconfig -h fcbr13 -r drd drd-do-local-io=0
drd-do-local-io: reconfigured
```

### 5.3 クラスタ内およびクラスタからのリモート・アクセスの管理

クラスタからの `rlogin`、`rsh`、または `rcp` コマンドでは、省略時のクラスタ別名が送信元アドレスとして使用されます。したがって、非クラスタ・ホストで、クラスタ内の任意のアカウントからのリモート・ホスト・アクセスを許可しなければならない場合は、非クラスタ・メンバの `.rhosts` ファイルに、`/etc/hosts` ファイル内のいずれかの形式、あるいはネットワーク情報サービス (NIS) またはドメイン・ネーム・システム (DNS) で解決できる形式で、クラスタ別名を登録する必要があります。

この要件は、クラスタ・メンバ間で機能する `rlogin`、`rsh`、または `rcp` にも適用されます。クラスタ作成時に、`clu_create` ユーティリティで、必要なホストすべての名前への入力が必要で、その名前が適切な形式で正しい場所に設定されます。`clu_add_member` では、新しいメンバがクラスタに追加されるときに同じ処理が行われます。`/.rhosts` を編集しなくても、クラスタ・メンバからクラスタ別名に対して、あるいは個々のメンバ間で `/bin/rsh` コマンドを使用できます。`/etc/hosts` と `/.rhosts` 内に生成された名前へのエントリを変更しないでください。

`/etc/hosts` と `/.rhosts` のファイルが間違っていて構成された場合、多くのアプリケーションは正しく機能しません。たとえば、AdvFS (Advanced File System) の `rmvol` と `addvol` のコマンドでは、そのコマンドが実行されるメンバがドメインのサーバでない場合は、`rsh` が使用されます。これらのコマンドは、`/etc/hosts` または `/.rhosts` が正しく構成されていない場合には失敗します。

次のエラーは、`/etc/hosts` や `/.rhosts` のファイルが間違っていて構成されたことを示しています。

```
rsh cluster-alias date
Permission denied.
```

## 5.4 クラスタのシャットダウン

クラスタのすべてのメンバを停止するには、`shutdown` コマンドに `-c` オプションを使用します。たとえば、5 分以内にクラスタをシャットダウンするには、次のコマンドを入力します。

```
# shutdown -c +5 Cluster going down in 5 minutes
```

クラスタの特定のメンバのシャットダウンについては、5.5 節を参照してください。

シャットダウンの猶予期間内 (クラスタの `shutdown` コマンドが入力されてからシャットダウンが実際に行われるまでの時間)、`clu_add_member` コマンドは無効であり、新規メンバをクラスタに追加できません。

この猶予期間内にクラスタのシャットダウンを取り消すには、次の手順に従って、`shutdown` コマンドに関連付けられたプロセスを強制終了します。

1. `shutdown` コマンドに関連付けられたプロセス ID (PID) を取得します。たとえば、次のように入力します。

```
# ps ax | grep -v grep | grep shutdown
14680 tty5      I <      0:00.01 /usr/sbin/shutdown +20 going down
```

猶予期間内での `shutdown` の進行状況に応じて、`ps` は `/usr/sbin/shutdown` または `/usr/sbin/clu_shutdown` のいずれかを表示します。

2. 任意のメンバから、取得した PID を指定して `kill` コマンドを実行することにより、シャットダウン・プロセスをすべて強制終了します。たとえば、次のように入力します。

```
# kill 14680
```

猶予期間内にシャットダウン・プロセスを強制終了した場合、シャットダウンが取り消されます。

コマンド `clu_quorum`、`clu_add_member`、`clu_delete_member`、または `clu_upgrade` が実行中である場合、`shutdown -c` コマンドを実行するとエラーが生じます。

クラスタ全体をリブートするコマンドはありません。コマンド `shutdown -r` , `reboot` , および `halt` は、それらの実行元のメンバのみに影響します。コマンド `halt` , `reboot` , および `init` は、クラスタにマウントされたファイル・システムをアンマウントしないように修正されています。そのためクラスタは、いずれか 1 つのメンバが停止またはリブートされても、クォーラムを維持している限り、稼働し続けます。

詳細は、`shutdown(8)` を参照してください。

## 5.5 クラスタの特定のメンバのシャットダウンと起動

メンバをブートするときは、`clu_add_member` コマンドによって作成されたブート・ディスクからブートしなければなりません。ブート・ディスクのコピーからはブートできません。

クラスタの特定のメンバのシャットダウンはスタンドアロン・サーバのシャットダウンより複雑です。クォーラムの維持に必須のポートを持つクラスタ・メンバ (重要な投票メンバと呼びます) を停止した場合、クラスタはクォーラムを失い、ハングアップします。その結果、停止したメンバをリブートするまで、クラスタのどのメンバからもコマンドを入力できません。したがって、クラスタのメンバをシャットダウンする前に、まずそのメンバのポートがクォーラムの維持に必須かどうか調べなければなりません。また、シャットダウンしようとしているクラスタ・メンバが、`restricted` 配置ポリシーで 1 つ以上のアプリケーションの唯一のホスト・メンバかどうかとも判断する必要があります。

### 5.5.1 重要な投票メンバの識別

重要な投票メンバのあるクラスタは、縮退モードで稼働しているか (たとえば、1 つ以上の投票メンバまたは 1 つのクォーラム・ディスクがダウンしている)、または最初から可用性を考慮して構成されていません (たとえば、各メンバにポートが割り当てられている 2 メンバ構成)。重要な投票メンバをクラスタから削除すると、クラスタがハングして、可用性が低下します。クラスタ・メンバを停止または削除する前に、重要な投票をしていないことを確認してください。

メンバが重要な投票メンバかどうかを調べるには、次の手順に従います。

1. 可能であれば、クラスタのすべての投票メンバが稼働中であることを確認します。



2. `clu_quorum` コマンドを入力し、問題のメンバの現在のポート、クォーラム・ポート、およびノード・ポートの実行時の値をメモしておきます。
3. 現在のポートの値からメンバのノード・ポートの値を引きます。その結果がクォーラム・ポートの値より小さい場合、そのメンバは重要な投票メンバです。そのメンバをシャットダウンすると、クラスタがクォーラムを失い、ハングします。

### 5.5.2 重要な投票メンバの停止または削除の準備

重要な投票メンバの停止または削除を行う前に、クォーラムを維持するクラスタでそのポートが必要とされなくなっていることを確認してください。それを確認する最善の方法は、期待ポートを増やさずに、ノード・ポートまたはクォーラム・ディスク・ポートをクラスタにリストアすることです。具体的な方法は、以下のとおりです。

- 現在停止している投票メンバをブートする。
- 停止しているメンバのポートを削除 (`clu_quorum -f -m` コマンドを使用) し、ポートが1のクォーラム・ディスクを構成 (`clu_quorum -f -d add` コマンドを使用) する。こうすれば、期待ポートを増やさず、またクォーラム・ポートの値も変えずに、クラスタに現在のポートが追加される。

クラスタのポートが偶数の場合は、新たに投票メンバを追加するか、クォーラム・ディスクを構成すれば、重要な投票メンバを重要でないメンバにすることもできます。このような場合には、期待ポートは増えますが、クォーラム・ポートは変わりません。

### 5.5.3 重要でないメンバの停止

重要でないメンバ、つまりポートを持たないメンバ、またはクォーラムの維持に必須でないポートを持つメンバは、スタンドアロン・システムのようにシャットダウン、停止、またはリブートできます。

シャットダウンするメンバ上で `shutdown` コマンドを実行します。メンバを停止するには、次のコマンドを入力します。

```
# shutdown -h time
```

メンバをリブートするには、次のコマンドを入力します。

```
# shutdown -r time
```

重要な投票メンバの識別については、5.5.1 項を参照してください。

#### 5.5.4 ホスト・メンバのシャットダウン

アプリケーションを実行させるメンバは、CAA (cluster application availability) プロファイルの中で各ホスト・メンバを空白で区切って指定します。ホスト・メンバのリストは、`caa(4)` で説明しているように、アプリケーション・リソースのフェイルオーバー・ポリシー (favored または restricted) と組み合わせて使用します。

配置ポリシーが restricted であるアプリケーションをただ 1 つのホスト・メンバだけがサポートしている場合は、そのクラスタ・メンバをシャットダウンするときに他のホスト・メンバを指定する必要があります。そうしなければ、そのメンバがダウンしている間、アプリケーションを実行できなくなります。こうした場合に備えて、他のホスト・メンバを追加するか、または既存のホスト・メンバを他のメンバに切り換えます。

この処理は次の手順で行います。

1. 現在のホスト・メンバと配置ポリシーを確認します。

```
# caa_profile -print resource-name
```

2. シャットダウンするクラスタ・メンバが唯一のホスト・メンバである場合は、ホスト・メンバをリストに追加するか、または既存のホスト・メンバを切り換えます。

```
# caa_profile -update resource-name -h hosting-member another-hosting-member
# caa_profile -update resource-name -h hosting-member
```

3. CAA のレジストリ・エントリを最新のリソース・プロファイルで更新します。

```
# caa_register -u resource-name
```

4. アプリケーションを他のメンバに再配置します。

```
# caa_relocate
resource-name -c member-name
```

#### 5.6 クラスタ・メンバをシャットダウンしてシングルユーザ・モードにする

クラスタ・メンバをシャットダウンしてシングルユーザ・モードにする必要がある場合は、メンバを停止してからシングルユーザ・モードでブートしなければなりません。この方法でメンバをシャットダウンすれば、そのメ

ンバがクラスタに対して提供するサービスを最低限に抑えることができます。また、稼働中のクラスタがシングルユーザ・モードで動作しているメンバに依存する割合も最低にすることができます。特に、クラスタ・メンバの状態を `DOWN` にしてからフェイルオーバを行わなければならないサービスがある場合、メンバを停止することによって、その要件を満たすことができます。クラスタ・メンバを最初に停止しなければ、サービスが期待どおりにフェイルオーバしません。

クラスタ・メンバをシングルユーザ・モードに移行するには、`shutdown -h` コマンドを用いてメンバを停止させ、次にメンバをシングルユーザ・モードでブートします。システムがシングルユーザ・モードになったら、`init s`、`bcheckrc`、および `lmf reset` コマンドを実行します。以下に例を示します。

---

#### 注意

---

クラスタ・メンバを停止する前に、そのメンバのポートがなくなってもクラスタがクォーラムを維持できることを確認しておいてください。また、そのクラスタ・メンバが、`restricted` 配置ポリシーで 1 つ以上のアプリケーションの唯一のホスト・メンバでないことも確認してください。

---

```
# /sbin/shutdown -h now

>>> boot -fl s

# /sbin/init s
# /sbin/bcheckrc
# /usr/sbin/lmf reset
```

シャットダウンされてシングルユーザ・モードになったクラスタ・メンバ(つまり、停止してからシングルユーザ・モードでブートするという推奨シャットダウン手順を使わなかった場合)は、状態が `UP` のままです。クラスタ・メンバをこのようにシャットダウンしてシングルユーザ・モードにした場合は、そのメンバの投票状態には影響しません。シャットダウンされてシングルユーザ・モードになる前に投票メンバであった場合、そのメンバはシングルユーザ・モードでも引き続き投票メンバになります。

## 5.7 クラスタ・メンバのリブート

すべてのメンバを同時にリブートしてはなりません。すべてのクラスタ・メンバを同時にリブートしようとする、クォーラム喪失のため、1 つまたは複数のメンバがダウンの途中でハングし、他のリブート中のノードが、クラスタに再度加わることができないため、ブートに失敗します (ハング・ノード)。

クラスタ全体をリブートするために使用する方法は、目的によって異なります。

- クラスタを停止することなくすべてのクラスタ・ノードをリブートするには、1 度に 1 メンバをリブートして、次のメンバをリブートする前に、リブート中のメンバがクラスタに再加入できるようにします。

---

### 注意

---

クラスタ・メンバをリブートする前に、そのメンバのポートがなくても、クラスタがクォーラムを維持できることを確認します。また、そのクラスタ・メンバが、restricted 配置ポリシーで 1 つ以上のアプリケーションの唯一のホスト・メンバでないことも確認してください。

- クラスタ全体をリブートするには (クラスタ状態は失われる)、shutdown -c コマンドでクラスタ全体をシャットダウンしたのち、メンバをブートします。

## 5.8 クラスタからのメンバの削除

クラスタからメンバを削除するには、clu\_delete\_member コマンドを使用します。

---

### 警告

---

TruCluster Server を再インストールする場合は、TruCluster Server 『クラスタ・インストール・ガイド』を参照してください。既存クラスタからメンバを削除し、削除したそのメンバから 1 メンバ構成の新規クラスタを作成しないでください。その新規クラスタが既存クラスタと同じ名前の場合、新しくインス

ツールしたシステムが既存クラスタに参加する場合があります。  
これによってデータの破壊が起こる可能性があります。

---

`clu_delete_member` コマンドの構文は次のとおりです。

**`/usr/sbin/clu_delete_member [-f] [-m memberid]`**

メンバ ID を指定しなかった場合、削除するメンバのメンバ ID が要求されます。

`clu_delete_member` コマンドは次の処理を行います。

- メンバのブート・パーティションをマウントし、ブート・パーティション内のファイルをすべて削除します。その結果、このシステムはこのディスクからブートできなくなります。

---

#### 警告

---

`clu_delete_member -f` コマンドは、メンバのブート・ディスクにアクセスできないときでも、そのメンバを削除します。そのため、ブート・ディスクに障害が発生したメンバでも削除できます。ただし、アクセスできないブート・ディスクのメンバを削除すると、`clu_delete_member -f` は、(正常な `clu_delete_member` コマンドがするようには)稼働中のクラスタの期待ポートを調整しません。削除されるメンバが投票メンバの場合、期待ポートを適正に調整するように、メンバが削除された後で `clu_quorum -e` を使用してください。

このコマンドでメンバのブート・ディスクにアクセスできない場合は、クラスタのどのメンバもそのディスクから不用意にブートしないようにする必要があります。そのためには、そのディスクをクラスタから削除するか、再フォーマットするか、`disklabel` コマンドを使って非ブート・ディスクにします。

---

- メンバがポートを持っている場合は、クラスタ単位のクラスタ期待ポートの値を調節します。

- クラスタ全体のファイル・システムからメンバ固有のディレクトリとファイルをすべて削除します。

---

注意

---

`clu_delete_member` コマンドは、ディレクトリ `/cluster`、`/usr/cluster` および `/var/cluster` からメンバ固有のファイルを削除します。ただし、アプリケーションまたは管理者が `/usr/local` などのその他のディレクトリ内にメンバ固有のファイルを作成している場合もあります。その場合は、`clu_delete_member` の実行後に、これらのファイルを手作業で削除しなければなりません。この作業をしなかった場合は、新規メンバを追加し、同じメンバ ID を再使用すると、新規メンバはこれらの古い (おそらくエラーの原因になる) ファイルにアクセスしてしまいます。

- ファイル `/.rhosts` および `/etc/hosts.equiv` から、削除されたメンバのクラスタ・インターコネクト用のホスト名を削除します。
- ここまでの処理内容を削除ログ `/cluster/admin/clu_delete_member.log` に書き込みます。付録 C に削除ログ `clu_delete_member` の例を示しています。

クラスタからメンバを削除するには、次の手順を実行します。

1. そのメンバがクラスタの重要な投票メンバかどうかを調べます。そのメンバがクラスタに重要なボートを投じる場合、そのメンバを停止すると、クラスタはクォーラムを失い、動作が中断されます。メンバを停止する前に、5.5 節の手順に従って、そのメンバを停止しても安全かどうかを調べます。  
  
また、5.5.4 項で説明しているように、そのクラスタ・メンバが `restricted` 配置ポリシーで 1 つ以上のアプリケーションの唯一のホスト・メンバかどうか判断する必要があります。
2. 削除するメンバを停止します。
3. 可能であれば、クラスタのすべての投票メンバが稼働中であることを確認します。

4. 別のメンバから `clu_delete_member` コマンドを使って、そのメンバをクラスタから削除します。たとえば、停止したメンバのメンバ ID が 3 であり、そのメンバを削除するには、次のコマンドを入力します。

```
# clu_delete_member -m 3
```

5. `clu_delete_member` の実行時に、メンバのブート・ディスクがアクセスできない場合は、その旨を通知するメッセージを表示して、終了します。

たとえ、メンバのブート・ディスクにアクセスできないとしても、`-f` オプションの指定で、強制的にメンバを削除できます。この場合、削除されるメンバが投票メンバなら、メンバが削除された後にクラスタの期待ポートを手動で 1 ポート低くする必要があります。これは、次のコマンドで行います。

```
# clu_quorum -e expected-votes
```

メンバ削除時のログ・ファイル `/cluster/admin/clu_delete_member.log` の例については、付録 C を参照してください。

## 5.9 クラスタからのメンバの削除とスタンドアロン・システムとしての復元

スタンドアロン・システムとしてクラスタのメンバを復元するには、次の手順を実行します。

1. 5.5 節と 5.8 節の手順に従って、メンバを停止して削除します。
2. 停止したメンバをクラスタから物理的に切断し、クラスタ・インターコネクトとストレージを切断します。
3. 停止したメンバ上で、このメンバのローカル・ディスクを選択し、Tru64 UNIX をインストールします。システム・ソフトウェアのインストールについては、Tru64 UNIX 『インストール・ガイド』マニュアルを参照してください。

非クラスタ化システムへのクラスタ化 LSM (Logical Storage Manager) ボリュームの移動については、Tru64 UNIX 『*Logical Storage Manager*』を参照してください。

## 5.10 別の IP サブネットへのクラスタの移動

この節では 1 つの IP サブネットから別の IP サブネットへのクラスタの移動方法について説明します。通常、クラスタの外部 IP アドレスが別の IP サブネットを指すようにサイトのネットワーク・トポロジの再構成を行うときにだけ必要となります。

増大する複雑さを回避するために、クラスタを別の IP サブネットに移動する場合には、次の項目を変更する必要があります。

- クラスタの外部ネットワーク・インタフェース、インタフェースの IP 別名、およびクラスタ別名に関連した IP アドレス
- クラスタの外部ネットワーク・インタフェース、インタフェースの IP 別名、およびクラスタ別名に関連した IP アドレスとホスト名
- クラスタの外部ネットワーク・インタフェース、インタフェースの IP 別名、およびクラスタ別名に関連した IP アドレスとホスト名、および内部のクラスタ・インターコネクトに関連した IP アドレスとインタフェース名

---

### 注意

---

クラスタ管理を簡単にするため、メンバの外部ホスト名を変更した場合、メンバのクラスタ・インターコネクト名のホスト部分も変更したくなります。しかし、クラスタ・インターコネクトに使用されるネットワークがクラスタ専用であるため、クラスタ・インターコネクトで使われる IP 名とアドレスを変更せずに別のネットワークにクラスタを再配置することができます。この方法については、十分な情報が提供されています。メンバのホスト名とクラスタ・インターコネクト名のホスト名部分を一致させるかどうかはユーザ次第です。

---

この節には、情報収集と移動の実行の際に使用する表が用意されています。この表では、3 つの移動方法に対してそれぞれ必要とされるファイルの編集方法について説明しています。計画している移動方法に該当する表を使用してください。5.10.2 項は移動を開始する前の情報収集に役立ち、このチェックリストは編集の履歴を残せます。

この手順を用いる前に、次の要件に注意してください。



- クラスタ単位のリブートが必要です。
- システム構成ファイルは注意深く編集する必要があります。これらのファイルの中にはコンテキスト依存シンボリック・リンク (CDSL) のものがあります。各クラスタ・メンバのターゲット・ファイルを編集する必要があり、ファイルをコピーするときには `mv` よりむしろ `cp` を使用するようにしてください。 `mv` を使うと、CDSL のターゲットにファイルをコピーせず CDSL に上書きします。
- ファイルを編集またはコピーするときには、クラスタをブートして形成する際にクラスタ・メンバが停止するような間違いをする可能性があります。更新したファイルを正しい位置に置いてクラスタを停止させる前に、編集結果を 2 度確認してください。
- IP アドレスを変更すると、そのアドレスと関連付けられているサブネット・マスクも変更する必要があります。
- この節で説明する表と手順では、ホスト名と IP アドレスを含む最も一般的なシステム構成ファイルを扱っています。ホスト名、クラスタ別名、インタフェース IP 別名、またはサブネット・マスクを含む、CAA (cluster application availability) スクリプトまたはネットワーク・リソースのような、他のすべてのファイルを見直して、編集する必要があります。

クラスタを別の IP サブネットに移動するには、次の手順を実行します。

1. 移動のために必要となる IP 名とアドレスを取得します。この情報を記録するために 5.10.2 項の表を使用します。移動のために必要なサブネット・マスクのすべての変更を記録します。移動した結果、クラスタが別のネームサーバを使う場合、`/etc/resolv.conf` ファイルに加える変更を記録します。
2. この移動が物理的なネットワーク接続の変更を要する場合、新しい物理的なネットワーク接続が適切で、使用できる状態にあることを確認します。
3. いつ移動を実行するかを、ユーザ、他のクラスタ、およびネットワーク管理者に知らせます。他のシステムやクラスタが変更を計画している IP 名とアドレスに依存している場合、それらの管理者と連携をとる必要があります。たとえば、NIS、DNS、またはメール・サーバが影響を受けます。クラスタが、停止してはならないサービスを実行中であれば、クラスタがシャットダウンしている間にサービスを実行する、別のシステムやクラスタを準備します。

4. 現在のクラスタの IP 名とアドレスが共通のシステム構成ファイル中で使用されている場所を調べます。これを実行する 1 つの方法は、`clu_get_info` を使用し、現在のクラスタの情報を取得した後、`grep` を使用して共通のシステム構成ファイル内で情報を探します。
5. ホスト名、IP アドレス、クラスタ別名、インタフェース IP 別名、名前の別名、またはサブネット・マスクを含む可能性のある CAA スクリプト・ファイルのようなすべてのファイルを探します。
6. 更新しようとする構成ファイルの保存用、作業用の両方のファイルを作成します。

---

注意

---

CAA スクリプトやサイト固有のスクリプトが変更対象の IP アドレス、ホスト名、別名、またはサブネット・マスクを参照する場合、それらのファイルのコピーを作り、それらに加えた変更の履歴を残しておいてください。

---

7. 計画している変更該当する表を使用します。
  - 外部 IP アドレスだけの変更 (5.10.1.1 項)
  - 外部 IP アドレスとホスト名の変更 (5.10.1.2 項)
  - 外部および内部 IP アドレスとホスト名の変更 (5.10.1.3 項)
- a. 表の情報を使用して、原本ではなくコピーの作業ファイルを編集します。CDSL では、メンバ固有のディレクトリ内の作業ファイルを編集することを覚えておいてください。

---

注意

---

`sysman` を使うか、またはエディタを使うかは自由です。正しく編集することが重要です。作業ファイルから原本ファイルの位置へコピーし、すべてのクラスタ・メンバを停止する前に、情報が正しいことを 100 パーセント確実にする必要があります。

作業ファイルを編集する場合、新しいサブネット上のサブネット・マスクが現在のサブネット用のものと同

じでないなら、それらを編集することを覚えておいてください。

`/.rhosts` や `/etc/hosts.equiv` 作業ファイルを編集する場合、新しいホスト名を追加しますが、そこに現在リストされているものを削除しないでください。新しいサブネットでクラスタが正常にブートできた後、旧ホスト名を削除してください。

- 
- b. 編集をすべて終えた後で、作業ファイルと原本ファイルの内容を比較します。編集内容を見直してください。
8. すべての編集内容の確認がとれた後、次のことを実施します。
- a. 影響を受ける全員にメッセージを送ります。いつクラスタへのアクセスが無効になる予定か、いつ実際の移動を開始するのかを知らせます。
  - b. ログインとすべての外部ネットワーク・インタフェースを無効にします。たとえば、`wall -c` コマンドを使って、ユーザにすぐにログインが無効になることを知らせ、`touch` コマンドを使って、`/etc/nologin` ファイルを作成し、各メンバで `rcinet stop` コマンドを使ってそのメンバ上のネットワークを停止させます。
  - c. 原本のファイル位置に作業ファイルをコピーします。編集後のファイルで置き換えた各ファイルの履歴を残しておいてください(更新した他のすべてのファイルの履歴も残しておいてください)。
  - d. 編集した構成ファイルがすべて適切な位置に置かれた後で、すべての情報が正しいことを確認します。クラスタをシャットダウンした後のリブートの成否は編集結果によります。移動を必要とした変更範囲を考慮して、次の項目のうちいくつかあるいは全部が正しく変更されていることを確認します。
    - IP アドレス、インタフェース IP 別名、およびサブネット・マスク
    - ホスト名と名前の別名
    - 省略時のクラスタ別名とその他のクラスタ別名

9. クラスタの各メンバを停止します。たとえば，各メンバ上で次のコマンドを実行します。

```
# shutdown -h now
```

注意

shutdown -c コマンドは，編集対象ファイルから影響を受けるため，それぞれのメンバを停止させる必要があります（編集ファイルが所定位置に戻された後で clu\_get\_info -full を実行すると，表示される情報のうちいくつかは，編集結果を反映しています）。

10. クラスタが停止したら，ネットワーク接続への必要な変更を行います。
11. クラスタをブートします。
12. 5.10.3 項の情報を使って，すべてが期待通りに動作しているかどうかを調べます。
13. /.rhosts と /etc/hosts.equiv から無効になったエントリを削除します。

5.10.1 ファイル編集に関する表

この項には，種々の移動用に，一般的なシステム構成ファイルに加える必要のある項目をリストした表があります。

- 外部 IP アドレスだけの変更 ( 5.10.1.1 項)
- 外部 IP アドレスとホスト名の変更 ( 5.10.1.2 項)
- 外部および内部 IP アドレスとホスト名の変更 ( 5.10.1.3 項)

5.10.1.1 外部 IP アドレスだけの変更

一般的な構成ファイルで，クラスタの外部 IP アドレスを変更する場合に必要なとなる編集項目を示します。

ファイル	編集項目
共用ファイル	
/etc/hosts	各クラスタの外部インタフェースとクラスタ別名に関連する IP アドレス。

ファイル	編集項目
/etc/networks	定義されていれば、ネットワークのアドレス。
/etc/resolv.conf	クラスタが新しいサブネット上で別のネームサーバを使用する場合には、それらのアドレス。
<b>CDSL</b>	
/etc/clu_alias.config	各メンバに対して、IP アドレス (可能であれば、サブネット・マスクも含む) によって定義された別名 (おそらく、DEFAULTALIAS エントリは変更不要)。
/etc/inet.local	各メンバに対して、このファイルをインタフェース IP 別名を構成するために使用する場合。
/etc/ntp.conf	各メンバに対して、IP アドレスをサーバやピア・システムのエントリで変更する場合。
/etc/rc.config	各メンバに対して、IFCONFIG エントリ。必要なら、サブネット・マスクを変更する。
/etc/routes	各メンバに対して、ルートが定義されている場合。必要なら、サブネット・マスクを変更する。
<b>メンバ固有 (非 CDSL)</b>	
/etc/gated.conf.membern	各メンバに対して、IP アドレスを変更する必要があるエントリ。CLUAMGR_ROUTE_ARGS がメンバの rc.config ファイルで nogated に設定されている場合、メンバの /etc/gated.conf (CDSL) ファイルを変更する。
/cluster/admin/.membern.cfg	これらのファイルを使用する場合、IP アドレスへの変更を更新する (使用しない場合、これらのファイルを clu_create や clu_add_member で使うときには、旧の値に戻す)。

#### 5.10.1.2 外部 IP アドレスとホスト名の変更

一般的な構成ファイルで、クラスタの外部 IP アドレスとホスト名を変更する必要がある編集項目を示します。

ファイル	編集項目
<b>共用ファイル</b>	
/.rhosts	新しいクラスタ名を追加する。エントリを削除しない。
/etc/cfgmgr.auth	メンバのホスト名。
/etc/hosts	クラスタの外部インタフェースとクラスタ別名に関連する IP アドレス、ホスト名、および別名。

ファイル	編集項目
/etc/hosts.equiv	新しいクラスタ名を追加する。エントリを削除しない。
/etc/networks	定義されている場合、ネットワークのアドレス。
/etc/resolv.conf	新しいサブネット上でドメイン名やネームサーバを変更した場合。
<b>CDSL</b>	
/etc/clu_alias.config	各メンバに対して、ホスト名や IP アドレス (可能であれば、サブネット・マスクも含む) によって定義された別名 (おそらく、DEFAULTALIAS エントリは変更不要)。
/etc/inet.local	各メンバに対して、このファイルをインタフェース IP 別名を構成するために使用する場合。
/etc/ntp.conf	各メンバに対して、IP アドレスとホスト名をサーバやピア・システムのエントリで変更する場合。
/etc/rc.config	各メンバに対して、HOSTNAME、IFCONFIG、および CLUSTER_NET エントリ。必要なら、サブネット・マスクを変更する。
/etc/routes	各メンバに対して、ルートが定義されている場合。必要なら、サブネット・マスクを変更する。
/etc/sysconfigtab	各メンバに対して、cluster_name と cluster_node_name。
<b>メンバ固有 (非 CDSL)</b>	
/etc/gated.conf.membern	IP アドレスを変更する必要があるエントリ。CLUAMGR_ROUTE_ARGS がメンバの rc.config ファイルで nogated に設定されている場合には、メンバの /etc/gated.conf (CDSL) ファイルを変更する。
/cluster/admin/.membern.cfg	これらのファイルを使用する場合、クラスタ名、ホスト名、IP アドレスへの変更を更新する (使用しない場合、これらのファイルを clu_create や clu_add_member で使うときには、旧の値に戻す)。

### 5.10.1.3 外部および内部 IP アドレスとホスト名の変更

一般的な構成ファイルで、クラスタの外部および内部 IP アドレスとホスト名を変更する必要がある編集項目を示します。

ファイル	編集項目
共用ファイル	
<code>/.rhosts</code>	クラスタ・インターコネクトに関連する新しいクラスタ名と新しいホスト名を追加する。バージョン 5.0A から 5.1 では、 <code>*-mc0</code> 。バージョン 5.1A 以降では、 <code>*-ics0</code> 。エントリを削除しない。
<code>/etc/cfgmgr.auth</code>	メンバのホスト名。
<code>/etc/hosts</code>	クラスタの外部インタフェース、クラスタ別名およびクラスタ・インターコネクトのインタフェースに関連する IP アドレス、ホスト名、および別名。
<code>/etc/hosts.equiv</code>	クラスタ・インターコネクトに関連する新しいクラスタ名と新しいホスト名を追加する。バージョン 5.0A から 5.1 では、 <code>*-mc0</code> 。バージョン 5.1A 以降では、 <code>*-ics0</code> 。エントリを削除しない。
<code>/etc/networks</code>	定義されていれば、ネットワークのアドレス。
<code>/etc/resolv.conf</code>	新しいサブネット上でドメイン名やネームサーバを変更した場合。
CDSL	
<code>/etc/clu_alias.config</code>	各メンバに対して、ホスト名や IP アドレス (可能であれば、サブネット・マスクも含む) によって定義された別名 (おそらく、 <code>DEFAULTALIAS</code> エントリは変更不要)。
<code>/etc/ifaceaccess.conf</code>	クラスタ・インターコネクトに関連する IP アドレス。必要なら、サブネット・マスクを変更する。
<code>/etc/inet.local</code>	各メンバに対して、このファイルをインタフェース IP 別名を構成するために使用する場合。
<code>/etc/ntp.conf</code>	各メンバに対して、IP アドレスやホスト名をサーバやピア・システムのエントリで変更する場合 (クラスタ・インターコネクトに関連する名前を変更する場合、それらのピア名の変更を確実に行う)。
<code>/etc/rc.config</code>	各メンバに対して、 <code>HOSTNAME</code> 、 <code>IFCONFIG</code> 、および <code>CLUSTER_NET</code> エントリ。必要なら、サブネット・マスクを変更する。
<code>/etc/routes</code>	各メンバに対して、ルートが定義されている場合。必要なら、サブネット・マスクを変更する。
<code>/etc/sysconfigtab</code>	各メンバに対して、 <code>cluster_name</code> 、 <code>cluster_node_name</code> 、および <code>cluster_node_inter_name</code> の値。
メンバ固有 (非 CDSL)	

ファイル	編集項目
/etc/gated.conf.membern	各メンバに対して、IP アドレスを変更する必要があるエントリ。CLUAMGR_ROUTE_ARGS がメンバの rc.config ファイルで nogated に設定されている場合、メンバの /etc/gated.conf (CDSL) ファイルを変更する。
/cluster/admin/.membern.cfg	これらのファイルを使用する場合、クラスタ名、ホスト名、クラスタ・インターコネクト名、および IP アドレスへの変更を更新する (使用しない場合、これらのファイルを clu_create や clu_add_member で使うときには、旧の値に戻す)。

## 5.10.2 属性のチェックリスト表

この項の表を使用して、新しいサブネットにクラスタを移動するのに必要な IP 名とアドレスの情報を記録してください。5 つ以上のクラスタ・メンバ、または 4 つ以上のクラスタ別名がある場合、必要に応じて、該当する表をコピーし、行に名前を書き込んでください。

### 5.10.2.1 外部ホスト名と IP アドレス

メンバ	属性	値	
メンバ 1	ホスト名	旧	
		新	
	IP アドレス (およびサブネット・マスク)	旧	
		新	
メンバ 2	ホスト名	旧	
		新	
	IP アドレス (およびサブネット・マスク)	旧	
		新	
メンバ 3	ホスト名	旧	
		新	
	IP アドレス (およびサブネット・マスク)	旧	
		新	



メンバ	属性	値	
メンバ 4	ホスト名	旧	
		新	
	IP アドレス (およびサブネット・マスク)	旧	
		新	

### 5.10.2.2 クラスタ名とクラスタ別名

クラスタ別名	値	
完全に修飾されたクラスタ名 (クラスタ名は省略時のクラスタ名)	旧	
	新	
省略時クラスタ別名 IP アドレス(およびサブネット・マスク)	旧	
	新	
追加のクラスタ別名 #1	旧	
	新	
追加のクラスタ別名 #1 の IP アドレス (およびサブネット・マスク)	旧	
	新	
追加のクラスタ別名 #2	旧	
	新	
追加のクラスタ別名 #2 の IP アドレス (およびサブネット・マスク)	旧	
	新	

### 5.10.2.3 インタフェース IP 別名

メンバ	属性	値	
メンバ 1	IP 別名 #1 (およびサブネット・マスク)	旧	
		新	
	IP 別名 #2 (およびサブネット・マスク)	旧	
		新	

メンバ	属性	値	
メンバ 2	IP 別名 #1 (およびサブネット・マスク)	旧	
		新	
	IP 別名 #2 (およびサブネット・マスク)	旧	
		新	
メンバ 3	IP 別名 #1 (およびサブネット・マスク)	旧	
		新	
	IP 別名 #2 (およびサブネット・マスク)	旧	
		新	
メンバ 4	IP 別名 #1 (およびサブネット・マスク)	旧	
		新	
	IP 別名 #2 (およびサブネット・マスク)	旧	
		新	

#### 5.10.2.4 外部サーバ

新しいサブネット上の BIND , NIS , または NTP のようなネットワーク・サービスに別のサーバを使用する場合 , 次の表の中にこれらのサービスで使用する IP アドレスの新旧を記録してください。

サーバ	IP アドレス	
	旧	
	新	
	旧	
	新	
	旧	
	新	
	旧	
	新	

### 5.10.2.5 チェックリスト

チェックリストの状況欄に、構成ファイルの作業用コピーへの編集の履歴を記録してください。

ファイル	状況	
共用ファイル		
/.rhosts		
/etc/cfgmgr.auth		
/etc/hosts		
/etc/hosts.equiv		
/etc/networks		
/etc/resolv.conf		
CDSL		
/etc/clu_alias.config	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/etc/ifaccess.conf	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/etc/inet.local	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/etc/ntp.conf	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	

ファイル	状況	
/etc/rc.config	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/etc/routes	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/etc/sysconfigtab	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
メンバ固有 (非 CDSL)		
/etc/gated.conf.membern	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	
/cluster/admin/.membern.cfg	メンバ 1	
	メンバ 2	
	メンバ 3	
	メンバ 4	

### 5.10.3 正常性の検証

クラスタを別の IP サブネットへ移動するための手順を適用した後に、正常に実行されたかどうかを検証します。

編集結果がすべて正しく、編集されたファイルが適切な場所に置かれている場合、システムをブートし、クラスタを形成して、新しい稼働環境になります。次のコマンドを使用して、クラスタとそのサブシステムが正しく動作していることを検証します。

```
# hostname
# ifconfig -a
# netstat -i
# clu_get_info -full | more
# cluamgr -s all
# caa_stat
```

ping , rlogin , および rpcinfo のような標準的なネットワーク用のコマンドを使用しても、クラスタ・メンバが動作状態であるかどうか、ログインが受け付けられるかどうか、他のシステムと通信できるかどうかを検証できます。

手順が正常に終了しなかった場合、問題の特定と解決についての情報は 5.10.4 項を参照してください。

5.10.4   トラブルシューティング

5.10.3 項に説明されているように、手順が正常に終了しなかった場合は、次の表を使用して問題の特定と解決を図ってください。

問題	解決策
メンバがブートできない。	<p>障害がブート・パスの初期段階で発生した場合、おそらく sysconfigtab ファイルの編集に間違いがある。</p> <p>1 つのメンバがブートできる場合、ブートできないメンバの boot_partition をマウントし、編集結果を修正する。</p> <p>どのメンバもブートできない場合、まず、どのメンバもクォーラムが得られず失敗しているのかどうかを調べる。その場合、1 つのメンバに対して対話形式のブートを実行し、期待ボートの値をゼロに設定する。そのメンバをブートして、他のメンバのファイルを修正し、それらのメンバをブートした後、クォーラムの期待ボートを再度調整する。</p> <p>どのメンバも対話形式でブートできない場合は、Tru64 UNIX オペレーティング・システムをブートし、クラスタの最初のメンバのブート・パーティションをマウントしてそのメンバへの編集結果を修正し、Tru64 UNIX システムを停止したのちに、そのクラスタ・メンバを(必要であれば、対話形式で)ブートして、残りのメンバの編集結果を修正する。</p>

問題	解決策
クラスタはブートするがマルチユーザ・モードでネットワークに問題が発生する。	問題を特定し、どのメンバのどのファイルで最も誤った編集があり得るかを判断する。編集結果を修正し、それらのシステム上のネットワーク・サービスを停止して、再開する。
上記の解決策でも、正常にならない。	原本ファイルの保存用コピーで復元する。旧のネットワーク接続を復元する。何が異常かを調査している間、クラスタがクライアントにサービスを続けられるように、旧サブネット上でクラスタをブートする。

## 5.11 クラスタ名または IP アドレスの変更

この節では、クラスタ名や IP アドレスの変更の方法について説明します。クラスタの名前は省略時のクラスタ別名でもあるので、クラスタ名を変更すると省略時のクラスタ別名も変わります。

クラスタの名前を変更するには、クラスタ全体のシャットダウンとリブートが必要です。クラスタの IP アドレスを変更するには、個々のメンバそれぞれのシャットダウンとリブートが必要です。

### 5.11.1 クラスタ名の変更

クラスタの名前を変更するには、次の手順を注意して実行します。間違えると、クラスタがブートできなくなるおそれがあります。

1. `clubase` サブシステム・スタンザ・エントリに対する `cluster_name` 属性を新しくしたファイルを作成します。たとえば、クラスタ名を `deli` に変更するには、次の `clubase` サブシステム・スタンザ・エントリに名前を追加します。

```
clubase:
  cluster_name=deli
```

#### 注意

作成したファイルのすべての行末に改行 (LF) があることを確認してください。改行がなければ、`sysconfigtab` ファイルを変更した場合に、同じ行に属性が 2 つあることになり、システムがブートできなくなるおそれがあります。

クラスタのルート・ディレクトリにファイルを作成すれば、そのファイルをコピーしなくても、クラスタ内のすべてのシステムで使用できるようになります。

2. 各クラスタ・メンバで、新しく作成したファイルにある clubase サブシステム属性と /etc/sysconfig ファイルにある clubase サブシステム属性を、`sysconfigdb -m -f file clubase` コマンドを使用してマージします。

たとえば、`cluster-name-change` ファイルに手順 1 の例で示した情報が入っているものとします。`cluster-name-change` ファイルを使用してクラスタ名を `poach` から `deli` に変更するには、次のコマンドを使用します。

```
# sysconfigdb -m -f cluster-name-change clubase
Warning: duplicate attribute in clubase:
was cluster_name = poach, now cluster_name = deli
```

---

#### 警告

変更する属性が 1 つまたは 2 つしか入っていないファイルを指定して `sysconfigdb -u` コマンドを実行しないでください。`-u` フラグの指定により、サブシステム・エントリ (`clubase` など) が入力ファイルのサブシステム・エントリに置き換わります。そのため、`clubase` サブシステムに `cluster_name` 属性だけを指定してこのコマンドを実行すると、新しい `clubase` サブシステムの内容は `cluster_name` 属性だけになり、他の必要な属性が消えてしまいます。

3. 次の各ファイルでクラスタ名を変更します。
  - /etc/hosts
  - /etc/hosts.equiv
4. 新しいクラスタ名を `.rhosts` ファイル (すべてのクラスタ・メンバに共通) に追加します。

ファイル内の現在のクラスタ名は残しておきます。現在の名前は、次の手順で `shutdown -c` コマンドを実行する際に必要です。

必要に応じて、クライアントの `.rhosts` ファイルを変更します。

5. `shutdown -c` コマンドを使用してクラスタ全体をシャットダウンし、クラスタ内の各システムをリブートします。
6. `/.rhosts` ファイルから以前のクラスタ名を削除します。
7. 次のように `/usr/sbin/clu_get_info` コマンドを実行して、クラスタ名が変更されたことを確認します。

```
# /usr/sbin/clu_get_info
Cluster information for cluster deli
:
```

### 5.11.2 クラスタの IP アドレスの変更

クラスタの IP アドレスは、次の手順で変更します。

1. `/etc/hosts` ファイルを編集して、クラスタの IP アドレスを変更します。
2. 一度に 1 つずつ (クォーラムを維持するため)、クラスタ・メンバ・システムをシャットダウンしてリブートします。

クラスタに現在入っていないシステムで `/usr/sbin/ping` コマンドを実行し、クラスタの IP アドレスが変更されたことを確認します。そのクラスタのアドレスを使用したときにクラスタからエコーが返ってくれば、変更されています。

```
# /usr/sbin/ping -c 3 16.160.160.160
PING 16.160.160.160 (16.160.160.160): 56 data bytes
64 bytes from 16.160.160.160: icmp_seq=0 ttl=64 time=26 ms
64 bytes from 16.160.160.160: icmp_seq=1 ttl=64 time=0 ms
64 bytes from 16.160.160.160: icmp_seq=2 ttl=64 time=0 ms

----16.160.160.160 PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/9/26 ms
```

## 5.12 メンバ名、IP アドレス、クラスタのインターコネクト・アドレスの変更

メンバ名、メンバ IP アドレス、またはクラスタのインターコネクト・アドレスを変更するには、クラスタからメンバを削除し、新しいメンバ名やアドレスを指定して元に戻します。これを行う前に、クラスタがクォーラ



ムを失っていないことと、CAA の `restricted` 配置ポリシーが影響していないことを確認します。

### 5.12.1 メンバ削除前のクォーラムのチェック

メンバを削除する前に、クラスタのメンバを削除しても問題なく稼働できるように十分な投票メンバ (クォーラム・ディスク・ポートを含む) がクラスタで動作していることを確認します。シングル・クラスタ・メンバのシャットダウンについての情報は、5.5 節を参照してください。

### 5.12.2 メンバ削除前の CAA の `restricted` 配置ポリシーのチェック

CAA プロファイルで `restricted` 配置ポリシーを使っているかどうかを調べる必要があります。その場合、`HOSTING_MEMBERS` リソースが、変更したいメンバ・システムの名前だけを含むかどうかを調べる必要があります。

`/usr/sbin/caa_profile -print` コマンドを使用して、CAA プロファイルを表示します。アプリケーションの `PLACEMENT` リソースがアプリケーションに対して制限 (`PLACEMENT=restricted`) されている場合、および `HOSTING_MEMBERS` リソースに名前やアドレスを変更するメンバの名前だけが含まれる場合には、次のことを行います。

1. アプリケーション・リソース・プロファイルを更新して、このアプリケーションを実行できる別のメンバ・システムをメンバ・リストに追加します。たとえば、`HOSTING_MEMBERS` リソースでメンバの *provolone* がアプリケーションの実行を制限していることを示している場合、`HOSTING_MEMBERS` リソースに *pepicelli* を追加します。この追加は、次のコマンドを実行します。

```
# /usr/sbin/caa_profile -update resource_name -h provolone pepicelli
```

---

#### 注意

---

名前やアドレスを変更するシステムの名前を削除しないでください。

2. クラスタ・メンバ間での不整合を防ぐために、既存の CAA レジストリ・エントリを最新のリソース・プロファイルで更新します。次のようにコマンドを実行します。

```
# /usr/sbin/caa_register resource_name -u
```

3. HOSTING\_MEMBERS リソースに追加されたシステムに、次のコマンドでアプリケーションを再配置します。

```
# /usr/sbin/caa_relocate resource_name -c pepicelli
```

### 5.12.3 メンバの削除と追加

次の手順に従って、既存のライセンス・データを保存し、メンバを削除および追加して、ライセンス・データを復元します。

1. 変更したいメンバ名、メンバ IP アドレス、またはクラスタ・インターコネクト・アドレスがあるメンバ・システムにログインします。
2. lmf コーティリティを使用し、システム上で実行を許可されている製品の PAK (product authorization key) を再構成します。次の例にある KornShell スクリプトは、再構成されたすべての PAK を /licenses ディクショナリに格納します。

```
# mkdir /licenses
# for i in `lmf list | grep -v Product | awk '{print $1}'`
do
lmf issue /licenses/${i}.license $i
done
```

3. メンバを停止します。1 つのクラスタ・メンバをシャットダウンする方法については、5.5 節を参照してください。
4. クラスタのアクティブ・メンバ上で、シャットダウンしたばかりのメンバを削除します。これは、コマンド `clu_delete_member` を実行して行います。

```
# clu_delete_member -m memberid
```

削除するメンバのメンバ ID を知るには、コマンド `clu_get_info` を使います。

`clu_delete_member` の使い方については、5.8 節を参照してください。

5. コマンド `clu_add_member` を使って、必要なメンバ名、メンバ IP アドレス、クラスタのインターコネクト・アドレスを指定し、クラスタにシステムを追加して元に戻します。

クラスタにメンバを追加する方法についての詳細は、TruCluster Server 『クラスタ・インストレーション・ガイド』を参照してください。

6. 新しく再インストールされたブート・ディスクから `genvmunix` をブートすると、新しいメンバが自動的にサブセットを構成し、カスタマイズされたカーネルを構築して、マルチユーザ・モードになります。次のように、ログインして保存されたライセンスを登録します。

```
# for i in /licenses/*.license
do
lmf register - < $i
done
# lmf reset
```

7. システムをリブートし、カスタマイズされたクラスタ・カーネルで動作させます。

```
# shutdown -r now
```
8. アプリケーションへの配置ポリシーが `avored` か `restricted` であり、クラスタ・メンバ・システムで名前が変更されていて、そのアプリケーションに対する `HOSTING_MEMBERS` リソースにリストされている場合、次のように、リソースの古い名前を削除して新しい名前を追加します。

- a. `HOSTING_MEMBERS` リソースを変更し、古い名前を削除して新しい名前を追加します。
- b. 最新のリソース・プロファイルで、既存の CAA レジストリを次のように更新します。

```
# /usr/sbin/caa_register resource_name -u
```

## 5.13 ソフトウェア・ライセンスの管理

クラスタに新規メンバを追加したら、そのメンバ上で実行するアプリケーションに関して、そのメンバ上でのアプリケーション・ライセンスを登録しなければなりません。

クラスタへの新規メンバの追加と Tru64 UNIX のライセンス登録については、TruCluster Server 『クラスタ・インストール・ガイド』のメンバの追加に関する章を参照してください。

## 5.14 レイヤード・アプリケーションのインストールと削除

アプリケーションのインストールや削除の手順は、通常、クラスタとスタンドアロン・システムのどちらも同じです。クラスタでは、アプリケーションは 1 回のインストールでかまいません。ただし、一部のアプリケーションでは、さらに手順を要します。

- アプリケーションのインストール

アプリケーションにメンバ固有の構成要件が適用される場合は、アプリケーションが動作する各メンバにログインしてアプリケーションを構成する必要があります。詳細は、アプリケーションの構成マニュアルを参照してください。

- アプリケーションの削除

setld を使用してアプリケーションを削除する前に、そのアプリケーションが実行中でないことを確認してください。確認するには、いくつかのメンバ上のアプリケーションを停止させる必要があるかもしれません。たとえば、マルチ・インスタンス・アプリケーションの場合、アプリケーションを停止させると複数のクラスタ・メンバ上で起動中のデーモンを強制終了させてしまう可能性があります。

CAA で管理されているアプリケーションの場合は、次のコマンドを使用して高可用性アプリケーションの状態を確認します。

```
# caa_stat
```

削除対象のアプリケーションが実行中 (STATE=ONLINE) の場合は、それを停止して、次のコマンドで CAA レジストリから削除します。

```
# caa_stop application_name  
# caa_unregister application_name
```

アプリケーションが停止すれば、それを setld コマンドで削除します。そのアプリケーションのマニュアルに指示が記載されている場合は、それに従ってください。アプリケーションが現在利用できないメンバ上にインストールされている場合は、そのメンバがクラスタに参加すると、アプリケーションは利用できないメンバから自動的に削除されます。

## 5.15 課金サービスの管理

システムの課金サービスは、クラスタ対応ではありません。このサービスでは、メンバ固有のファイルとデータベースが利用されます。このため、クラスタで課金サービスを利用するには、メンバ単位にサービスの設定や管理を行う必要があります。

/usr/sbin/acct ディレクトリは CDSL です。/usr/sbin/acct における課金サービス・ファイルは各クラスタ・メンバに固有です。

クラスタ上で課金サービスを設定するには、Tru64 UNIX 『システム管理ガイド』のシステムの課金サービスの管理に関する章に記載されている手順に、以下のような変更が加えられます。

1. すべてのクラスタ・メンバ上で課金サービスを有効にするには、各メンバ上で次のコマンドを入力します。

```
# rcmgr -c set ACCOUNTING YES
```

特定のメンバ上でのみ課金サービスを有効にする場合は、rcmgr コマンドに -h オプションを使用します。たとえば、メンバ 2, 3, および 6 で課金サービスを有効にするには、以下のコマンドを入力します。

```
# rcmgr -h 2 set ACCOUNTING YES
# rcmgr -h 3 set ACCOUNTING YES
# rcmgr -h 6 set ACCOUNTING YES
```

2. 各メンバ上で課金サービスを起動する必要があります。課金サービスを起動したい各メンバにログインして、次のコマンドを入力します。

```
# /usr/sbin/acct/startup
```

メンバ上の課金サービスを停止するには、そのメンバにログインして、コマンド `/usr/sbin/acct/shutacct` を実行する必要があります。

`/usr/spool/cron` ディレクトリは CDSL です。また、このディレクトリ内のファイルはメンバに固有であるため、そのファイルを使用して課金サービスをメンバ単位に調整できます。課金サービスを調整するには、課金サービスの実行対象の各メンバにログインします。必要に応じて、`crontab` コマンドを使用して `crontab` ファイルを変更します。詳細は、Tru64 UNIX 『システム管理ガイド』のシステムの課金サービスの管理に関する章を参照してください。

`/usr/sbin/acct/holidays` ファイルは CDSL です。このため、メンバ単位に課金サービスの休止予定を設定します。

課金サービスの詳細については、`acct(8)` を参照してください。



---

## クラスタ内のネットワーク管理

この章では次の項目について説明します。

- ネットワーク・インタフェースの変更時の `ifaccess.conf` の更新 (6.1 節)
- ネットワーク・インタフェースのフェイルオーバー (6.2 節)
- IP ルータの実行 (6.3 節)
- ネットワークの構成 (6.4 節)

スタンドアロン・システムのネットワーク管理については、Tru64 UNIX の『ネットワーク管理ガイド：接続編』および『ネットワーク管理ガイド：サービス編』を参照してください。

### 6.1 ネットワーク・インタフェースの変更時の `ifaccess.conf` の更新

新たにネットワーク・インタフェースを追加する場合や既存のインタフェースを変更または置換する場合に、変更が生じたクラスタ・メンバの `/etc/ifaccess.conf` ファイルを更新する必要があります。この目的は、信頼できないサブネットからのクラスタ・インターコネクต์へのアクセスを拒否するためです。

Memory Channel インターコネクต์を使用するクラスタでは、クラスタ・インターコネクต์の仮想インタフェースのアドレス (`ics0`) を追加する必要があります。LAN インターコネクต์を使用するクラスタでは、クラスタ・インターコネクต์の仮想インタフェースのアドレスとクラスタ・インターコネクต์の物理インタフェースのアドレスとを別々の行で追加する必要があります。

省略時には、インストール・プログラムはクラスタ・インターコネクต์仮想インタフェースに 10.0.0 サブネット IP アドレスを設定します。この IP アドレスのホスト部分には、メンバ ID が設定されます。LAN インターコネクต์では、インストール・プログラムはクラスタ・インターコネクต์物理インタ

フェースに 10.1.0 サブネット IP アドレスを設定します。この IP アドレスのホスト部分には、メンバ ID が設定されます。

次のように変更します。

1. ネットワーク・インタフェースが変更されたメンバにログインします。
2. `ifconfig` コマンドを使って、ネットワーク・インタフェースの名前を調べます。例を次に示します。

```
# ifconfig -l
ics0 lo0 sl0 ee0 ee1 ee2 tu0 tun0
```

3. メンバが使用するクラスタ・インターコネクトのアドレスを調べます。メンバの `/etc/ifaccess.conf` ファイルの `10.1.xxx.xxx` と `10.0.xxx.xxx` エントリを調べます。

また、`ifconfig` コマンドを使って、メンバが使用するクラスタ・インターコネクトのアドレスを調べて、クラスタ・インターコネクトのネットワーク・インタフェースを識別することもできます。コマンドの出力は、使用するインターコネクトによって変わります。次の例は、Memory Channel からの出力を示しています。

```
# ifconfig -a | grep -p CLUIF
ics0: flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
      inet 10.0.0.2 netmask ffffffff00 broadcast 10.0.0.255 ipmtu 7000
```

次の例は、LAN からの出力を示しています。

```
# ifconfig -a | grep -p CLUIF
alt0: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
      inet 10.1.0.20 netmask ffffffff00 broadcast 10.1.0.255 ipmtu 1500

ics0: flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
      inet 10.0.0.20 netmask ffffffff00 broadcast 10.0.0.255 ipmtu 7000
```

4. 変更された各インタフェースに応じて、`/etc/ifaccess.conf` ファイルを適切に編集します。次の形式を使用して入力します。

```
interface-name interconnect_net_address 255.255.255.0 deny
```

この行は、「`interconnect_net_address` を持つホストから `interface-name` へのすべてのパケットを拒否する」ことを意味しています。マスクは、`interconnect_net_address` のビット・ポジションで意味のあるビットを 1 にします。

たとえば、クラスタ・インターコネクトの仮想インタフェース・アドレスが 10.0.0.1 で新しい `tu0` インタフェース・カードがクラスタに追加された場合は、`/etc/ifaccess.conf` に次の行を追加します。

```
tu0 10.0.0.1 255.255.255.0 deny
```



この行は、「10.0.0 を持つホストからの tu0 へのすべてのパケットを拒否する」ことを意味しています。

クラスタが LAN インターコネクトを使用している場合、クラスタ・インターコネクトの物理インタフェースのアドレスの 1 行をさらに追加する必要があります。クラスタ・インターコネクトの物理インタフェースのアドレスを 10.1.0.1 と仮定すると、仮想ネットワーク・アドレス用の行のほかに、次の行を追加する必要があります。

```
tu0 10.1.0.1 255.255.255.0 deny
```

---

#### 注意

---

クラスタ・インターコネクトの仮想インタフェース (ics0) とクラスタ・インターコネクトの物理インタフェースへのパケットを拒否するようなエントリを追加しないでください。これらのインタフェースはクラスタ・インターコネクトでアクセスできなければなりません。たとえば、クラスタ・インターコネクトのメンバによって使用されるネットワーク・インタフェースが alt0 であれば、次のようなエントリはクラスタ・インターコネクトからのアクセスを拒否します。  
/etc/ifaccess.conf ファイルに指定してはなりません。

```
alt0 10.1.0.0 255.255.255.0 deny
```

詳細は、ifaccess.conf(4) を参照してください。

## 6.2 ネットワーク・インタフェースのフェイルオーバー

NetRAIN (Redundant Array of Independent Network Adapters) インタフェースによって、ネットワークの接続に関するある種の障害からクラスタを保護できます。NetRAIN は、同じ LAN セグメント上の複数のネットワーク・インタフェースを NetRAIN セットという 1 つの仮想インタフェースに統合します。このセット内の 1 つのネットワーク・インタフェースが稼働状態になり、その他のインタフェースはアイドル状態になります。稼働状態のインタフェースに障害が発生した場合、NetRAIN セット内のアイドル状態のインタフェースの 1 つが同じ IP アドレスでオンラインになります。

NIFF (Network Interface Failure Finder) は、これらのネットワーク・インタフェースの状態を監視し、ネットワークの障害の徴候を報告する追加の

機能です。NIFF を使用すれば、NetRAIN デバイスなどのネットワーク・デバイスに障害が発生したときに、イベントが生成されるようにすることができます。これらのイベントの監視によって、障害発生時に適切な対処策をとることができます。

NIFF デーモン `niffd` はクラスタ内で自動的に起動されます。

ネットワーク・リソースに依存するアプリケーションのフェイルオーバーについては、TruCluster Server 『クラスタ高可用性アプリケーション・ガイド』を参照してください。

NIFF と NetRAIN についての詳しい説明は、Tru64 UNIX の『ネットワーク管理ガイド：サービス編』、『ネットワーク管理ガイド：接続編』、『`niffd(8)`』、『`niff(7)`』、および『`nr(7)`』を参照してください。`rcmgr` コマンドについての詳しい説明は、『`rcmgr(8)`』を参照してください。

## 6.3 IP ルータの実行

クラスタのメンバは IP ルータになることもできます。複数のメンバを IP ルータとして構成できます。ただしそのためには、TruCluster Server の `gated` の構成を使用する以外に方法はありません。`gated` の構成をカスタマイズして、独自のルーティング環境を実行するようにできます。たとえば、OSPF (Open Shortest Path First) のようなルーティング・プロトコルを実行することができます。

`gated` の構成をカスタマイズして有効にするには、そのメンバにログインして、次の手順を実行します。

1. `gated` が実行中の場合は、次のコマンドを入力してそれを停止します。  

```
# /sbin/init.d/gateway stop
```
2. 次のコマンドを入力します。  

```
# cluamgr -r start,nogated
```
3. `gated.conf` ファイル (または任意の名前に変更した同じ構成ファイル) を変更します。`cluamgr -r nogated,start` コマンドで作成された `/etc/gated.conf.membern` を基礎として使用し、そのファイルを編集してカスタマイズした `gated` 構成ファイルを作成します。カスタマイズした構成ファイルに `/etc/gated.conf.membern` から構成別名情報を正しくマージする必要があります。

4. 次のコマンドで、`gated` デーモンを再起動します。

```
# /sbin/init.d/gateway start
```

`cluamgr -r start,nogated` コマンドは次のようなタスクを実行します。

- メンバ固有の `gated.conf` ファイルを異なる名前で作成する。
- `gated` デーモンを起動しない。
- `gated` が動作していない場合はクラスタ別名宛のルーティングのフェイルオーバーが機能しないというコンソール警告メッセージを生成し、作成された新規 `gated` ファイルを参照する。
- イベント・マネージャ (EVM) 警告メッセージを発行する。

`gated` の構成カスタマイズ用のオプションは、経験豊富なシステム管理者専用に用意されたものであり、そのような管理者は TruCluster Server に標準の `gated.conf` を変更して、そのメンバのルーティング操作に必要なエントリを追加できます。このファイルの変更に `gated` を実行すると、そのメンバはカスタマイズされたルータとして動作します。

詳細は、`cluamgr(8)` を参照してください。

---

#### 注意

---

`cluamgr` の `nogated` は、`routed` の使用を許可するオプションではありません。

TruCluster Server バージョン 5.1B より前のリリースでは、クラスタでルーティング・デーモンとして `gated` を実行する必要がありました。3.14 節では、バージョン 5.1B のルーティング・オプションについて説明しています。

クラスタのメンバはクラスタ別名宛のルーティングのみを扱い、ネットワーク内の汎用の IP ルーティングは、この機能用に調整された汎用ルータが扱うようにすることを強くお勧めします。

---

## 6.4 ネットワークの構成

ネットワークの構成は通常、Tru64 UNIX バージョン 5.1B ソフトウェアのインストール時に行います。後でネットワークの構成の変更が必要になった場

合は、次の情報が役立ちます。sysman net\_wizard コマンドか、それと同等の netconfig コマンドを使って、次の項目を構成します。

- ネットワーク・インタフェース・カード
- 静的経路 (/etc/routes)
- ルーティング・サービス (gated , IP ルータ)
- hosts ファイル (/etc/hosts)
- hosts.equiv ファイル (/etc/hosts.equiv)
- リモート who サービス (rwhod)
- DHCP (Dynamic Host Configuration Protocol) サーバ (joind)
- networks ファイル (/etc/networks)

フォーカスを指定せずに nfsconfig コマンドを実行できます。この場合、構成はクラスタ全体に適用され、すべての構成は /etc/rc.config.common ファイルに保存されます。

コマンド行または Sysman Menu からクラスタのメンバにフォーカスを指定した場合、構成は指定したメンバのみに適用され、メンバ固有の /etc/rc.config ファイルに保存されます。

次の項目の構成には、メンバへのフォーカスの指定が必要です。

- ネットワーク・インタフェース
- ゲートウェイ・ルーティング・デーモン (gated)
- 静的経路 (/etc/routes)
- リモート who デーモン (rwhod)
- インターネット・プロトコル (IP) ルータ

ネットワーク・サービスの起動と停止にも、メンバへのフォーカスの指定が必要です。

これらの項目の構成作業にメンバへのフォーカスの指定が必要なのは、これらの項目がメンバに固有なためです。ネットワーク・サービスの再起動または停止は、クラスタ全体に対して行うとクラスタ全体が稼働不能になる可能性があります。このため、この操作にもメンバへのフォーカスの指定が必要であり、この操作は一度に1メンバずつ行います。

次の項目の構成は、クラスタ全体に対して行う必要があります。

- DHCP サーバ・デーモン
- hosts ファイル (/etc/hosts)
- hosts.equiv ファイル (/etc/hosts.equiv)
- networks ファイル (/etc/networks)

DHCP の構成については , 7.1 節を参照してください。



---

## ネットワーク・サービスの管理

TruCluster Server 『クラスタ・インストール・ガイド』では、ネットワーク・サービスの初期構成の方法について説明しています。ネットワーク・サービスの構成はクラスタの作成前に行うよう強くお勧めします。クラスタの作成後にこの構成を行うと、構成の手順が複雑になることがあります。

この章では、クラスタ作成後のネットワーク・サービスの構成手順について取り上げ、次の項目について説明します。

- DHCP の構成 ( 7.1 節)
- NIS の構成 ( 7.2 節)
- 印刷の構成 ( 7.3 節)
- DNS/BIND の構成 ( 7.4 節)
- 時刻同期の管理 ( 7.5 節)
- NFS の管理 ( 7.6 節)
- inetd の構成の管理 ( 7.7 節)
- メール・サービスの管理 ( 7.8 節)
- RIS 用のクラスタの構成 ( 7.9 節)
- X Windows アプリケーションのリモートでの表示 ( 7.10 節)

### 7.1 DHCP の構成

クラスタは高可用性の DHCP (Dynamic Host Configuration Protocol) サーバとして構成できますが、DHCP クライアントとしては構成できません。クラスタは静的なアドレス指定を使用する必要があります。クラスタ内では DHCP は、フェイルオーバー機能を提供する CAA (Cluster Application Availability) サブシステムと連携して、シングル・インスタンス・アプリケーションとして動作します。DHCP サーバとして動作できるのは常にクラスタの 1 つのメンバのみです。フェイルオーバーが行われた場合、新規 DHCP サーバは、前のサーバが使用していたのと同じ共通データベースを使用します。

DHCP サーバは、自身のホスト名および IP アドレスと DHCP データベース内のエントリとを一致させようとしています。クラスタ・メンバのホスト名と IP アドレスを使ってデータベースを構成している場合は、問題が発生する可能性があります。メンバがダウンした場合、DHCP サーバは別のメンバに自動的にフェイルオーバーされます。しかし、この新規 DHCP サーバのホスト名と IP アドレスは、DHCP データベース内のエントリと一致しません。この問題とその他の関連する問題を回避するには、次の手順を実行します。

1. Tru64 UNIX 『ネットワーク管理ガイド：接続編』の DHCP に関する章を参照し、この章で説明されている DHCP サーバの構成手順について把握します。
2. 初期 DHCP サーバとして実行するクラスタ・メンバ上で、`/usr/bin/X11/xjoin` を実行し、DHCP を構成します。
3. `[Server/Security]` を選択します。
4. 現在表示されている `[Server/Security Parameters]` のプルダウン・メニューから、`[IP Ranges]` を選択します。
5. `[DHCP Server]` エントリに、省略時のクラスタ別名の IP アドレスを設定します。

DHCP データベースには、DHCP サーバの IP アドレスが複数存在する場合があります。作業を簡単にするために、`jdbdump` コマンドを使って、DHCP データベースの内容をテキスト・ファイルに出力します。次に、テキスト・エディタを使って、このファイル内の元の DHCP サーバの IP アドレスをクラスタ別名の IP アドレスに変更します。最後に、`jdbmod` を使って、編集したファイルを DHCP データベースに読み込みます。たとえば、テキスト・ファイルへの出力とこのファイルの編集を行うには、次のコマンドを入力します。

```
# jdbdump > dhcp_db.txt
# vi dhcp_db.txt
```

`dhcp_db.txt` ファイルを編集して、元の DHCP サーバの IP アドレスを省略時のクラスタ別名の IP アドレスに変更します。

データベースを更新し、変更を反映させるには、次のコマンドを入力します。

```
# jdbmod -e dhcp_db.txt
```



6. xjoin を使って DHCP を終了したら、DHCP を高可用性アプリケーションにします。DHCP には処理スクリプトとリソース・プロファイルが既に用意されており、DHCP は CAA デーモンに既に登録されています。CAA サブシステムを使って DHCP を起動するには、次のコマンドを入力します。

```
# caa_start dhcp
```

7. /etc/join/server.pcy を編集して、次の行を追加します。

```
canonical_name cluster_alias
```

*cluster\_alias* は省略時のクラスタ別名です。

8. DHCP を停止して再起動します。

```
# caa_stop dhcp  
# caa_start dhcp
```

高可用性アプリケーションと CAA サブシステムについては、TruCluster Server 『クラスタ高可用性アプリケーション・ガイド』を参照してください。

## 7.2 NIS の構成

NIS (ネットワーク情報サービス) デーモン *ypxfrd* および *rpc.yppasswdd* は、クラスタのあらゆるメンバ上で動作し、メンバの可用性を高めます。

3.1 節で説明されているように、クラスタ別名を通してアクセスするサービスのポートは、*in\_single* または *in\_multi* のいずれかで定義します (この定義は、サービスを複数のクラスタ・メンバで同時に実行できるかどうかとは無関係です)。

*ypxfrd* は *in\_multi* サービスで動作します。つまり、クラスタ別名サブシステムは、このサービスに宛てられた接続要求とパケットを、その別名に対して資格のあるメンバ全員に送ります。

*rpc.yppasswdd* は *in\_single* サービスで動作します。つまり、このサービスに宛てられた接続要求とパケットは、1 つの別名メンバだけが受信します。そのメンバが使用できなくなると、クラスタ別名サブシステムは同じ別名内の他のメンバを選択し、このサービスに宛てられたすべての要求とパケットをそのメンバに受信させます。

NIS パラメータは /etc/rc.config.common ファイルに保存されています。NIS データベース・ファイルは /var/yp/src ディレクトリにあります。これらのファイルはクラスタのすべてのメンバによって共用されます。クラス

タは、スレーブ、マスタ、またはクライアントです。クラスタ内で、スレーブ、マスタ、およびクライアントの機能を備えたメンバは混在できません。

クラスタの作成時に NIS を構成した場合、NIS の構成に限り、クラスタに対するメンバの追加または削除時に何も行う必要はありません。

クラスタの稼働後に NIS を構成するには、次の手順を実行します。

1. `nissetup` コマンドを実行し、Tru64 UNIX 『ネットワーク管理ガイド：サービス編』の NIS に関する章の指示に従って、NIS を構成します。

NIS がバインドするホスト名として、クラスタ別名をホスト名のリストに追加しなければなりません。

2. クラスタの各メンバ上で、次のコマンドを実行します。

```
# /sbin/init.d/nis stop  
# /sbin/init.d/nis start
```

### 7.2.1 クラスタ内の NIS マスタへのエンハンスト・セキュリティ機能の追加

拡張ユーザ・プロファイルとパスワードによって NIS データベースを保護するように、NIS マスタを構成できます。NIS とエンハンスト・セキュリティ機能については、Tru64 UNIX 『セキュリティ管理ガイド』を参照してください。NIS マスタへのエンハンスト・セキュリティ機能の追加についての詳細は、同じマニュアルのクラスタ用のエンハンスト・セキュリティ機能に関する付録を参照してください。

## 7.3 印刷の構成

多少の例外はありますが、クラスタ内でのプリンタの構成はスタンドアロン Tru64 UNIX システム上でのプリンタの構成と同じです。プリンタ・システムの管理についての概説は、Tru64 UNIX 『システム管理ガイド』を参照してください。

クラスタでは、メンバはクラスタ内のどの場所にあるプリンタにもプリント・ジョブを送信できます。プリンタ・デーモン (`lpd`) はクラスタの各メンバ上で動作します。親デーモンである `lpd` は、ローカルの `lpr` の要求にも、リモートからのプリント・ジョブの要求にも応答します。

各ノードで動作する親プリンタ・デーモンは `/var/spool/lpd` を使用します。このパスは、`/cluster/members/{memb}/spool/lpd` へコンテキ

スト依存シンボリック・リンク (CDSL) でつながっています。他の目的では `/var/spool/lpd` を使用しないでください。

クラスタの各ローカル・プリンタにはそれぞれ専用のスプーリング・ディレクトリがあります。そのディレクトリは通常、`/usr/spool` の下にあります。スプーリング・ディレクトリは CDSL であってはなりません。

プリンタ特性 (`:on`) がクラスタ内での印刷をサポートするために新規に追加されました。プリンタを構成するには、クラスタの任意のメンバ上で `printconfig` または `lprsetup` を実行します。

プリンタが COM ポート (`/dev/tty01`) またはパラレル・ポート (`/dev/lp0`) 経由でメンバに接続されているローカル・プリンタである場合、`:on` にプリンタ接続先のメンバの名前を設定します。たとえば、`:on=memberA` と設定します。

この例では、プリンタはメンバ `memberA` に接続されています。

プリンタが TCP/IP 経由でメンバに接続されるネットワーク・プリンタである場合、`:on` 特性の値として次の 2 つの選択肢があります。

- `:on=localhost`

クラスタのあらゆるメンバが印刷を担当するようにするには、`localhost` を指定します。プリント・ジョブが送信されると、応答した最初のメンバが、キューが空になるまで、すべてのプリント・ジョブを処理します。ローカル・ジョブの場合、最初にジョブを投入したメンバが最初に応答するメンバとなります。外から入ってくるリモート・ジョブの場合、そのジョブのサービスはクラスタ別名に基づいて行われます。

- `:on=member1,member2,...,memberN`

クラスタのいずれか 1 つのメンバにすべての印刷を担当させるには、特定のメンバを列挙指定します。`:on` に列挙された最初のメンバがすべてのプリント・ジョブを処理します。そのメンバが使用できなくなった場合は、列挙された次のメンバが処理を引き継ぐという具合に続きます。

### Advanced Printing Software の使用

クラスタでの Advanced Printing Software のインストールおよび使用については、*Tru64 UNIX Advanced Printing Software* 『システム管理/操作ガイド』を参照してください。

## 7.4 DNS/BIND の構成

クラスタを BIND (Berkeley Internet Name Domain) サーバとして構成することは、スタンドアロン Tru64 UNIX システムを BIND サーバとして構成することに似ています。クラスタの場合は、クラスタ内の 1 つのメンバ上で `named` デーモンが動作します。Tru64 UNIX システムの場合は、そのシステム自体が実際の BIND サーバになります。クラスタでは、クラスタ別名サブシステムが名前の照会に応答するので、クラスタ全体が BIND サーバのように見えます。フェイルオーバー機能は CAA サブシステムによって提供されます。`named` デーモンが動作しているメンバが使用できなくなった場合、CAA サブシステムは別のメンバ上でこのデーモンを起動します。

`bindconfig` コマンドでは、クラスタをクライアントとするかサーバとするかのどちらか一方を指定します。クラスタはクライアントとサーバの両方として動作できるので、この選択はいくぶん誤解を招く恐れがあります。特に、クラスタを BIND サーバとして構成する場合、`/etc/resolv.conf` にネームサーバを指定していないときは、自動的にそのサーバの BIND クライアントとして構成されます。

- `/etc/resolv.conf` にネームサーバのエントリがある場合、クラスタは BIND クライアントです。各クラスタ・メンバは、`/etc/resolv.conf` に指定されたネームサーバの BIND クライアントです。
- 省略時のクラスタ別名の名前が `/etc/resolv.conf` のネームサーバとして指定されていて、クラスタも `named` デーモンを実行している場合、クラスタは BIND クライアントと BIND サーバの両方になります。

クラスタが BIND サーバとして構成され、`/etc/resolv.conf` にネームサーバの指定が無い場合、`bindconfig` がクラスタ別名を最初のネームサーバとして `/etc/resolv.conf` に自動的に追加します。これは、初期インストール中にシステムを BIND サーバとして構成する場合に行われます。ただし、クラスタを最初に BIND クライアントになるようにセットアップした後、BIND サーバにするように `bindconfig` を実行すると、`/etc/resolv.conf` にはすでに少なくとも 1 つネームサーバが指定されているものと想定します。この場合には、`bindconfig` は自動的にクラスタ別名を最初のネームサーバとして追加しません。これを変更するには `bindconfig` を使用します。

BIND 環境変数はクラスタ単位の `/etc/rc.config.common` に格納されているので、すべてのクラスタ・メンバはブート時に同一に構成されます。同

様に、`/etc/resolv.conf` はクラスタ単位ファイルなので、すべてのクラスタ・メンバは同じネームサーバを使用します。

BIND の構成手順は、クラスタの作成時でもクラスタの稼働後でも同じです。

クラスタを BIND クライアントとして、または BIND サーバとして構成するには、コマンド `bindconfig` または `sysman dns` を使用します。

どのメンバでコマンドを実行しても問題ありません。BIND サーバを構成する場合、CAA はネームサーバ `named` が実行するメンバを決めます。クライアントかサーバかに特定のメンバを構成しているのではなく、むしろクラスタ全体をクライアントかサーバかに構成しているので、`sysman -focus` オプションは、BIND を構成するには適切ではありません。つまり、BIND サーバ構成を行うメンバ上では、`named` ネームサーバは実行する必要はありません。CAA は `named` を任意のメンバ上で起動します。

`/etc/resolv.conf` および `/etc/svc.conf` ファイルはクラスタ単位ファイルです。

BIND の構成についての詳細は、Tru64 UNIX 『ネットワーク管理ガイド：サービス編』の DNS (ドメイン・ネーム・サービス) に関する章を参照してください。

## 7.5 時刻同期の管理

クラスタのすべてのメンバは時刻の同期を必要とし、そのためには NTP (Network Time Protocol) が必要です。この理由から、クラスタの作成時に `clu_create` コマンドを実行すると、初期メンバ上で NTP が構成されます。その後、クラスタに新規メンバが追加されるたびに、新規メンバ上で NTP が自動的に構成されます。つまり、クラスタのメンバはすべて NTP ピアとして構成されます。

自サイトで NTP を使用しない場合は、使用するタイム・サービスが『RFC 1035 Network Time Protocol (Version 3) Specification, Implementation and Analysis』で定義された細分性の要件を満たしているかどうかを確認してください。

クラスタのメンバ間でわずか数秒でもシステム時間の誤差があってはならないので、`timed` デモンによる時刻の同期化はお勧めしません。

## 7.5.1 NTP 構成

『クラスタ・インストール・ガイド』では、クラスタ・ソフトウェアをインストールして初期クラスタ・メンバのシステムを生成する前に、Tru64 UNIX システム上で NTP を構成することをお勧めしています。このようにしなければ、`clu_create` と `clu_add_member` が各クラスタ・メンバごとに自動的に NTP を構成してしまいます。この構成では、各メンバの NTP サーバが `localhost` になります。メンバはそれぞれが NTP ピアとして設定され、クラスタのインターコネクト・インタフェースの IP アドレスが使用されます。

`localhost` エントリは、そのメンバが稼働中のノードだけで使用されます。ピア・エントリは全クラスタ・メンバの同期をとるために動作するので、クラスタ内の時間誤差はマイクロ秒です。NTP 構成を後で変更して外部サーバを追加したとしても、初期サーバとピア・エントリは変更しないでください。

クラスタの稼働後に NTP 構成を変更するには、クラスタの各メンバ上で `ntpconfig` または `sysman ntp` を実行しなければなりません。これらのコマンドは常にクラスタの 1 つのメンバ上で動作します。NTP の構成対象のメンバを指定するには次の 2 とおりの方法があります。つまり、メンバにログインすれば、そのメンバが構成対象になります。あるいは、`sysman` の `-focus` オプションを使って、構成対象のメンバを指定します。NTP デーモン (`xntpd`) の起動と停止は、クラスタ全体に対して行くと、クラスタ全体が稼働不能になる可能性があるので、一度に 1 つのメンバに対して行う必要があります。

`sysman` を使用すれば、クラスタ全体または 1 つのメンバを対象に NTP デーモンの状態がわかります。

## 7.5.2 すべてのメンバでの同一外部 NTP サーバの使用

外部 NTP サーバをクラスタの 1 メンバだけに追加できます。ただし、単一点障害となりえます。これを回避するには、外部サーバの同じセットを全クラスタ・メンバに追加する必要があります。

外部 NTP サーバのリストを全メンバで同じにすることを強くお勧めします。メンバごとに外部サーバ・リストを異なるもので構成する場合は、サーバがすべて同じ階層レベルであり、これらの時間差を小さくする必要があります。

### 7.5.2.1 時間のずれ

クラスタ・メンバ間の時間のずれに気が付いた場合は、メンバ相互に同期をとる必要があります。この作業を行うには、クラスタの各メンバにログインして、`ntp -s -f` コマンドを入力し、ログインしたメンバ以外のメンバのクラスタ・インターコネクト名を指定する必要があります。省略時には、クラスタ・インターコネクト名は、`-ics0` が付加された簡略ホスト名です。たとえば、`provolone` がクラスタ・メンバで、`provolone` 以外のメンバにログインしている場合は、次のコマンドを入力します。

```
# ntp -s -f provolone-ics0
```

次に、他のクラスタ・メンバにログインして、このコマンドを繰り返しますが、どのメンバにおいても、ログインしているシステム以外のクラスタ・インターコネクト名を使用します。

## 7.6 NFS の管理

クラスタは高可用性のネットワーク・ファイル・システム (NFS) サービスを提供できます。クラスタが NFS サーバとして動作するとき、クラスタの外部クライアント・システムからは、クラスタはクラスタ別名で参照される単一システムとして見えます。クラスタが NFS クライアントとして動作している場合に、クラスタの外部 NFS ファイル・システムがクラスタの 1 つのメンバによってマウントされると、そのファイル・システムにはクラスタのすべてのメンバがアクセスできます。クラスタの外部 NFS サーバへのファイルのアクセスは、マウント元のこのメンバ経由で行われます。外部 NFS サーバからは、クラスタは独立したノードの集合として見え、クラスタのメンバ間でファイル・システムが共有されていることは意識されません。

### 7.6.1 NFS の構成

NFS を構成するには、`nfsconfig` または `sysman nfs` コマンドを使用します。

---

#### 注意

---

クラスタ内で `nfssetup` コマンドを使用しないでください。このコマンドはクラスタ対応ではないので、NFS が正しく構成されません。

---

クラスタの 1 つ以上のメンバで、NFS デーモンと mount デーモンの実行が可能であり、同時にクライアント・デーモン lockd および statd の実行も可能です。

`nfsconfig` または `sysman nfs` によって、クラスタ全体または特定のメンバに対して、次のことを行うことができます。

- NFS デーモンの起動，再起動，または停止
- サーバ・デーモンの構成または構成除外
- クライアント・デーモンの構成または構成除外
- NFS の構成状態の表示
- NFS デーモンの状態の表示

特定のメンバの NFS を構成するには、`sysman` の `-focus` オプションを使用します。

フォーカスを指定しないで NFS を構成すると、その構成はクラスタ全体に適用され、`/etc/rc.config.common` に保存されます。フォーカスを指定して NFS を構成すると、その構成はフォーカスを指定されたクラスタ・メンバのみに適用され、そのメンバの CDSL ファイル `/etc/rc.config` に保存されます。

ローカルの NFS 構成はクラスタ全体の NFS 構成を上書きします。たとえば、メンバ `mutt` を NFS サーバにならないように構成した場合、クラスタ全体を NFS サーバとして構成しても、その構成は `mutt` に適用されません。つまり、`mutt` は引き続き NFS サーバにはなりません。

ここで、もっと興味深い例を挙げます。たとえば、`alpha`、`beta`、`gamma` という 3 つのメンバから構成されるクラスタがあるとします。まず、8 つの TCP サーバ・スレッドをクラスタ全体に対して構成します。次に、メンバ `alpha` にフォーカスを指定し、10 個の TCP サーバ・スレッドを構成します。この条件で、各メンバ上で `ps` コマンドを実行してみます。その結果、次のことがわかります。つまり、メンバ `alpha` 上には 10 個の TCP サーバ・スレッドがあります。しかし、メンバ `beta` および `gamma` 上には 8 つの TCP サーバ・スレッドしかありません。今度は、クラスタ全体にフォーカスを指定し、TCP サーバ・スレッドの構成を 8 つから 12 個に変更します。この条件で、各メンバ上で `ps` コマンドを実行してみます。この場合、メンバ `alpha` 上には前と



同じ 10 個の TCP サーバ・スレッドがあります。しかし、メンバ beta および gamma には現在、それぞれ 12 個の TCP サーバ・スレッドがあります。

クラスタのメンバで `nfsd` を実行する場合は `mountd` も実行しなければなりません。逆に、`mountd` を実行する場合は `nfsd` を実行しなければなりません。`nfsconfig` または `sysman nfs` を使って NFS を構成するときは、この操作は自動的に行われます。

クラスタのメンバ上でロックが有効な場合は、デーモン `rpc.lockd` および `rpc.statd` がそのメンバ上で起動されます。ロックがクラスタ全体に対して構成された場合 (`rpc.lockd -c` と `rpc.statd -c`)、デーモン `lockd` および `statd` はクラスタ単位で起動されます。この場合、これらのデーモンは高可用性で、CAA サブシステムによって管理されます。NFS サーバは自身のアドレスとして、省略時のクラスタ別名または `/etc/exports.aliaes` で指定されている別名を使用します。

クラスタが NFS サーバとして動作するとき、クラスタの外部クライアント・システムからは、クラスタはクラスタ別名で参照される単一システムとして見えます。外部クライアント・システムがクラスタ内のディレクトリの CDSL をマウントしても、このシステムが見えるのはクラスタ・メンバ上のパスだけです。ただし、このメンバでは `statd` と `lockd` のペアがクラスタ単位で動作している必要があります。

NFS サービスの起動と停止は、特定のメンバに対してもクラスタ全体に対しても行うことができます。通常、クラスタ単位で動作する `lockd` と `statd` のペアの起動と停止は自動的に行われます。ただし、これらのデーモンを手動で停止する場合は、次のコマンドを入力します。

```
# caa_stop cluster_lockd
```

これらのデーモンを手動で起動する場合は、次のコマンドを入力します。

```
# caa_start cluster_lockd
```

サーバ・デーモン `lockd` および `statd` のペアを別のメンバに再配置するには、次のように `caa_relocate` コマンドを入力します。

```
# caa_relocate cluster_lockd
```

高可用性アプリケーションの起動および停止方法についての詳細は、第 8 章を参照してください。

## 7.6.2 クラスタ内での NFS の使用上の考慮事項

ここでは、クラスタ内とスタンドアロン・システム内で NFS を使用する際の違いについて説明します。

### 7.6.2.1 CFS での NFS ファイル・システムのサポート

CFS は NFS (ネットワーク・ファイル・システム) クライアントに読み取り/書き込みアクセスをサポートします。クラスタでファイル・システムを NFS マウントすると、CFS はすべてのクラスタ・メンバからの読み取り/書き込みアクセスを可能にします。実際にマウントしたメンバが、他のクラスタ・メンバに対するファイル・システムのサーバとなります。

NFS ファイル・システムをマウントしているメンバがシャットダウンされるか、障害が発生すると、ファイル・システムは自動的にアンマウントされ、CFS はマウント・ポイントをクリーンアップします。クリーンアップ処理の間、マウント・ポイントにアクセスするメンバはクリーンアップの進展状況により、次のような状態になります。

- ファイル・システム上でメンバがファイルをオープンしている場合、書き込みデータは NFS マウントされた実際のファイル・システムでなく、ローカル・キャッシュに送られる。
- ファイル・システム上ですべてのファイルをクローズした後では、ファイル・システムを再マウントしない限り、ファイル・システム上のファイルをオープンしようとしてもオープンできず、EIO エラーが発生する。アプリケーションは「Stale NFS handle」メッセージを通知される場合がある。これは、クラスタと同様にスタンドアロン・システムでも一般的に起こりえる。

CFS のクリーンアップが完了するまでは、メンバは NFS ファイル・システムのローカル・マウント・ポイントで(または、マウント・ポイントの下にローカルに作られたディレクトリ内で) 新しいファイルを作成することができます。

AutoFS か Automount を使用していなければ、NFS ファイル・システムは自動的に別のクラスタ・メンバにフェイルオーバーしません。別のクラスタ・メンバから、手動で同じマウント・ポイントか別のマウント・ポイントに再マウントして、再度使用可能にする必要があります。もう 1 つの方法としては、クラスタ・メンバをブートして、`/etc/fstab` ファイルに登録されて

いて、クラスタで現在マウントされておらず、サービスされていないファイル・システムを再マウントします。

#### 7.6.2.2 クライアントはクラスタ別名を使用しなければならない

クラスタが NFS サーバとして動作している場合、外部クライアントは、クラスタからファイル・システムをマウントするとき、省略時のクラスタ別名か、または `/etc/exports.aliases` で指定されている別名を使ってホストを指定しなければなりません。外部クライアントがクラスタからファイル・システムをマウントしようとしたが、省略時のクラスタ別名または `/etc/exports.aliases` で指定されている別名を使用しなかった場合は、「connection refused」というエラー・メッセージがそのクライアントに返されます。

`mountd` 経由で動作するその他のコマンド (たとえば `umount` と `export`) が外部クライアントから発行されたが、省略時のクラスタ別名も `/etc/exports.aliases` で指定されている別名も使用されなかった場合、「Program unavailable」というエラー・メッセージがそのクライアントに返されます。

NFS サーバとして使用する別名を追加する場合は、その前に、『クラスタ概要』にある、NFS およびクラスタ別名と NFS, TCP, および UDP (User Datagram Protocol) トラフィックがどのように相互作用するか説明した節を参照してください。また、`exports.aliases(4)` と `/etc/exports.aliases` ファイルの冒頭にあるコメントも参照してください。

#### 7.6.2.3 CDSL を使って NFS ファイル・システムをマウントする

クラスタが NFS クライアントとして動作している場合、1 つのクラスタ・メンバがマウントした NFS ファイル・システムは、すべてのクラスタ・メンバからアクセスできます。クラスタ・ファイル・システム (CFS) では、外部の NFS サーバにあるファイルへアクセスする場合、そのファイル・システムをマウントしたメンバを経由します。つまり、マウントを実行したクラスタ・メンバが、NFS ファイル・システムの CFS サーバになり、外部 NFS と通信するノードになります。その結果、CFS では、クラスタ・メンバ全体でキャッシュ内容が同一に保たれ、全メンバから NFS ファイル・システムが常に同じように見えます。

ただし、マウント・メンバが使用できなくなると、フェイルオーバーは起こりません。他のクラスタ・メンバが NFS ファイル・システムをマウントするまで、NFS ファイル・システムへアクセスできません。

ファイル・システムが使用できなくなった場合の対応方法には何通りかあります。その中でも、AutoFS を使用して NFS ファイル・システムを自動的にフェイルオーバーする方法が最も確実です。この方法を使えば、可用性を損なうことなくクラスタ・メンバの間でキャッシュの整合性を保つことができます。AutoFS をクラスタ環境で使用方法は、7.6.3 項に記載されています。

AutoFS とは別の方法として、`mkcdsl -a` コマンドを用いてマウント・ポイントを CDSL に変換する方法があります。この変換する方法では、すべてのメンバのメンバ固有領域に、既存のディレクトリがコピーされます。そうしておいて、CDSL を NFS ファイル・システムのマウント・ポイントとして使用します。この方法では、ファイル・システムに対する NFS サーバが 1 つだけですが、各クラスタ・メンバがそれぞれ NFS クライアントになります。クライアント・メンバは、NFS ファイル・システムの CFS サーバとして機能する 1 つのクラスタ・メンバに依存しているわけではありません。1 つのクラスタ・メンバが使用できなくなっても、他のクラスタ・メンバが行う NFS ファイル・システムへのアクセスは影響を受けません。ただし、CFS はクラスタ・メンバ間でキャッシュの整合性を保証できません。クラスタ・メンバは NFS の通常の方法を使用したキャッシュの整合性に依存することになります。当然ですが、シングル・システムの場合とは異なります。

NFS のレベルで提供されるファイル・システムの整合性を受け容れられる環境では、次の手順を実行して、CDSL をマウント・ポイントとして使用します。

1. マウント・ポイントがない場合は作成します。

```
# mkdir /mountpoint
```

2. `mkcdsl -a` コマンドを使用して、ディレクトリを CDSL に変換します。これにより、全メンバのメンバ固有領域に、既存のディレクトリがコピーされます。

```
# mkcdsl -a /mountpoint
```

3. 同じ NFS サーバを使用して、各クラスタ・メンバに NFS ファイル・システムをマウントします。

```
# mount server:/filesystem /mountpoint
```

マウント情報を `/etc/fstab` ファイルに追加し、各クラスタ・メンバでマウントが自動的に実行されるようにすることをお勧めします。

#### 7.6.2.4 ループバック・マウントはサポートされていない

NFS ループバック・マウントはクラスタでは機能しません。クラスタからファイル・システムをクラスタ内のディレクトリに NFS マウントしようとする、「Operation not supported」というエラー・メッセージが返されます。

#### 7.6.2.5 NFS マウントしたパスに非 NFS ファイル・システムをマウントしてはならない

CFS では、NFS マウントしたパスに非 NFS ファイル・システムをマウントしてはなりません。この制限によって、NFS ファイル・システムをマウントしたクラスタ・メンバがダウンした場合でも、対応する物理ファイル・システムの可用性には問題は発生しません。

### 7.6.3 クラスタでの AutoFS の使用

NFS ファイル・システムを自動マウントにする場合には、AutoFS を使用します。AutoFS は、CAA の自動マウント・サービスを使用して自動フェイルオーバーを実現します。あるメンバが自動マウントされたファイル・システムの CFS サーバとして動作し、AutoFS デーモン `autofs` のアクティブ・コピーを起動します。このメンバに障害が発生すると、CAA は別のメンバ上で `autofs` を開始します。

AutoFS の構成方法については、Tru64 UNIX 『ネットワーク管理ガイド：サービス編』のリモート・ファイル・システムの自動マウントについての節を参照してください。AutoFS を構成した後は、次のようにしてデーモンを起動する必要があります。

```
# caa_start autofs
```

`autofs` リソースを自動的に起動させたい場合は、`/usr/sbin/caa_profile -update` コマンドを使用して、`auto_start` のプロファイル・オプションを 1 に設定します。

AutoFS を使用する場合は、次の事項を考慮してください。

- 1 つの単一 NFS サーバから多数のファイル・システムをインポートしたり、特に遅いデータリンクを使うサーバからインポートしたりするクラ

スタでは、autofs サブシステム内の mount\_timeout カーネル属性の値を増やす必要があるかもしれません。mount\_timeout の省略時の値は 30 秒です。メンバが稼働中にこの属性の値を変更するには、sysconfig コマンドを使用します。たとえば、タイムアウト値を 50 秒に変更するには、次のコマンドを使用します。

```
# sysconfig -r autofs mount_timeout=50
```

- 自動マウントしたファイル・システムを表示すると、ファイル・システムは何度かマウントされたかのように見えます。カーネルが自動マウント要求のタイムアウトを検出するが、NFS マウントが継続されたときにこの現象が発生します。つまり、こタイムアウトにより、次の自動マウント要求が処理された結果、重複したマウントが発生します。この場合には、操作上の影響はありません。
- autofs daemon を起動するか、autofsmount を実行して自動ファイル・システムのマッピングを処理するとき、AutoFS はすべてのクラスター・メンバが同じバージョンの TruCluster Server ソフトウェアを実行していることを確認します。

TruCluster Server バージョン 5.1A では、SCRIPT\_TIMEOUT 属性の値が 3600 に増やされ、autofs がタイムアウトしにくくなりました。この値を増やすことはできますが、減らすことはお勧めできません。

TruCluster Server の以前のバージョンでは、インポートしているファイル・システムの数、データ・リンクの速度、インポートされたファイル・システムのサーバ間での配布方法に応じて、次のような CAA メッセージが表示される可能性があります。

```
# CAAD[564686]: RTD #0: Action Script \
/var/cluster/caa/script/autofs.scr(start) timed out! (timeout=180)
```

この場合、autofs の CAA プロファイル内の属性 SCRIPT\_TIMEOUT の値を 180 より大きい値に増やす必要があります。値を増やすには、/var/cluster/caa/profile/autofs.cap を編集するか、コマンド caa\_profile -update autofs を使ってプロファイルを更新します。

たとえば、SCRIPT\_TIMEOUT を 3600 秒に増やすには、次のようなコマンドを入力します。

```
# caa_profile -update autofs -o st=3600
```

CAA プロファイルとコマンド caa\_profile の使用方法については、caa\_profile(8) を参照してください。

### 7.6.3.1 ファイル・システムの強制アンマウント

クラスタ・メンバの AutoFS が停止したか使用できなくなった場合 (CAA `autofs` リソースが停止した場合など) でも、インターセプト・ポイントと AutoFS が自動マウントしたファイル・システムは使用できます。ただし、AutoFS が、使用中のファイル・システムのあるクラスタ・メンバ上で停止した後、別のメンバで開始されると、AutoFS インターセプト・ポイントが前のクラスタ・メンバをサーバとして認識したままになります。これは、AutoFS インターセプト・ポイントが、自分のところにマウントされているファイル・システムが使用中であると自分自身も使用中になり、前のクラスタ・メンバをサーバとして要求するからです。これらのインターセプト・ポイントでは新しい自動マウントを行うことはできません。

#### 7.6.3.1.1 強制アンマウントが必要かどうかの判断

この問題は次の状況で発生します。

- 自動マウントしたファイル・システムへアクセスしたときに問題の発生することが明らかな場合
- CAA `autofs` リソースを移動した場合

自動マウントしたファイル・システムへアクセスしたときに問題の発生することが明らかな場合は、自動マウントしたファイル・システムが期待したとおりに動作していることを確認します。手順は、次のとおりです。

1. `caa_stat autofs` コマンドを使用して、`autofs` リソースの動作している場所を CAA がどのように把握しているかを確認します。
2. 次の `ps` コマンドを使用して、CAA の期待しているメンバで確かに `autofs` デーモンが動作していることを確認します。  

```
# ps agx | grep autofs
```

動作してない場合は、動作させて問題が解決されるかどうかを確認します。
3. アクセスできないファイル・システムに対応する、自動マウントのマップ・エントリを調べます。たとえば、エントリを対象に `/etc/auto.x` を検索して調べます。
4. `cfsmgr -e` コマンドを使用して、マウント・ポイントが存在しかつ期待するメンバがサービスを行っているかどうかを調べます。  
サーバが CAA の期待どおりでなければ、問題があります。

CAA リソースを他のメンバに移動する場合は、`mount -e` コマンドを使用して AutoFS インターセプト・ポイントを識別するとともに、`cfsmgr -e` コマンドを使用して、すべてのマウント・ポイントのサーバを表示します。AutoFS を停止したメンバ上で、すべての AutoFS インターセプト・ポイントと自動マウントしたファイル・システムがアンマウントされていることを確認してください。

`mount -e` コマンドを使用する場合は、その出力を次のように検索して `autofs` の出現する行を探します。

```
# mount -e | grep autofs
/etc/auto.direct on /mnt/mytmp type autofs (rw, noatime, direct)
```

`cfsmgr -e` コマンドを使用する場合は、出力を検索して次のようなマップ・ファイル・エントリを探します。「Server Status」フィールドには、ファイル・システムが実際にサービスされているかどうかは表示されません。「Server Name」フィールドで、AutoFS が停止されたメンバの名前を検索してください。

```
# cfsmgr -e
Domain or filesystem name = /etc/auto.direct
Mounted On = /mnt/mytmp
Server Name = provolone
Server Status : OK
```

#### 7.6.3.1.2 問題の解決

問題になっている使用中のファイルがアクティブでなくなるまで待てる場合は、待ってください。そして、アクティブでなくなってから以前の AutoFS サーバ・ノードで `autofs mount -U` コマンドを実行して、そのファイル・システムをアンマウントします。このアプローチは時間がかかりますが、比較的安全な方法です。

問題になっている使用中のファイル・システムがアクティブでなくなるまで待てない場合は、以前の AutoFS サーバ・ノードで `cfsmgr -K directory` コマンドを使用して、AutoFS インターセプト・ポイントと、そのノードがサービスしている自動マウントされたファイルをすべて、使用中かどうかに関係なく強制的にアンマウントします。

#### 注意

`cfsmgr -K` コマンドは AutoFS インターセプト・ポイントと、そのノードがサービスしている自動マウントされたファイル・システムをすべてアンマウントする最も強力な方法です。しかし、



`cfsmgr -K` コマンドが常に成功するとは限りません。たとえば、NFS サーバがダウンするか NFS サーバと通信できなくなったために NFS が止まっていると、`cfsmgr -K` コマンドは動作しません。

`cfsmgr -K` コマンドを実行すると、アプリケーションは、自分がオープンしているファイルの中に影響を受けるファイル・システムにあるものがあると、そのファイルに対して I/O エラーを受信します。影響を受けるファイル・システムに現在の作業ディレクトリがあるアプリケーションは、そのファイル・システムのネームスペースで相対パス名を用いて位置を指定することができなくなります。

---

次の手順を実行して、`autofs` CAA リソースを再配置し、AutoFS インターセプト・ポイントと自動マウントしたファイル・システムを強制的にアンマウントします。

1. 可能であればシステムを静止させて、ユーザとアプリケーションの混乱を最小限に抑えます。
2. 次のコマンドを入力して、`autofs` CAA リソースを停止します。  

```
# caa_stop autofs
```

CAA は、自動マウントしたファイル・システムにまだ使用中のものがあっても、`autofs` リソースを停止すべきであると解釈します。
3. 次のコマンドを入力して、AutoFS インターセプト・ポイントと自動マウントしたファイル・システムがすべてアンマウントされたことを確認します。出力で `autofs` の参照を検索します。  

```
# mount -e
```
4. アンマウントされていないものがまだある場合は、次のコマンドを入力して AutoFS インターセプトと自動マウントされたファイル・システムを強制的にアンマウントします。  

```
#cfsmgr -K directory
```
5. AutoFS インターセプト・ポイントまたは自動マウントされたファイル・システムがマウントされているディレクトリを指定します。インターセプトと自動マウントされたファイル・システム (同じノードでサービスされているもの) をすべて削除するために入力する必要があるマウント・ディレクトリは 1 つだけです。

6. 次のコマンドを入力して `autofs` リソースを起動します。

```
# caa_start autofs -c cluster_member_to_be_server
```

## 7.6.4 Automount から AutoFS への移行

この項では、Automount から AutoFS への移行にあたって、考えられる3つの場面について説明します。

- クラスタ・メンバのリブートなしの移行(7.6.4.1 項)
- アクティブな自動マウント・サーバ・ノードをリブートする移行(7.6.4.2 項)
- クラスタ全体をリブートする移行(7.6.4.3 項)

### 7.6.4.1 クラスタ・メンバのリブートなしの移行

クラスタ・メンバのリブートなしの移行は、多くの手順が必要ですが、最高の可用性をもたらします。リブートによって、Automount のインターセプト・ポイントをクリーンアップし、AutoFS を自動的に起動することができないため、最も多くの手順が必要になります。

---

#### 注意

---

ほとんどの Automount の環境では、1 つの automount インスタンスで、すべてのマップ・ファイル进行处理します。ここで説明する手順では、この一般的なケースについて説明します。

各マップ・ファイルを複数の automount インスタンスで処理する複雑な Automount 環境の場合、`/etc/rc.config.common` ファイルをカスタマイズしたり、`ps` コマンドで戻された複数のプロセス識別子を強制終了する必要があったり、すべての NFS ファイル・システムに対して1つのメンバだけが自動マウント・サーバ・ノードでないなどの状況が発生します。

Automount 環境に適合する手順を考慮する場合、アクティブな Automount サーバ・プロセスを強制終了させるときに、Automount サービスのフェイルオーバーが発生しないように、Automount の“standby”プロセスを強制終了してください。

---

クラスタ・メンバのリブートなしに Automount から AutoFS へ移行する手順は、次のとおりです。

1. `rc.config.common` ファイルを変更します。

- a. `autofsmount` の引数を調べます。これらの引数は、通常、すでに `AUTOMOUNT_ARGS` 環境変数で指定された引数のサブセットです。この変数の値を表示するには、次の例で示すように `rcmgr -get` コマンドを使用します。

```
# /usr/sbin/rcmgr -c get AUTOMOUNT_ARGS
-D MACH=alpha -D NET=f /- /etc/auto.direct
```

`-D` オプションを使用して設定された環境変数は、`automount` のマップ・ファイル・エントリ定義内のプレースホルダを解決します。たとえば、マップ・ファイルに、次のような関連する `NET` エントリがあるとします。

```
vsx ${NET}system:/share/hunch/usr/projects2/vsx
```

このプレースホルダを解決すると、次のようになります。

```
vsx filesystem:/share/hunch/usr/projects2/vsx
```

- b. 前の手順での決定に従って、`autofsmount` に渡す引数を設定します。これを行うには、次の例で示すように、`rcmgr -set` コマンドを使用します。

```
# /usr/sbin/rcmgr -c set AUTOFSMOUNT_ARGS -D MACH=alpha -D NET=f /- /etc/auto.direct
```

- c. `autofsd` デーモンに渡す引数を次の例で示すように設定します。

```
# /usr/sbin/rcmgr -c set AUTOFSD_ARGS -D MACH=alpha -D NET=f
```

これらの引数は、`AUTOMOUNT_ARGS` に設定されたのと同じように、`-D` オプションの環境変数と一致する必要があります。

- d. `mount -e` コマンドを使用して、`automount` によって使用されるファイル・システムを識別します。

```
# mount -e | grep "(pid"
deli.zk3.dec.com:(pid524825) on /net type nfs (v2, ro, nogrpid, udp, hard, intr, noac, timeo=350, retrans=5)
```

自動マウントされるファイル・システムは、`hostname:(pid)` によって示されます。

- e. 次の例で示すように、どのクラスタが前の手順で識別された NFS ファイル・システムの Automount サーバ・ノードかを調べます。

```
# cfsmgr -p /net
Domain or filesystem name = /net
Server Name = swiss
Server Status: OK
```

- f. 前の手順で識別した Automount サーバを除くすべてのクラスタ・メンバでの Automount サービスを停止します。これを行うには、`ps -ef` コマンドを使用し、プロセス識別子を表示して、その出力から `automount` のインスタンスを探した後、`kill -TERM` コマンドを使用して各プロセスを強制終了します。TERM は省略時の値です。

```
# ps -ef | grep automount
root 1049132 1048577 0.0 May 10 ?? 0:00.00 /usr/sbin/automount -D MACH=alpha -D NET=f /- /etc/auto.direct

# kill 1049132
```

Tru64 UNIX バージョン 5.1A からは、`kill` コマンドがクラスタ単位で動作し、どのクラスタ・メンバからでもプロセスを強制終了できます。

- g. 次のように、`rc.config.common` ファイルで Automount を無効にし、AutoFS を有効にします。

```
# /usr/sbin/rcmgr -c set AUTOMOUNT 0
# /usr/sbin/rcmgr -c set AUTOFS 1
```

2. すべての自動マウントされたファイル・システムが休止状態になるまで待ちます。
3. サーバとして動作しているクラスタ・メンバでの Automount サービスを停止します。これを行うには、`ps -ef` コマンドを使用し、プロセス識別子を表示して、その出力から `automount` のインスタンスを探した後、`kill -TERM` を使用して各プロセスを強制終了します。TERM は省略時の値です。`automount` デーモンに `SIGTERM` シグナル送ると、マウントされていたすべてのファイル・システムをアンマウントして終了します。

```
# ps -ef | grep automount
root 524825 524289 0.0 May 10 ?? 0:00.01 /usr/sbin/automount -D MACH=alpha -D NET=f /- /etc/auto.direct

# kill 524825
```

4. `mount -e` コマンドを使用し、`tmp_mnt` の出力、または `automount -M` コマンドで指定されたディレクトリを検索し、自動マウントされたファイル・システムがマウントされていないことを確認します。

```
# mount -e | grep tmp_mnt
```

マウント・ポイントがまだ存在していても、もはや期待されるパス名で使用することはできません。ただし、`/tmp_mnt/...` というフル・パス名では使用することができます。AutoFS は `/tmp_mnt` マウント・ポイントを使用しないので、競合がなく、完全な自動マウントのネームスペースが AutoFS で使用可能になります。これらの `tmp_mnt` マウント・ポイントが後でアイドル状態になった場合は、`umount` コマン

ドの `-f` オプションを使用してアンマウントできます。このオプションは、サーバに通知することなしにリモート・ファイル・システムをアンマウントします。

5. AutoFS を起動します。AutoFS は CAA を使用して自動マウント・サービスの自動フェイルオーバー機能を提供します。つまり、1 つのクラスタ・メンバが自動マウントされたファイル・システムの CFS サーバとして動作し、AutoFS デーモンの 1 つのアクティブ・コピーを実行します。このクラスタ・メンバに障害が発生すると、CAA は別のメンバ上で `autofs` リソースを起動します。

AutoFS をどのノードで実行してもかわない場合は、クラスタ・メンバを指定しないで `/usr/sbin/caa_start autofs` コマンドを使用します。AutoFS を実行するクラスタ・メンバが必要であれば、`/usr/sbin/caa_start autofs -cmember-name` コマンドでそのメンバを指定します。

```
# /usr/sbin/caa_start autofs
```

`-c` オプションで指定したクラスタ・メンバが配置ポリシとリソースの依存性を満足している場合は、そのメンバ上で `autofs` リソースを起動します。クラスタ・メンバが配置ポリシとリソースの依存性を満足していない場合、`caa_start` コマンドは失敗します。指定されたメンバが使用可能でない場合には、そのコマンドは失敗します。

リソース・ファイル・オプションについては、`caa_profile(8)` を参照してください。

6. `caa_stat autofs` コマンドを使用して、`autofs` リソースを期待通りに起動したことを確認します。

```
# /usr/bin/caa_stat autofs
NAME=autofs
TYPE=application
TARGET=ONLINE
STATE=ONLINE on swiss
```

#### 7.6.4.2 クラスタ・メンバをリブートする移行

クラスタ・メンバをリブートする移行は、リブートなしの移行よりも、可用性は落ちますが実行手順が少なくなります。

---

## 注意

---

クラスタ・メンバをシャットダウンする前に、そのクラスタ・メンバが不可欠な投票メンバであるかどうかと、restricted ポリシで指定された 1 つ以上のアプリケーションの唯一のホスト・メンバであるかどうかを調べる必要があります。これらについては、5.5 節で説明しています。

ほとんどの Automount の環境では、1 つの automount インスタンスで、すべてのマップ・ファイルを処理します。ここで説明する手順では、この一般的なケースについて説明します。

各マップ・ファイルを複数の automount インスタンスで処理する複雑な Automount 環境の場合、`/etc/rc.config.common` ファイルをカスタマイズしたり、`ps` コマンドで戻された複数のプロセス識別子を強制終了する必要があるったり、すべての NFS ファイル・システムに対して 1 つのメンバだけが自動マウント・サーバ・ノードでないなどの状況が発生します。

Automount 環境に適合する手順を考慮する場合、アクティブな Automount サーバ・プロセスを強制終了させるときに、Automount サービスのフェイルオーバーが発生しないように、Automount の “standby” プロセスを強制終了してください。

---

クラスタ・メンバをリブートする、Automount から AutoFS への移行手順は、次のとおりです。

1. `rc.config.common` ファイルを変更します。
  - a. `autofs` コマンドに渡す引数を調べます。これらの引数は、通常、すでに `AUTOMOUNT_ARGS` 環境変数で指定されている引数のサブセットです。この変数の値を表示するには、次の例で示すように `rcmgr -get` コマンドを使用します。

```
# /usr/sbin/rcmgr -c get AUTOMOUNT_ARGS
-m -D MACH=alpha -D NET=f /- /etc/auto.direct
```

`-D` オプションを使用して設定された環境変数は、`automount` のマップ・ファイル・エントリ定義内のプレースホルダを解決します。たとえば、マップ・ファイルに、次のような関連する `NET` エントリがあるとします。

```
vsx ${NET}system:/share/hunch/usr/projects2/vsx
```

このエントリは、次のように解決されます。

```
vsx fsystem:/share/hunch/usr/projects2/vsx
```

- b. 前の手順での決定に従って、autofs mount に渡す引数を設定します。これを行うには、次の例で示すように、rcmgr -set コマンドを使用します。

```
# /usr/sbin/rcmgr -c set AUTOFSMOUNT_ARGS -D MACH=alpha -D NET=f /- /etc/auto.direct
```

- c. 次の例で示すように、autofs dデーモンに渡す引数を設定します。

```
# /usr/sbin/rcmgr -c set AUTOFS_ARGS -D MACH=alpha -D NET=f
```

これらの引数は、AUTOMOUNT\_ARGS に設定されたのと同じように、-D オプションの環境変数と一致する必要があります。

- d. mount -e コマンドを使用して、automount で使用されるファイル・システムを識別します。

```
# mount -e | grep "(pid"
deli.zk3.dec.com:(pid524825) on /net type nfs (v2, ro, nogrpid, udp, hard, intr, noac, timeo=350, retrans=5)
```

自動マウントされたファイル・システムは hostname:(pid) によって示されます。

- e. 次の例で示すように、どのクラスタが前の手順で識別された NFS ファイル・システムの Automount サーバ・ノードかを調べます。

```
# cfsmgr -p /net
Domain or filesystem name = /net
Server Name = swiss
Server Status: OK
```

- f. 前の手順で識別した Automount サーバを除くすべてのクラスタ・メンバでの Automount サービスを停止します。これを行うには、ps -ef コマンドを使用し、プロセス識別子を表示して、その出力から automount のインスタンスを探した後、kill -TERM コマンドを使用して各プロセスを強制終了します。TERM は省略時の値です。

```
# ps -ef | grep automount
root 1049132 1048577 0.0 May 10 ?? 0:00.00 /usr/sbin/automount -D MACH=alpha -D NET=f /- /etc/auto.direct
```

```
# kill 1049132
```

Tru64 UNIX バージョン 5.1A からは、kill コマンドがクラスタ単位で動作し、どのクラスタ・メンバからでもプロセスを強制終了できます。

- g. 次のように、rc.config.common ファイルで Automount を無効にし、AutoFS を有効にします。

```
# /usr/sbin/rcmgr -c set AUTOMOUNT 0
# /usr/sbin/rcmgr -c set AUTOFS 1
```

2. (オプション) AutoFS サーバを指定します。AutoFS は、CAA を使用して自動マウント・サービスの自動フェイルオーバー機能を提供します。つまり、1 つのクラスタ・メンバが自動マウントされたファイル・システムの CFS サーバとして動作し、AutoFS デーモンの 1 つのアクティブ・コピーを実行します。このクラスタ・メンバに障害が発生すると、CAA は別のメンバ上で `autofs` リソースを起動します。

CAA ホスト・ポリシーおよび配置ポリシーがある場合は、`caa_profile autofs -print` コマンドを使用して表示できます。ホスト・ポリシーでは、アプリケーション・リソースのホスト・メンバを空白で区切った順序付きリストで指定します。配置ポリシーでは、CAA がアプリケーション・リソースの起動または再起動を行うメンバを選択する際に従うポリシーを指定します。自動起動ポリシーは、リブートの前に停止していたか動作していたかにかかわらず、自動的にリソースを起動するかどうかを決定します。リブートの前に動作していたかどうかにかかわらずリソースを起動させたい場合は、`auto_start=1` に設定します。

```
# /usr/sbin/caa_profile autofs -print
NAME=autofs
TYPE=application
ACTION_SCRIPT=autofs.scr
ACTIVE_PLACEMENT=0
AUTO_START=0
CHECK_INTERVAL=0
DESCRIPTION=Autofs Services
FAILOVER_DELAY=0
FAILURE_INTERVAL=0
FAILURE_THRESHOLD=0
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=
PLACEMENT=balanced
REQUIRED_RESOURCES=
RESTART_ATTEMPTS=3
SCRIPT_TIMEOUT=3600
```

省略時および推奨する設定は、`balanced` の配置ポリシーを使って、どのクラスタ・メンバ上でも実行できる動作です。このポリシーが適切でない環境では、`/usr/bin/caa_profile -update` コマンドを使用して、`autofs` リソース・プロファイルを変更します。

リソース・ファイル・オプションについては、`caa_profile(8)` を参照してください。



変更した場合、CAA の `/usr/sbin/caa_register -u autofs` コマンドを使用して、変更を有効にします。

3. クラスタ・メンバをリブートします。クラスタ・メンバをシャットダウンする前に、そのクラスタ・メンバが不可欠な投票メンバでないことと、restricted 配置ポリシーで指定された 1 つ以上のアプリケーションの唯一のホスト・メンバでないことを確認します。これらについては、5.5 節で説明しています。

リブートすると、AutoFS が起動し、Automount はクラスタで実行されません。

```
# /sbin/shutdown -r now
```

#### 7.6.4.3 クラスタをリブートする移行

クラスタ全体をリブートする移行は、リブートなしの移行や単一メンバをリブートする移行より実行手順が少なくなりますが、クラスタの可用性は失われます。

クラスタのリブートは、思い切った方法であり、推奨する移行方法ではありません。

クラスタのリブート時に次の手順に従って、Automount から AutoFS へ移行します。

1. `rc.config.common` ファイルを変更します。
  - a. `autofsmount` に渡す引数を調べます。これらの引数は、通常、すでに `AUTOMOUNT_ARGS` 環境変数で指定されている引数のサブセットです。この変数の値を表示するには、次の例で示すように `rcmgr -get` コマンドを使用します。

```
# /usr/sbin/rcmgr -c get AUTOMOUNT_ARGS
-D MACH=alpha -D NET=f /- /etc/auto.direct
```

`-D` オプションを使用して設定された環境変数は、`automount` のマップ・ファイル・エントリ定義内のプレースホルダを解決します。たとえば、マップ・ファイルに、次のような関連する `NET` エントリがあるとします。

```
vsx ${NET}system:/share/hunch/usr/projects2/vsx
```

このエントリは、次のように解決されます。

```
vsx fsystem:/share/hunch/usr/projects2/vsx
```

- b. 前の手順での決定に従って、autofsmount に渡す引数を設定します。これを行うには、次の例で示すように、rcmgr -set コマンドを使用します。

```
# /usr/sbin/rcmgr -c set AUTOFSMOUNT_ARGS -D MACH=alpha -D NET=f /- /etc/auto.direct
```

- c. 次の例で示すように、autofs daemon に渡す引数を設定します。

```
# /usr/sbin/rcmgr -c set AUTOFS_ARGS -D MACH=alpha -D NET=f
```

これらの引数は、AUTOMOUNT\_ARGS に設定されたのと同じように、-D オプションの環境変数と一致する必要があります。

- d. 次のように、rc.config.common ファイルで Automount を無効にし、AutoFS を有効にします。

```
# /usr/sbin/rcmgr -c set AUTOMOUNT 0
# /usr/sbin/rcmgr -c set AUTOFS 1
```

2. (オプション) AutoFS サーバを指定します。AutoFS は、CAA を使用して自動マウント・サービスの自動フェイルオーバー機能を提供します。つまり、1 つのクラスタ・メンバが自動マウントされたファイル・システムの CFS サーバとして動作し、AutoFS デーモンの 1 つのアクティブ・コピーを実行します。このクラスタ・メンバに障害が発生すると、CAA は別のメンバ上で autofs リソースを起動します。

CAA ホスト・ポリシおよび配置ポリシがある場合は、

/usr/bin/caa\_profile autofs -print コマンドを使用して表示できます。ホスト・ポリシでは、アプリケーション・リソースのホスト・メンバを空白で区切った順序付きリストで指定します。配置ポリシでは、CAA がアプリケーション・リソースの起動または再起動を行うメンバを選択する際に従うポリシを指定します。自動起動ポリシは、リブートの前に停止していたか動作していたかにかかわらず、自動的にリソースを起動するかどうかを決定します。リブートの前に動作していたかどうかにかかわらずリソースを起動させたい場合は、auto\_start=1 に設定します。

```
# /usr/bin/caa_profile autofs -print
NAME=autofs
TYPE=application
ACTION_SCRIPT=autofs.scr
ACTIVE_PLACEMENT=0
AUTO_START=0
CHECK_INTERVAL=0
DESCRIPTION=Autofs Services
FAILOVER_DELAY=0
FAILURE_INTERVAL=0
```

```
FAILURE_THRESHOLD=0
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=
PLACEMENT=balanced
REQUIRED_RESOURCES=
RESTART_ATTEMPTS=3
SCRIPT_TIMEOUT=3600
```

省略時および推奨する設定は、balanced の配置ポリシーを使って、どのクラスタ・メンバ上でも実行できる動作です。このポリシーが適切でない環境では、`/usr/sbin/caa_profile -update` コマンドを使用して、`autofs` リソース・プロファイルを変更します。

リソース・ファイル・オプションについては、`caa_profile(8)` を参照してください。

変更した場合、CAA の `/usr/sbin/caa_register -u autofs` コマンドを使用して、変更を有効にします。

3. クラスタをリブートします。リブートすると、Automount はクラスタ内で実行されておらず、AutoFS が起動しています。

```
# /sbin/shutdown -c now
```

## 7.7 inetd の構成の管理

インターネット・サーバ・デーモン (`inetd`) の構成データは、次の 2 つのファイルに保存されています。

- `/etc/inetd.conf`

クラスタのすべてのメンバによって共用されるクラスタ単位のファイル。`/etc/inetd.conf` を使って、あらゆるメンバ上で同じ動作をするネットワーク・サービスを構成します。

- `/etc/inetd.conf.local`

クラスタの各メンバに固有の構成データを保存するファイル。メンバごとに異なる動作をするネットワーク・サービスを構成するときに使用します。

ローカル・メンバ上のクラスタ単位のサービスを無効にするには、そのメンバの `/etc/inetd.conf.local` 内で、無効にするサービスの `ServerPath` フィールドに `disable` を入力します。たとえば、`inetd.conf` 内でクラスタ全体に対する `finger` が有効になっている場合、1 つのメンバに対して `finger` を無効にするには、そのメンバの `inetd.conf.local` ファイルに次の行を追加します。

```
finger stream tcp      nowait root    disable      fingerd
```

/etc/inetd.conf.local がメンバ上にないときは、/etc/inetd.conf 内のエントリが使用されます。inetd.conf.local がメンバ上にあるときは、そのファイル内のエントリが inetd.conf 内のエントリよりも優先されます。

## 7.8 メール・サービスの管理

TruCluster Server は次のメール・プロトコルをサポートしています。

- SMTP (シンプル・メール転送プロトコル)
- DECnet Phase IV
- DECnet/OSI
- MTS (メッセージ・トランスポート・システム)
- UUCP (UNIX 間コピー・プログラム)
- X.25

SMTP はクラスタ単位でクラスタ別名を使用することができます。他のメール・プロトコルはクラスタ環境で実行できますが、各クラスタ・メンバがスタンドアロン・システムであるかのように動作します。

クラスタでは、すべてのメンバのメール・サービスの構成を同一にしなければなりません。DECnet, SMTP, またはその他の任意のプロトコルを 1 つのメンバ上で構成した場合は、そのプロトコルを他のすべてのメンバ上でも構成する必要があります。また、そのプロトコルの構成は各メンバ上で同一でなければなりません。クラスタは、メール・サーバまたはメール・クライアントにするか、スタンドアロン構成にできます。ただし、構成の適用対象は常にクラスタ全体です。たとえば、あるメンバをメール・クライアントとして構成し、別のメンバをメール・サーバとして構成することはできません。

サポートされているプロトコルのうち SMTP だけがクラスタ対応なので、クラスタ別名を使用できるのは SMTP のみです。SMTP は、クラスタ別名宛の電子メールを処理し、返信アドレスとしてクラスタ別名を送信メールに付けます。

sendmail が構成されると、そのインスタンスはクラスタの各メンバ上で動作します。すべてのメンバは、メール・キュー・ファイルを共用するので、

処理待ちのメッセージを処理できます。また、各ユーザのメール・ボックスも共用するので、ローカルで配信されたメールを処理できます。

その他のメール・プロトコル (DECnet Phase IV, DECnet/OSI, MTS, UUCP, および X.25) もクラスタ環境で動作できます。しかしこれらのプロトコルからは、クラスタの各メンバがスタンドアロン・システムであるかのように見えます。これらのプロトコルの 1 つを使用する受信電子メールには、宛先アドレスとしてクラスタ別名ではなく、宛先のクラスタ・メンバのアドレスが書き込まれます。これらのプロトコルの 1 つを使用する送信電子メールには、返信アドレスとして送信元のクラスタ・メンバのアドレスが書き込まれます。

クラスタでの DECnet Phase IV, DECnet/OSI, MTS, UUCP, または X.25 の構成は、スタンドアロン・システムでの構成に似ています。この構成は、クラスタの各メンバ上で行わなければなりません。プロトコルに必要なハードウェアは、クラスタの各メンバにインストールされていなければなりません。

以降の各項で、メール・サービスの管理についてより詳しく説明します。

## 7.8.1 メール・サービスの構成

メール・サービスの構成には、`mailsetup` または `mailconfig` コマンドを使用します。各コマンドにはそれぞれ固有の構成形式があるので、どちらのコマンドを選択しても、クラスタ内で以降に行うメール・サービスの構成では、選択した同じコマンドを使用しなければなりません。

### 7.8.1.1 メール・ファイル

次のメール・ファイルはクラスタ全体で共用される共通ファイルです。

- `/usr/adm/sendmail/sendmail.cf`
- `/usr/adm/sendmail/aliases`
- `/var/spool/mqueue`
- `/usr/spool/mail/*`

次のメール・ファイルはメンバ固有のファイルです。

- `/usr/adm/sendmail/sendmail.st`
- `/var/adm/sendmail/protocols.map`

`/var/adm/sendmail` 内で、ファイル名に `hostname` を含むファイルは、`hostname` の代わりに省略時のクラスタ別名を使用します。たとえば、クラスタ別名が `accounting` の場合は、`/var/adm/sendmail` には `accounting.m4` と `Makefile.cf.accounting` という名前のファイルが格納されます。

メールの統計情報ファイル (`/usr/adm/sendmail/sendmail.st`) がメンバに固有なので、メールの統計情報はクラスタの各メンバに固有です。`mailstat` コマンドは、その実行元のメンバだけにに関するメールの統計情報を返します。

SMTP 以外のメール・プロトコルを構成すると、メンバ固有の `/var/adm/sendmail/protocols.map` ファイルには、使用中のプロトコルに関するメンバ固有の情報が保存されます。`protocols.map` には、プロトコルのリストに加えて、DECnet Phase IV および DECnet/OSI で使用される別名のリストも保存されます (これらのプロトコルが構成されている場合のみ)。

#### 7.8.1.2 Cw マクロ (システム・ニックネーム・リスト)

メール・サービスの構成に使用したのが `mailsetup` であっても `mailconfig` であっても、その構成プログラムは `sendmail.cf` ファイル内の `Cw` マクロ (ニックネーム・リスト) に、クラスタのすべてのメンバの名前とクラスタ別名を自動的に追加します。ニックネーム・リストにこれらの名前は必須です。メール・サービスの構成中に、ニックネーム・リストからクラスタ別名またはメンバ名を不用意に削除してしまっても、この構成プログラムによって、削除された名前は自動的に再追加されます。

メール・サービスの構成中にクラスタのニックネームを追加できます。ただし、`mailsetup` のクイック・セットアップを実行した場合は、ニックネーム・リストの更新は要求されません。クラスタのメンバの名前とクラスタ別名は `Cw` マクロに自動的に追加されます。

#### 7.8.1.3 クラスタ作成時のメール・サービスの構成

`clu_create` コマンドの実行前に Tru64 UNIX システム上でメール・サービスの構成を行うことをお勧めします。SMTP のみを実行している場合は、クラスタに新規メンバを追加するたびに、そのメンバのメール・サービスを構成する必要はありません。`clu_add_member` コマンドによって、新規メンバのメール・サービスは自動的に正しく構成されます。

DECnet Phase IV, DECnet/OSI, MTS, UUCP, または X.25 を構成した場合は, クラスタに新規メンバを追加するたびに, `mailsetup` または `mailconfig` を実行し, 新規メンバ上でプロトコルを構成しなければなりません。

#### 7.8.1.4 クラスタ稼働後のメール・サービスの構成

すべてのメンバのメール・サービスの構成は同一にしなければなりません。SMTP のみを実行する場合は, この構成は一度行うだけで済みます。クラスタの任意のメンバから `mailsetup` または `mailconfig` を実行できます。

SMTP 以外のプロトコルを実行する場合は, 各メンバ上で `mailsetup` または `mailconfig` を実行し, プロトコルを構成しなければなりません。各メンバには, プロトコルに必要なハードウェアをインストールしておく必要があります。プロトコルの構成は, クラスタの各メンバに対して行い, すべてのメンバ上で同一でなければなりません。

コマンド `mailsetup` および `mailconfig` では, クラスタの特定のメンバにフォーカスを指定できません。SMTP の場合, これらのコマンドはクラスタ全体を対象にメール・サービスを構成します。その他のメール・プロトコルの場合, これらのコマンドは実行元のメンバのみを対象にメール・プロトコルを構成します。

`-focus` オプションを付けて `mailsetup` を実行しようとする, 次のエラー・メッセージが表示されます。

```
Mail can only be configured for the entire cluster.
```

SMTP 以外のメール・プロトコルを実行している場合は, クラスタに新規メンバを追加するたびに, `mailconfig` または `mailsetup` を実行し, 新規メンバ上でプロトコルを構成しなければなりません。SMTP のみを実行している場合は, 新規メンバを追加するたびに, メール・サービスは自動的に構成されます。

クラスタからメンバを削除したときは, 実行中のプロトコルにかかわらず, メール・サービスの再構成は必要ありません。

#### 7.8.2 クラスタ・メンバ間でのメールの負荷分散

SMTP によって処理されるメールの負荷分散を行うには, クラスタ別名の選択優先順位 (`selp`) と選択重み (`selw`) を使って, クラスタのメンバ間で

ネットワーク接続の負荷分散を行います。選択優先順位と選択重みは、次のような仕組みになっています。

- 最高を選択優先順位のクラスタ・メンバはすべての接続要求を受信します。

選択優先順位の値として 1 ~ 100 のいずれかの整数を指定できます。省略時の値は 1 です。

- 選択重みによって、同じ選択優先順位のメンバ間での接続要求の分配が決まります。メンバは、平均して選択重みと同数の接続要求を受信します。その後、後続の接続要求は同じ優先順位の次のメンバに送信されます。

選択重みの値として 0 ~ 100 のいずれかの整数を指定できます。選択重みが 0 のメンバは、要求の受信はしませんが、送信はできます。

省略時の設定では、クラスタのすべてのメンバには、各メンバの `/etc/clu_alias.config` ファイル内で設定された値と同じ選択優先順位 (`selpr=1`) と選択重み (`selw=1`) が割り当てられます (`clu_create` コマンドは省略時の選択重み 3 を使用しますが、別名を作成する場合は省略時の選択重みが 1 になります)。すべてのメンバが同じ選択優先順位と選択重みを共有するとき、接続要求はメンバ間で均等に分配されます。省略時のシステム構成では、各メンバが順番に 1 つの接続要求を受信します。

受信メール (およびその他すべての接続) を一部のクラスタ・メンバだけに処理させたい場合は、それらのクラスタ・メンバの選択優先順位を、その他のメンバの選択優先順位より高い値に統一します。

また、メールを処理させるクラスタ・メンバだけで新しいメール別名を作成することも、すべてのメンバを含むメール別名を作成し、選択優先順位を用いて、新しい接続要求を受信するメンバの順序を決定することもできます。

メンバの選択重みまたは選択優先順位を設定するには、そのメンバ上で `cluamgr` コマンドを実行します。メンバの `selpr` と `selw` に省略時の値が設定されている場合に、すべてのメール (およびその他のすべての接続要求) がクラスタの 1 つのメンバによって受信されるようにするには、そのメンバにログインして、`selpr` に省略時の値より大きい値を設定します。たとえば、次のコマンドを入力します。

```
# cluamgr -a alias=DEFAULTALIAS,selpr=50
```



8 メンバ構成のクラスタがあり、2 つのメンバ alpha および beta にすべての接続要求を受信させるとします。このとき、alpha と beta との間の負荷分散を 40/60 にするとします。そのためには、まず alpha にログインして、次のコマンドを入力します。

```
# cluamgr -a alias=DEFAULTALIAS,selp=50,selw=2
```

その後、beta にログインして、次のコマンドを入力します。

```
# cluamgr -a alias=DEFAULTALIAS,selp=50,selw=3
```

その他のメンバの selp に省略時の値の 1 が設定されているとすると、beta と alpha はすべての接続要求を受信します。まず beta が 3 つの接続要求を受信し、次に alpha が 2 つの接続要求を受信し、次に beta が 3 つの接続要求を受信する、という具合に続きます。

---

#### 注意

---

selp と selw をこのように設定すると、メールのトラフィックだけでなく、クラスタ別名を介するすべての接続に影響します。

---

接続要求の分配についての詳しい説明は、3.10 節および cluamgr(8) を参照してください。

## 7.9 RIS 用のクラスタの構成

クラスタ内に RIS (Remote Installation Services) サーバを作成するには、Tru64 UNIX 『*Sharing Software on a Local Area Network*』で説明した手順に加えて、次の手順を実行します。

- /etc/bootptab を修正して、NFS マウント・ポイントとして省略時のクラスタ別名を設定する。
- tftp サーバのアドレスとして省略時のクラスタ別名を設定する。

```
sa=default_cluster_alias
```

/etc/bootptab については、bootptab(4) を参照してください。

---

## 注意

---

ネットワークの構成に応じて、RIS サーバにクラスタ別名を登録するときは、一意かつ任意のハードウェア・アドレスを指定する必要があります。

---

RIS クライアントとしてクラスタを使用するには、次のことを行う必要があります。

1. RIS サーバに `setld` コマンドを使用して、クラスタ・メンバを登録します。これは、メンバ名とそのメンバのハードウェア・アドレスを登録することにより行います。
2. 省略時のクラスタ別名を登録します。

オペレーティング・システム・キットに対して登録する場合には、ハードウェア・アドレスの入力が必要となります。クラスタ別名には、ホスト名と対応した物理インタフェースがありません。代わりに、`/etc/bootptab` または `/usr/var/adm/ris/clients/risdb` にまだ存在しない物理アドレスならどれでも使用できます。

クラスタでクラスタ別名仮想 MAC (vMAC) 機能を使用している場合は、RIS サーバに省略時のクラスタ別名のハードウェア・アドレスとして仮想ハードウェア・アドレスを登録します。クラスタでクラスタ別名仮想 MAC 機能を使用していない場合でも、3.12 節に記述されたアルゴリズムを使用して仮想アドレスを生成することができます。

仮想 MAC アドレスは、接頭部 (省略時は AA:01) と、別名に対する 16 進数形式の IP アドレスで構成されます。たとえば、省略時のクラスタ別名が `deli` で、その IP アドレスが `16.140.112.209` である場合、クラスタの省略時の別名に対する vMAC アドレスは `AA:01:10:8C:70:D1` で表されます。アドレスは次のような方法で決定されます。

省略時の vMAC 接頭部:	AA:01
クラスタ別名 IP アドレス:	16.140.112.209
IP アドレス (16 進形式):	10.8C.70:D1
このアドレスの vMAC:	AA:01:10:8C:70:D1

そのため、省略時のクラスタ別名を RIS クライアントとして登録するとき、ホスト名は `deli` で、ハードウェア・アドレスは `AA:01:10:8C:70:D1` となります。

省略時のクラスタ別名とそのメンバの両方を登録していない場合，setld コマンドは，次のようなメッセージのいずれかを返します。

```
# setld -l ris-server:setld: Error contacting server
ris-server: Permission denied.
setld: cannot initialize ris-server:
```

または

```
# setld -l ris-server:setld: ris-server: not in server database
setld: cannot load control information
```

## 7.10 リモートへの X Window アプリケーションの表示

クラスタを構成すると，クラスタ外のシステムのユーザが，クラスタ上で X アプリケーションを実行し，クラスタ別名を使用してユーザのシステム上に X アプリケーションを表示できます。

次の例は，クラスタ・メンバから表示される X アプリケーションを単一システムのアプリケーションのように処理する手段として out\_alias を使用する方法を示しています。

/etc/clua\_services で，out\_alias 属性が X サーバ・ポート (6000) 用に設定されます。クラスタ外のシステムのユーザは，クラスタ・メンバ上の X アプリケーションを実行し，表示はユーザのシステム上に行う必要があります。out\_alias 属性はクラスタ内のポート 6000 に対して設定されるので，ユーザは，xhost コマンドの実行時に省略時のクラスタ別名を指定して，X クライアントがユーザのローカル・システムにアクセスできるようにする必要があります。たとえば，deli という名前のクラスタの場合，ユーザはローカル・システム上で次のコマンドを実行します。

```
# xhost +deli
```

このように out\_alias を使用すれば，どのクラスタ・メンバの X アプリケーションでも，そのユーザのシステム上に表示されるようにすることができます。クラスタ管理者は，メンバ単位にユーザがアクセスできるようにしたい場合には，/etc/clua\_services の Xserver の行をコメントにするか，その行から out\_alias 属性を削除します(その後で，各クラスタ・メンバ上で cluamgr -f を実行して，変更を有効にします)。

クラスタ別名の詳細については，第 3 章を参照してください。



---

## 高可用性アプリケーションの管理

この章では、CAA (Cluster Application Availability) サブシステムによる高可用性アプリケーションの管理について取り上げ、次の項目について説明します。

- リソースの状態の調査 (8.1 節)
- リソースの可用性計測レポート (8.2 節)
- アプリケーションの再配置 (8.3 節)
- アプリケーション・リソースの起動と停止 (8.4 節)
- アプリケーションの平衡 (8.5 節)
- アプリケーション・リソースの登録と登録取り消し (8.6 節)
- ネットワーク、テープ、メディア・チェンジャ・リソースの管理 (8.7 節)
- SysMan を使用した CAA の管理 (8.8 節)
- 起動時およびシャットダウン時の CAA サブシステムに関する考慮事項の理解 (8.9 節)
- CAA デーモン `caad` の管理 (8.10 節)
- EVM による CAA のイベントの表示 (8.11 節)
- イベントを使用したトラブルシューティング (8.12 節)
- コマンド行メッセージを使用したトラブルシューティング (8.13 節)

CAA サブシステムによるアプリケーションのセットアップについての詳細は、TruCluster Server 『クラスタ高可用性アプリケーション・ガイド』を参照してください。CAA サブシステムについての概説は、TruCluster Server 『クラスタ概要』を参照してください。

CAA サブシステムの制御下で動作する高可用性アプリケーションであれば、その管理は自動的に行われます。ただし、高可用性アプリケーションの管理を手動で行うことが必要な場合もあります。その場合には、たとえば次のような状況が発生します。

- クラスタ・メンバのシャットダウンまたはリブート  
シャットダウンするメンバ上でどの高可用性アプリケーションが動作しているかを調べ (caa\_stat を使用), そのメンバ上のアプリケーションを手動で再配置できます (caa\_relocate を使用)。
- 負荷分散  
クラスタのさまざまなメンバ上の負荷の変化に応じて, 負荷が軽めのメンバにアプリケーションを手動で再配置します (caa\_stat と caa\_relocate を使用)。caa\_balance を実行してアプリケーションの配置をチェックして, 優先度の高い別のクラスタ・メンバが利用可能な場合に限り, 再配置します。
- 新規アプリケーション・リソースのプロファイルの作成  
未登録, 未起動のアプリケーション・リソースのプロファイルを作成します (caa\_register と caa\_start を使用)。
- 既存アプリケーション・リソースのプロファイルの更新  
アプリケーション・リソースを更新し, 更新を有効にします (caa\_register -u を使用)。
- 既存アプリケーション・リソースの廃止  
廃止予定のアプリケーション・リソースを停止し, 登録を取り消します (caa\_stop と caa\_unregister を使用)。

アプリケーション・リソースを使っでの作業では, リソースの名前がそのリソースに関連付けられたアプリケーションの実際の名前と必ずしも一致するわけではありません。アプリケーション・リソースの名前は, リソース・プロファイルのルート名と一致します。たとえば, cluster\_lockd という名前のアプリケーション・リソースがあるとすると, このリソースのプロファイルは /var/cluster/caa/profile/cluster\_lockd.cap です。このリソースに関連付けられたアプリケーションは rpc.lockd と rpc.statd です。

このようにアプリケーション・リソースの名前とそのリソースに関連付けられたアプリケーションの名前が異なる場合があるので, アプリケーション名を使って, クラスタ内で動作するプロセスのリスト内でリソースを検索しても見つからない場合があります。CAA サブシステムを使ってアプリケーションを管理するときは, リソース名を使用しなければなりません。

## 8.1 リソースの状態の調査

登録済みのリソースは、次の3つのうちのいずれかの状態になります。

- ONLINE

アプリケーション・リソースの場合、そのリソースに関連付けられたアプリケーションは正常に動作しています。

ネットワーク、テープ、またはメディア・チェンジャ・リソースの場合、そのリソースに関連付けられたデバイスは使用可能であり、正常に機能しています。

- OFFLINE

そのアプリケーション・リソースは動作していません。そのリソースは、既に登録されていますが `caa_start` によって起動されなかったか、`caa_stop` によって早期に正常に停止された可能性があります。ネットワーク、テープ、またはメディア・チェンジャ・リソースの場合、そのリソースに関連付けられたデバイスが正常に機能していません。さらに、プロファイル内の `FAILURE_THRESHOLD` の値よりも多くの回数のエラーがそのリソースに発生した場合も、この状態になります。

- UNKNOWN

CAA はそのアプリケーションが実行中であるか、リソースの処理スクリプトの停止エントリ・ポイントの実行の失敗によるか、判断できません。この状態は、アプリケーション・リソースにのみ適用されます。リソースの処理スクリプトにある停止エントリ・ポイントを調べて、失敗 (0 以外の値が返される) の原因を調べます。

CAA は常に、アプリケーション・リソースの状態をそのターゲット状態に一致させようとします。ターゲット状態は、`caa_start` を使用すると `ONLINE` に設定され、`caa_stop` を使用すると `OFFLINE` に設定されます。ターゲット状態が、アプリケーション・リソースの状態と等しくない場合は、CAA によりアプリケーションを起動または停止している途中か、アプリケーションの実行または起動に失敗したかのいずれかです。非アプリケーション・リソースのターゲット状態が常に `OFFLINE` の場合、そのリソースは障害しきい値内で何度も障害が発生しています。詳細については、8.7 節を参照してください。

ターゲット・フィールドと状態フィールドの情報から、リソースの状態を判断することができます。その2つのフィールドの組み合わせの意味は表 8-1 (アプリケーション)、表 8-2 (ネットワーク)、表 8-3 (テープ、メディ

ア・チェンジャ) に示されています。両方とも ONLINE でないリソースの状態とターゲットの組み合わせがある場合、すべてのリソースは OFFLINE の状態になります。

表 8-1: アプリケーション・リソースのターゲットと状態の組み合わせ

ターゲット	状態	説明
ONLINE	ONLINE	アプリケーションが正常に開始した。
ONLINE	OFFLINE	起動コマンドが発行されたが、処理スクリプトの起動エントリ・ポイントの実行がまだ完了していない。  必要なリソースの障害のため、アプリケーションが停止した。  新クラスタ・メンバの開始または追加のため、アプリケーションが有効な配置状態で、再配置中である。  明示的再配置またはクラスタ・メンバの障害のため、アプリケーションが再配置中である。  アプリケーションを開始する適切なメンバが使用できない。
OFFLINE	ONLINE	停止コマンドが発行されたが、処理スクリプトの停止エントリ・ポイントの実行がまだ完了していない。
OFFLINE	OFFLINE	アプリケーションがまだ起動されていない。  障害のしきい値に到達したため、アプリケーションが停止した。  アプリケーションが正常に停止した。
ONLINE	UNKNOWN	処理スクリプトの停止エントリ・ポイントが障害を返した。
OFFLINE	UNKNOWN	状態が UNKNOWN のアプリケーションでアプリケーション停止コマンドが発行された。処理スクリプトの停止エントリ・ポイントはまだ障害を返す。アプリケーション状態を OFFLINE に設定するには、 <code>caa_stop -f</code> を使用する。

表 8-2: ネットワーク・リソースのターゲットと状態の組み合わせ

ターゲット	状態	説明
ONLINE	ONLINE	ネットワークが正常に機能している。
ONLINE	OFFLINE	クラスタ・メンバはネットワークへの直接的な接続を持たない。



表 8-2: ネットワーク・リソースのターゲットと状態の組み合わせ (続き)

ターゲット	状態	説明
OFFLINE	ONLINE	障害のしきい値に到達したため、ネットワーク・カードの障害と想定され、CAA で監視されなくなる。
OFFLINE	OFFLINE	ネットワークとマシンとの直接接続が不可である。 障害のしきい値に到達したため、ネットワーク・カードの障害と想定され、CAA で監視されなくなる。

表 8-3: テープ、メディア・チェンジャ・リソースのターゲットと状態の組み合わせ

ターゲット	状態	説明
ONLINE	ONLINE	テープまたはメディア・チェンジャはマシンと直接的に接続され、正常に機能している。
ONLINE	OFFLINE	リソースに関連したテープ装置またはメディア・チェンジャがイベント・マネージャ (EVM) イベントを送信し、正常に動作しなくなる。リソースには障害があると考えられる。
OFFLINE	ONLINE	障害のしきい値に到達したため、テープ装置またはメディア・チェンジャの障害と想定され、CAA で監視されなくなる。
OFFLINE	OFFLINE	テープ装置またはメディア・チェンジャがクラスタ・メンバへ直接接続できない。

### 8.1.1 リソースの状態の調査

リソースの状態を調べるには、次のように `caa_stat` コマンドを入力します。

```
# caa_stat resource_name
```

このコマンドのリターン値は次のとおりです。

- NAME  
リソース・プロファイルの NAME フィールドに指定されたリソース名が表示されます。
- TYPE  
リソースの種類 (application, tape, changer, または network) が表示されます。

- TARGET

アプリケーション・リソースの場合は、状態の ONLINE または OFFLINE が表示されます。CAA はアプリケーションをその状態にしようとしません。その他の種類のリソースの場合は、そのリソースに関連付けられたデバイスの障害回数が障害しきい値より高くない限り、ターゲットは常に ONLINE になります。高い場合、TARGET は OFFLINE になります。

- STATE

アプリケーション・リソースの場合は、ONLINE または OFFLINE が表示されます。リソースがオンラインのときは、現在稼働中のクラスタ・メンバの名前も表示されます。処理スクリプトの停止エントリ・ポイントで障害が返された場合、アプリケーションの状態は UNKNOWN にもなります。アプリケーション・リソースは、正常に停止するまで、アクティブになれません。その他の種類のリソースの場合は、クラスタのメンバごとに ONLINE または OFFLINE の状態が表示されます。

このコマンドの実行例は次のとおりです。

```
# caa_stat clock
NAME=clock
TYPE=application
TARGET=ONLINE
STATE=ONLINE on provolone
```

スクリプトを使ってリソースがオンラインかどうかを調べるには、`caa_stat` コマンドに `-r` オプションを使用します。たとえば、次のように記述します。

```
# caa_stat resource_name -r ; echo $?
```

リソースが ONLINE 状態の場合は、値 0 (ゼロ) が返ります。

スクリプトを使ってアプリケーション・リソースが登録済みかどうかを調べるには、`caa_stat` コマンドに `-g` オプションを使用します。たとえば、次のように記述します。

```
# caa_stat resource_name -g ; echo $?
```

リソースが登録済みの場合は、値 0 (ゼロ) が返ります。

### 8.1.2 特定のクラスタ・メンバ上のすべてのリソースの調査

`caa_stat -c cluster_member` コマンドは、`cluster_member` 上のすべてのリソースの状態を返します。このコマンドの実行例は次のとおりです。

```
# caa_stat -c polishham  
NAME=dhcp  
TYPE=application  
TARGET=ONLINE  
STATE=ONLINE on polishham
```

```
NAME=named  
TYPE=application  
TARGET=ONLINE  
STATE=ONLINE on polishham
```

```
NAME=xclock  
TYPE=application  
TARGET=ONLINE  
STATE=ONLINE on polishham
```

このコマンドが役立つのは、クラスタのメンバをシャットダウンする前に、どのアプリケーションをフェイルオーバーまたは手動再配置の対象にするか調べるときです。

### 8.1.3 すべてのクラスタ・メンバ上のすべてのリソースの調査

`caa_stat` コマンドは、すべてのクラスタ・メンバ上のすべてのリソースの状態を返します。このコマンドの実行例は次のとおりです。

```
# caa_stat  
NAME=dhcp  
TYPE=application  
TARGET=ONLINE  
STATE=ONLINE on polishham  
  
NAME=xclock  
TYPE=application  
TARGET=ONLINE  
STATE=ONLINE on provolone  
  
NAME=named  
TYPE=application  
TARGET=OFFLINE  
STATE=OFFLINE  
  
NAME=ln0  
TYPE=network  
TARGET=ONLINE on provolone  
TARGET=ONLINE on polishham  
TARGET=ONLINE on peppicelli  
STATE=OFFLINE on provolone  
STATE=ONLINE on polishham  
STATE=ONLINE on peppicelli
```

オプション `-t` を使用すると、表形式で情報が表示されます。たとえば、次のように表示されます。

```
# caa_stat -t
Name          Type          Target    State    Host
-----
cluster_lockd application  ONLINE   ONLINE  provolone
dhcp           application OFFLINE   OFFLINE
named          application OFFLINE   OFFLINE
ln0            network     ONLINE    ONLINE  provolone
ln0            network     ONLINE    OFFLINE  polishham
```

#### 8.1.4 障害回数と再起動回数およびターゲット状態の調査

`caa_stat -v` コマンドは、すべてのクラスタ・メンバ上のすべてのリソースの状態とともに、それらのリソースの障害回数と再起動回数も返します。このコマンドの実行例は次のとおりです。

```
# caa_stat -v
NAME=cluster_lockd
TYPE=application
RESTART_COUNT=0
RESTART_ATTEMPTS=30
REBALANCE=
FAILURE_COUNT=0
FAILURE_THRESHOLD=0
TARGET=ONLINE
STATE=ONLINE on provolone

NAME=dhcp
TYPE=application
RESTART_COUNT=0
RESTART_ATTEMPTS=1
REBALANCE=
FAILURE_COUNT=1
FAILURE_THRESHOLD=3
TARGET=ONLINE
STATE=OFFLINE

NAME=ln0
TYPE=network
FAILURE_THRESHOLD=5
FAILURE_COUNT=1 on provolone
FAILURE_COUNT=0 on polishham
TARGET=ONLINE on provolone
TARGET=OFFLINE on polishham
STATE=ONLINE on provolone
STATE=OFFLINE on polishham
```

-t オプションを使用すると、情報が表形式で表示されます。このコマンドの実行例は次のとおりです。

```
# caa_stat -v -t
Name      Type      R/RA  F/FT  Target  State  Host      Rebalance
-----
cluster_lockd application 0/30   0/0   ONLINE  ONLINE provolone
dhcp      application 0/1    0/0   OFFLINE OFFLINE
named     application 0/1    0/0   OFFLINE OFFLINE
ln0       network    0/5    0/5   ONLINE  ONLINE provolone
ln0       network    1/5    0/5   ONLINE  OFFLINE polishham
```

この情報によって、頻繁に障害が発生しているリソースや何度も再起動されたリソースを簡単に特定できます。

## 8.2 リソースの可用性計測レポート

CAA は最初に起動されてからの各アプリケーション・リソースの履歴を保持しています。caa\_report コマンドは、ONLINE 状態にあり、現在登録されているすべてのアプリケーション・リソースの ONLINE 状態であった時間の割合をまとめたレポートを出力します。このデータは、CAA 関連の Event Management イベントを分析した結果です。

アプリケーションの稼働履歴を追跡するファイルは、毎日、AM 3 時に自動的に更新されます。この時間に caa\_report コマンドがルートで実行されます。この定期的なマージを行う時間と頻度の変更は、/var/cluster/caa/clustercron/caa\_report.clustercronData で定義された crontab フォーマットを変更します。

このコマンドは、アプリケーションの開始時刻と、アプリケーションが ONLINE 状態であった時間 (総時間に対する割合) を示します。

出力例は次のとおりです。

```
#!/usr/bin/caa_report
Application Availability Report for rubble
Applications  starting/ending      uptime
-----
autofs        NEVER STARTED                0.00 %

cluster_lockd  Fri Jul 27 11:00:48 2001    99.80 %
               Thu Oct  4 12:31:14 2001

clustercron    Fri Jul 27 11:01:08 2001   100.00 %
               Thu Oct  4 12:31:14 2001

dmiller1      Tue Sep 25 13:57:51 2001    12.51 %
               Thu Oct  4 12:31:14 2001
```

指定された時間に実行されていないすべてのアプリケーションを対象外にできます。次に例を示します。

```
#/usr/bin/caa_report -o
Application Availability Report for rubble
Applications      starting/ending      uptime
-----
cluster_lockd     Fri Jul 27 11:00:48 2001    99.80 %
                  Thu Oct  4 12:31:14 2001
clustercron       Fri Jul 27 11:01:08 2001    100.00 %
                  Thu Oct  4 12:31:14 2001
dmiller1          Tue Sep 25 13:57:51 2001     12.51 %
                  Thu Oct  4 12:31:14 2001
```

開始時刻と終了時刻を指定して、その範囲で稼働時間の割合を測定をすることができます。開始時刻と終了時刻は、date(1) コマンド、または ISO 8061 で使用する形式で指定することができます。空白を含む時間指定形式の場合には、時間を二重引用符で囲む必要があります。

次のルールに従って時刻を指定します。

- 終了時刻は開始時刻より後にしなければならない。
- 開始時刻が指定されていない場合、アプリケーションが起動された最も早い時刻を使用する。
- 終了時刻が指定されていない場合、現時点の時刻を使用する。
- 終了時刻が現時点より以降に指定されている場合、現時点の時刻を使用する。
- 指定された開始時刻以前にアプリケーションが実行されていない場合、アプリケーションが起動された最初の時刻を使用する。
- 指定された開始時刻以前にアプリケーションが実行されていたが、指定開始時刻ではダウンしている場合、指定された開始時刻を使用する。
- 出力では、分析に使用された実時間を表示する。

開始時刻と終了時刻の指定例は、次のとおりです。

```
#/usr/bin/caa_report -b 12/03/01 -e 12/04/01
Application Availability Report for rubble
Applications      starting/ending      uptime
-----
autofs            NEVER STARTED        0.00 %
cluster_lockd     Wed Oct  3 00:00:00 2001    100.00 %
                  Thu Oct  4 00:00:00 2001
```

```

clustercron      Wed Oct  3 00:00:00 2001    100.00 %
                  Thu Oct  4 00:00:00 2001
dmiller1         Wed Oct  3 00:00:00 2001     92.54 %
                  Thu Oct  4 00:00:00 2001

```

## 8.3 アプリケーションの再配置

クラスタ・メンバ間でのアプリケーションの再配置では次のような状況が考えられます。

- クラスタの1つのメンバ上のすべてのアプリケーションを別のメンバに再配置する (8.3.1 項)
- クラスタの1つのメンバ上の特定のアプリケーションを別のメンバに再配置する (8.3.2 項)
- 依存関係にあるアプリケーションを別のメンバに再配置する (8.3.3 項)

アプリケーションを再配置するには、`caa_relocate` コマンドを使用します。アプリケーションを再配置すると必ず、システムから再配置を追跡するメッセージが返されます。たとえば、次のようなメッセージが返されます。

```

Attempting to stop 'cluster_lockd' on member 'provolone'
Stop of 'cluster_lockd' on member 'provolone' succeeded.
Attempting to start 'cluster_lockd' on member 'pepicelli'
Start of 'cluster_lockd' on member 'pepicelli' succeeded.

```

以降の各項で、アプリケーションの再配置についてより詳しく説明します。

### 8.3.1 クラスタ・メンバ上のすべてのアプリケーションの手動再配置

クラスタの1つのメンバをシャットダウンすると、そのメンバ上で動作する CAA 制御下のアプリケーションはすべて、各アプリケーションの配置ポリシーに従って、自動的に再配置されます。ただし次の理由から、メンバのシャットダウン前にそのメンバ上のすべてのアプリケーションの手動再配置が必要な場合があります。

- 複数のメンバをシャットダウンする場合は、即時シャットダウンするメンバにアプリケーションが自動的に再配置されないようにする必要があります。
- メンバに問題または障害が発生している場合は、そのメンバ上で動作するアプリケーション・リソースに対する性能劣化を最小にする必要があるため。

- 作業環境の崩壊を最小限にして、クラスタ・メンバ上で保守を行う場合。

member1 上のすべてのアプリケーションを member2 に再配置するには、次のコマンドを入力します。

```
# caa_relocate -s member1 -c member2
```

member1 上のすべてのアプリケーションを各アプリケーションの配置ポリシーに従って再配置するには、次のコマンドを入力します。

```
# caa_relocate -s member1
```

caa\_stat コマンドを使って、すべてのアプリケーション・リソースが正常に再配置されたことを確認します。

### 8.3.2 特定のアプリケーションの手動再配置

次の理由から、特定のクラスタ・メンバへの特定のアプリケーションの手動再配置が必要な場合があります。

- 特定のアプリケーションを現在実行中のクラスタ・メンバに過負荷がかかっているので、負荷が小さい別のメンバ上でそのアプリケーションを実行する必要があるため。
- クラスタのメンバをシャットダウンしようとしているが、再配置ポリシーによって選択されない可能性があるメンバ上で、特定のアプリケーションを実行する必要があるため。

member2 に特定のアプリケーションを再配置するには、次のコマンドを入力します。

```
# caa_relocate resource_name -c member2
```

caa\_stat コマンドを使って、アプリケーション・リソースが正常に再配置されたことを確認します。

### 8.3.3 依存関係にあるアプリケーションの手動再配置

相互依存するアプリケーションのグループの再配置が必要な場合があります。リソース・プロファイル内で REQUIRED\_RESOURCE フィールドに最低 1 つの他のアプリケーション・リソースが列挙指定されている場合、このプロファイルを持つアプリケーション・リソースは、列挙された他のアプリケーション・リソースと依存関係にあります。他のアプリケーション・リソースと依存関係にあるアプリケーション・リソースを再配置する場合



は、`caa_relocate` コマンドに `-f` オプションを使って手動再配置を行わなければなりません。

手動再配置を行うと、CAA デーモンは、指定されたアプリケーション・リソースおよびそれと依存関係にあるアプリケーション・リソースをすべて再配置します。つまり、指定されたアプリケーション・リソースとともに、指定されたリソースが依存するリソースも、指定されたリソースに依存する ONLINE 状態のリソースも再配置されます。依存関係が間接的な場合もあります。つまり、1 つのリソースが中間の 1 つ以上のリソースを介して別のリソースに依存している場合です。

特定のアプリケーション・リソースおよびそれと依存関係にあるアプリケーション・リソースを `member2` に再配置するには、次のコマンドを入力します。

```
# caa_relocate resource_name -f -c member2
```

`caa_stat` コマンドを使って、アプリケーション・リソースが正常に再配置されたことを確認します。

## 8.4 アプリケーション・リソースの起動と停止

以降の各項で、CAA アプリケーション・リソースの起動および停止方法について説明します。

### 注意

CAA が管理するアプリケーション・リソースの起動には `caa_start`、停止には `caa_stop`、または SysMan の同等のものを必ず使用してください。CAA に登録した後は、絶対に手動でアプリケーションを起動、停止しないでください。

### 8.4.1 アプリケーション・リソースの起動

アプリケーション・リソースを起動するには、起動するリソースの名目を後ろに指定して `caa_start` コマンドを実行します。アプリケーション・リソースを起動する前に、`caa_register` を使ってそのリソースを登録しておく必要があります。

caa\_start コマンドの実行後すぐにターゲットには ONLINE が設定されます。CAA のサブシステムがアプリケーションを起動し、状態をターゲットと同じものにします。リソースに関連するアプリケーションは、ターゲット状態が ONLINE に設定され、CAA サブシステムはこれらを起動しようとします。

リソースの再配置ポリシーによって選択されるクラスタ・メンバ上で、clock というアプリケーション・リソースを起動するには、次のコマンドを入力します。

```
# /usr/sbin/caa_start clock
```

たとえば、このコマンドの出力は次のようになります。

```
Attempting to start 'clock' on member 'polishham'  
Start of 'clock' on member 'polishham' succeeded.
```

このコマンドは、リソースの処理スクリプトが呼び出されるたびに、SCRIPT\_TIMEOUT の値に達するまで、そのスクリプトから成功または失敗の通知が来るのを待ちます。

クラスタの特定のメンバ上で clock というアプリケーション・リソースを起動するには (ただしそのメンバが再配置ポリシーによって許可されることが前提)、次のコマンドを入力します。

```
# /usr/sbin/caa_start clock -c member_name
```

指定したメンバが使用可能でない場合、そのリソースは起動されません。

指定したリソースと依存関係にあるリソースが、指定したメンバ上で使用可能でなく、起動できない場合、caa\_start の実行は失敗し、依存関係の問題でアプリケーション・リソースが起動できなかったというメッセージが表示されます。

指定したリソースおよびそれと依存関係にあるリソースをすべて強制的に起動または同じメンバへ再配置するには、次のコマンドを使用します。

```
# /usr/sbin/caa_start -f clock
```

---

#### 注意

---

システム・クラッシュが起こったクラスタ・メンバでアプリケーションを起動しようとする場合、caa\_start は中間結果を表示することができます。このシナリオでは、処理スクリプトの起動セクションが実行されますが、クラスタ・メンバは、コマンド行に開始の通知が表示される前にクラッシュします。caa\_start コマ

ンドは, `Remote start for [resource_name] failed on member [member_name]`. というエラーを表示して, 失敗状態を返します。アプリケーション・リソースは実際には `ONLINE` で, 別のメンバへのフェイルオーバーは, アプリケーションが間違ったメンバ上で起動されたかのように見えます。

そのメンバ上でアプリケーション・リソースの起動中にクラスタ・メンバが失敗した場合は, `caa_stat` でクラスタ上のリソースの状態をチェックして, そのリソースの状態を判断する必要があります。

---

詳細は, `caa_start(8)` を参照してください。

## 8.4.2 アプリケーション・リソースの停止

アプリケーション・リソースを停止するには, 停止するアプリケーション・リソースの名前を後ろに指定して `caa_stop` コマンドを使用します。前述したように, `kill` コマンドやその他の方法を使って, CAA 制御下のアプリケーション・リソースを停止しないでください。

`caa_stop` コマンドの実行後すぐにターゲットには `OFFLINE` が設定されます。CAA のサブシステムがアプリケーションを停止し, 状態をターゲットと同じにします。

たとえば, `clock` というアプリケーション・リソースを停止するには, 次のコマンドを使用します。

```
# /usr/sbin/caa_stop clock
```

指定したリソースと依存関係にある他のアプリケーション・リソースがある場合, このコマンドではそのアプリケーションは停止されません。代わりに, 依存関係の問題でアプリケーション・リソースが停止できなかったというメッセージが表示されます。指定したリソースおよびそれと依存関係にあるリソースをすべて強制停止するには, 次のコマンドを入力します。

```
# /usr/sbin/caa_stop -f clock
```

詳細は, `caa_stop(8)` を参照してください。

### 8.4.3 複数インスタンスが作成されないアプリケーション・リソース

複数または 1 つのメンバ上で同じアプリケーション・リソースに対して複数のコマンド `start` または `stop` (あるいはその両方) を同時に実行した場合、そのリソースにどのコマンドが反映されるかわかりません。ただし、`start` コマンドを複数回実行しても、アプリケーション・リソースのインスタンスは複数作成されません。

### 8.4.4 `caa_stop` を使用した UNKNOWN 状態のリセット

アプリケーション・リソースの状態が UNKNOWN に設定されている場合は、最初に `caa_stop` を実行してください。このコマンドでリソースが OFFLINE にリセットされない場合は、`caa_stop -f` コマンドを使用してください。このコマンドは、停止スクリプトで返されるあらゆるエラーを無視して、リソースを OFFLINE に設定し、アプリケーション・リソースを利用するすべてのアプリケーションを同様に OFFLINE に設定します。

アプリケーション・リソースを再起動する前に、処理スクリプトの停止エントリ・ポイントで確実にアプリケーションが正常に停止し、0 が返されるかどうかを確認してください。また、アプリケーションが現在実行されていない場合も、0 が返されることを確認してください。

## 8.5 アプリケーション・リソースの分散

アプリケーション・リソースの分散では、クラスタ上のリソースの現在の状態とリソースの配置ルールとに基づいてアプリケーション・リソースの配置を再評価します。アプリケーションの分散は、クラスタ単位として、メンバ単位として、または指定されたリソースで行われます。分散は、CAA 標準の配置決定メカニズムを使用し、負荷を考慮しないで決定されます。

アプリケーションの分散は、最も好ましいメンバに配置するか (`favored` または `restricted` の配置ポリシーを使用する場合)、または、より平等に、実行する CAA アプリケーション・リソースの数をクラスタ・メンバに割り当てます (`balanced` 配置ポリシーを使用する場合)。

あるクラスタ・メンバから別のクラスタ・メンバへのアプリケーションの再配置は、次の方法で行います。

- クラスタのすべてのアプリケーションを分散する (8.5.1 項)
- クラスタ・メンバのすべてのアプリケーションを分散する (8.5.2 項)

- 指定されたアプリケーションを分散する (8.5.3 項)
- 指定されたアプリケーションを特定の時刻で分散する (8.5.4 項)

caa\_balance コマンドは、アプリケーション・リソースだけに使用できません。ネットワーク、テープ、またはメディア・チェンジャ・リソースを分散することはできません。

クラスタ単位での分散では、クラスタ上のすべての ONLINE 状態のアプリケーション・リソースを再評価し、配置決定メカニズムによって選択されたクラスタ・メンバ上で動作していない場合、そのリソースを再配置します。

詳細は caa\_balance(8) を参照してください。

以降の各項で、アプリケーションの分散について詳しく説明します。

### 8.5.1 クラスタのすべてのアプリケーションを分散する

クラスタのすべてのアプリケーションを分散するには、次のコマンドを使用します。

```
# /usr/sbin/caa_balance -all
```

アプリケーション test と test2 は、ONLINE 状態のただ 2 つのアプリケーションで、balanced 配置ポリシーのメンバ rye 上で動作しているものとします。この場合には、次のように表示されます。

```
Attempting to stop 'test' on member 'rye'
Stop of 'test' on member 'rye' succeeded.
Attempting to start 'test' on member 'swiss'
Start of 'test' on member 'swiss' succeeded.
Resource test2 is already well placed
test2 is placed optimally. No relocation is needed.
```

ONLINE 状態のアプリケーションがクラスタにより多く存在する場合には、各アプリケーション・リソースで行う動作を反映して、出力も多くなります。

### 8.5.2 クラスタ・メンバのすべてのアプリケーションを分散する

クラスタ・メンバの rye で実行されるアプリケーションの配置を再評価するには、次のコマンドを使用します。

```
# /usr/sbin/caa_balance -s rye
```

アプリケーション test と test2 は、ONLINE 状態のただ 2 つのアプリケーションで、balanced 配置ポリシーのメンバ rye 上で動作しているものとします。この場合には、次のように表示されます。

```
Attempting to stop 'test' on member 'rye'
Stop of 'test' on member 'rye' succeeded.
Attempting to start 'test' on member 'swiss'
Start of 'test' on member 'swiss' succeeded.
Resource test2 is already well placed
test2 is placed optimally. No relocation is needed.
```

ONLINE 状態のアプリケーションがクラスタにより多く存在する場合には、各アプリケーション・リソースで行う動作を反映して、出力も多くなります。

### 8.5.3 指定されたアプリケーションを分散する

指定されたアプリケーションを分散するには、次のコマンドを使用します。

```
# /usr/sbin/caa_balance test test2
```

アプリケーション test と test2 は、balanced 配置ポリシーのメンバ rye 上で実行しているものとします。この場合には、次のように表示されます。

```
Attempting to stop 'test' on member 'rye'
Stop of 'test' on member 'rye' succeeded.
Attempting to start 'test' on member 'swiss'
Start of 'test' on member 'swiss' succeeded.
Resource test2 is already well placed
test2 is placed optimally. No relocation is needed.
```

### 8.5.4 アプリケーションを特定の時刻で分散する

単一アプリケーションを特定の時刻で分散するには、プロファイルに REBALANCE 属性を設定します。この機能は、決まった時刻でアプリケーションをフェイルバックさせるのに役立ちます。アプリケーション・リソースに対して REBALANCE 属性に特定の時刻を設定すると、そのアプリケーションはその時刻で最適なメンバに再配置されます。Active Placement 属性を使用しても、最適なメンバにフェイルバックされますが、フェイルバックが発生するのは、favored 配置ポリシーで指定されたメンバがクラスタに再び加わったときだけです。これは、アプリケーションの可用性を妨げたくない期間中に再配置を発生させることもあります。Active Placement 属性の代わりに REBALANCE 属性を使用すると、指定した時刻にアプリケーションが優先メンバにフェイルバックするだけです。

t:day:hour:min という形式で、プロファイルに時刻の値を指定する必要があります。ここで day は曜日 (0-6 で、日曜日が 0)、hour は時間 (0-23)、min は分 (0-59) で、この指定された時間で再評価が行われます。アスタリスクは、毎日、毎時間、毎分を指定するためにワイルドカードとして使用できます。

REBALANCE 属性は `clustercron` リソースを使用します。このリソースは、クラスタ単位の `cron` の CAA 固有の実装であり、次の URL からアクセスできる Best Practice ドキュメントの『*Using cron in a TruCluster Server Cluster*』に説明があります。 [http://www.tru64unix.com-paq.com/docs/best\\_practices/BP\\_CRON/TITLE.HTM](http://www.tru64unix.com-paq.com/docs/best_practices/BP_CRON/TITLE.HTM)。この固有の実装は、再分散のスケジュールのために CAA により内部的に使用され、その他の用途にはサポートされていません。

REBALANCE 属性の設定の方法の例は『クラスタ高可用性アプリケーション・ガイド』を参照してください。

## 8.6 アプリケーション・リソースの登録と登録取り消し

CAA がリソースを管理するには、そのリソースを CAA サブシステムに登録しておく必要があります。この登録作業はリソースごとに一度行うだけで済みます。

アプリケーション・リソースを登録するには、`/var/cluster/caa/profile` ディレクトリ内にそのリソースのプロファイルが必要です。リソース・プロファイルの作成手順については、TruCluster Server『クラスタ高可用性アプリケーション・ガイド』を参照してください。

クラスタ内のどのリソースが登録済みかを調べるには、次のように `caa_stat` コマンドを使用します。

```
# /usr/sbin/caa_stat
```

### 8.6.1 リソースの登録

アプリケーション・リソースを登録するには、次のように `caa_register` コマンドを使用します。

```
# caa_register resource_name
```

たとえば、`dtcalc` というアプリケーション・リソースを登録するには、次のコマンドを入力します。

```
# /usr/sbin/caa_stat dtcalc
```

登録するアプリケーション・リソースのプロファイルの

`REQUIRED_RESOURCES` 属性に、そのリソースと依存関係にあるリソースが列挙定義されている場合、この属性の列挙されたすべてのリソースをまず登録しなければなりません。

詳細については、`caa_register(8)` を参照してください。

## 8.6.2 リソースの登録取り消し

アプリケーション・リソースの登録を取り消して CAA サブシステムによる監視対象から削除することが必要な場合があります。アプリケーション・リソースの登録を取り消すには、まず最初に停止させ、リソースの状態を `OFFLINE` に変更する必要があります。アプリケーションの停止方法については、8.4.2 項を参照してください。

アプリケーション・リソースの登録を取り消すには、`caa_unregister` コマンドを使用します。たとえば、`dtcalc` というリソースの登録を取り消すには、次のコマンドを入力します。

```
# /usr/sbin/caa_unregister dtcalc
```

詳細は、`caa_unregister(8)` を参照してください。

SysMan Menu によるリソースの登録と登録取り消しについては、SysMan のオンライン・ヘルプを参照してください。

## 8.6.3 登録の更新

アプリケーション・リソースのプロファイルを変更した場合は、その登録を更新しなければならないことがあります。リソース・プロファイルについての詳細は、『クラスタ高可用性アプリケーション・ガイド』を参照してください。

リソースの登録を更新するには、`caa_register -u` コマンドを使用します。たとえば、`dtcalc` というリソースを更新するには、次のように入力します。

```
# /usr/sbin/caa_register -u dtcalc
```

---

### 注意

---

`caa_register -u` コマンドと SysMan Menu を使用すれば、`ONLINE` 状態のリソースのプロファイルにある `REQUIRED_RESOURCES` フィールドを更新して、`OFFLINE` 状態にあるリソースの名前を追加することができます。ただし、`REQUIRED_RESOURCES` フィールドを更新して `OFFLINE` 状態にあるアプリケーションを追加すると、システムがプロファイルと同期しなくなることがあります。このような更新を実行する場合



は、手動で必要なリソースを起動するか、更新したリソースを停止しなければなりません。

同じように、プロファイルにある `HOSTING_MEMBERS` リストの値を変更すると、その後の再配置と起動に影響します。  
ONLINE 状態のアプリケーション・リソースのプロファイルにある `HOSTING_MEMBERS` リストを更新して配置ポリシーを `restricted` にする場合は、そのリストに登録されているいずれかのクラスター・メンバでそのアプリケーションが実行中であることを確認してください。資格のあるどのメンバでもそのアプリケーションが実行されていないときは、`caa_register -u` を実行した後、そのアプリケーションに対して `caa_relocate` を実行します。

---

## 8.7 ネットワーク、テープ、およびメディア・チェンジャ・リソース

アプリケーション・リソースのみ `caa_stop` を使用して停止できます。ただし、非アプリケーション・リソースで、障害発生間隔の範囲でリソース障害しきい値を上回る障害が発生した場合は、`caa_start` を使用して該当するリソースを再起動できます。非アプリケーション・リソースを起動すると、その `TARGET` 値が `ONLINE` にリセットされます。これにより、このリソースに依存するアプリケーションはすべて起動されます。

ネットワーク、テープ、およびメディア・チェンジャ・リソースでは、ハードウェアの問題が原因で何度も障害が発生するおそれがあります。このようになった場合は、障害の発生するクラスター・メンバで CAA がそのデバイスを使用できないようにして、可能であれば、アプリケーション・リソースを再配置または停止してください。障害発生間隔の範囲で障害しきい値を上回ると、デバイスのリソースが使用できなくなります。リソースが使用できなくなった場合は、特定のクラスター・メンバ上のリソースの `TARGET` 状態が、`caa_stat resource_name` で示したように、`OFFLINE` に設定されます。たとえば、以下ようになります。

```
# /usr/sbin/caa_stat network1
NAME=network1
TYPE=network
TARGET=OFFLINE on provolone
TARGET=ONLINE on polishham
STATE=ONLINE on provolone
```

STATE=ONLINE on polishham

ネットワーク、テープ、またはチェンジャ・リソースの TARGET 状態が、障害発生間隔の範囲で障害回数が障害しきい値を上回ったために OFFLINE に設定された場合は、該当するリソースを利用するすべてのリソースの STATE が OFFLINE になります。ただし、TARGET は ONLINE のままです。このような依存関係にあるアプリケーションは、リソースが ONLINE になっている別のマシンに再配置されます。このリソースを ONLINE にしているクラスタ・メンバが存在しない場合、アプリケーションは、現在のメンバ上のリソースの STATE と TARGET が ONLINE になるまで、OFFLINE のままです。

非アプリケーション・リソースの TARGET 状態を ONLINE にリセットするには、`caa_start` (すべてのメンバの場合) コマンドまたは `caa_start -c cluster_member` コマンド (特定のメンバの場合) を使用します。処理が終わると、障害回数もゼロ (0) にリセットされます。

STATE の値が ONLINE になっている可能性があっても、障害回数が障害しきい値を上回ったために TARGET の値が OFFLINE に設定されている場合、リソースは、CAA によって OFFLINE であるかのように処理されます。

---

#### 注意

---

テープまたはメディア・チェンジャ・リソースが、クラスタの稼働中にデバイスが取り外された後や物理的な障害が発生した後にクラスタに再接続される場合、デバイスの再接続はクラスタで自動的に検出されません。したがって、`drdmgr -a DRD_CHECK_PATHdevice_name` コマンドを実行する必要があります。

---

## 8.8 SysMan を使用した CAA 管理

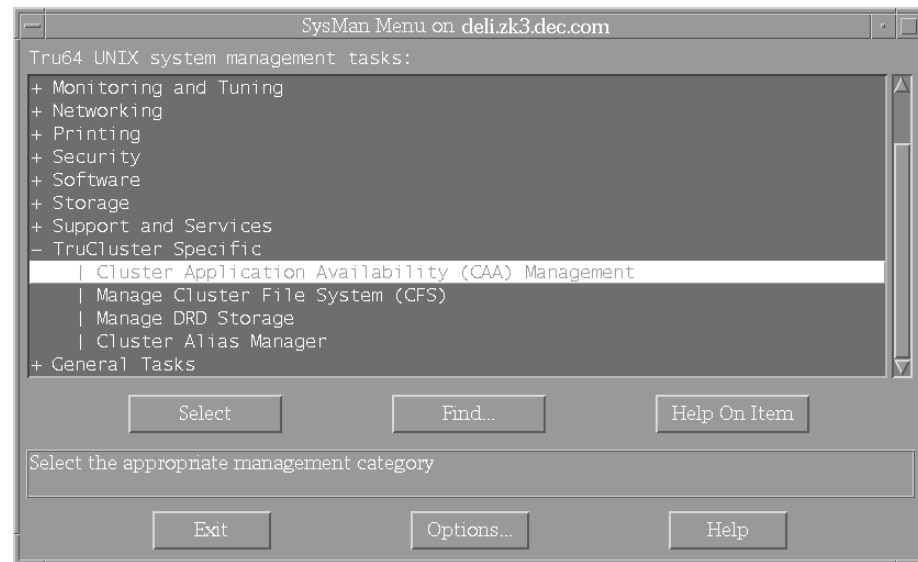
この節では、SysMan のツールを使用して CAA を管理する方法を説明します。SysMan の起動の一般的説明とクラスタでの使用方法については、第 2 章を参照してください。

### 8.8.1 SysMan Menu での CAA 管理

SysMan Menu から [Cluster Application Availability (CAA) Management] へのブランチは、図 8-1 で示すように、[TruCluster Specific] という見出しの

下にあります。[CAA Management] ダイアログ・ボックスを開くには、メニューの [Cluster Application Availability (CAA) Management] を選択し、[Select] ボタンをクリックするか、その文字列をダブルクリックします。

図 8-1: SysMan Menu の CAA へのブランチ



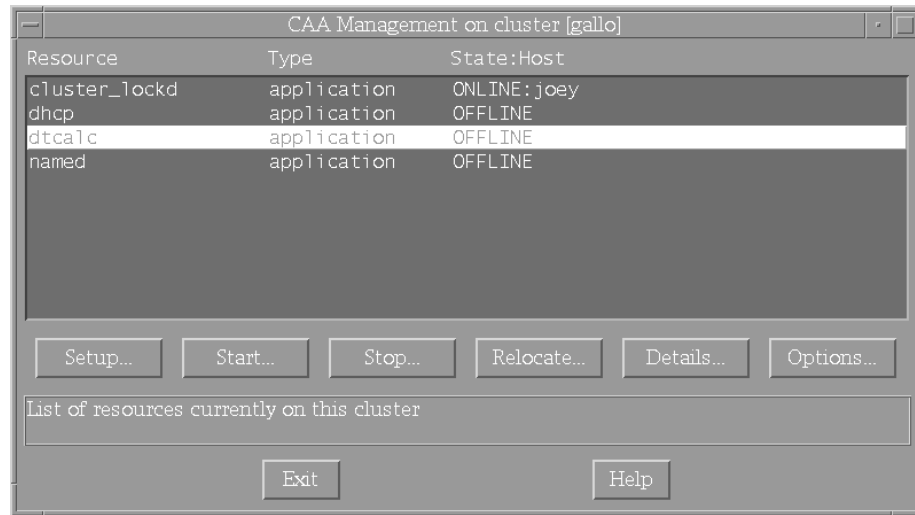
ZK-1756U-AI

#### 8.8.1.1 [CAA Management] ダイアログ・ボックス

[CAA Management] ダイアログ・ボックス (図 8-2) では、アプリケーションの起動、停止、再配置などを実行できます。アプリケーションを起動、再配置する場合、ダイアログ・ボックスにアプリケーションの配置についての問い合わせが出力されます。

また、[Setup] ダイアログ・ボックスを開いて、リソースの作成、変更、登録、登録取り消しなどを実行できます。

図 8-2: [CAA Management] ダイアログ・ボックス



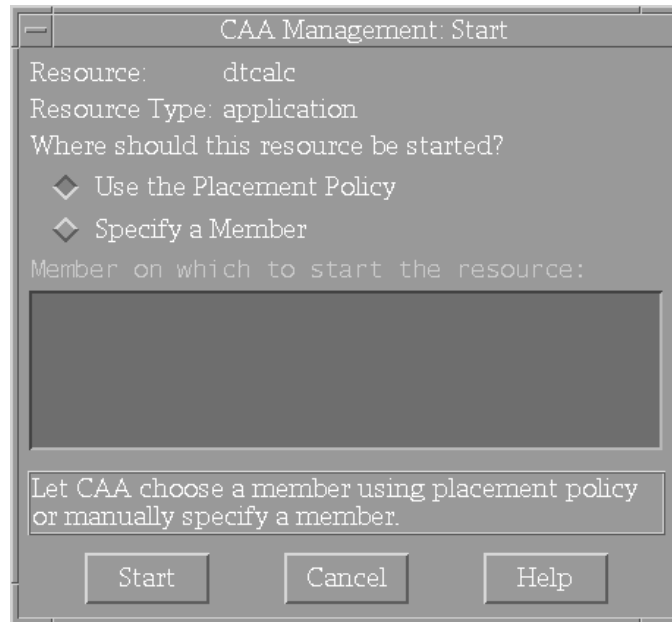
ZK-1755U-AI

#### 8.8.1.2 [Start] ダイアログ・ボックス

[Start] ダイアログ・ボックス (図 8-3) では、アプリケーション・リソースを配置ポリシーに従って配置するか、明示的に別のメンバ上に置くかを選択できます。

アプリケーションを明示的にメンバ上に配置できるのは、ホスト・メンバ・リストで許可されている場合のみです。配置ポリシーが `restricted` で、ホスト・メンバ・リストに含まれないメンバ上にアプリケーションを配置しようとする場合、起動は失敗します。

図 8-3: [Start] ダイアログ・ボックス

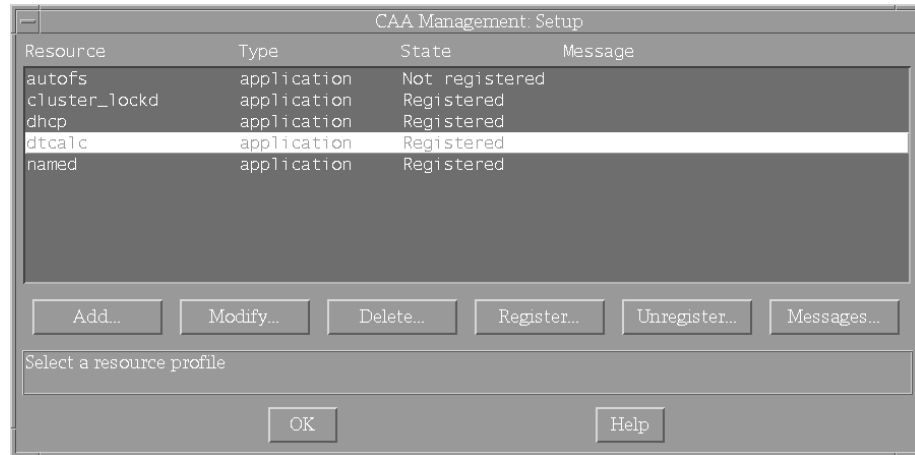


ZK-1757U-AI

#### 8.8.1.3 [Setup] ダイアログ・ボックス

プロファイルの追加，変更，登録，登録取り消しでは，図 8-4 に示す [Setup] ダイアログ・ボックスを使用します。このダイアログ・ボックスは，[CAA Management] ダイアログ・ボックスの [Setup...] ボタンから表示できます。SysMan Menu でリソースを設定する方法についての詳細は，オンライン・ヘルプを参照してください。

図 8-4: [Setup] ダイアログ・ボックス



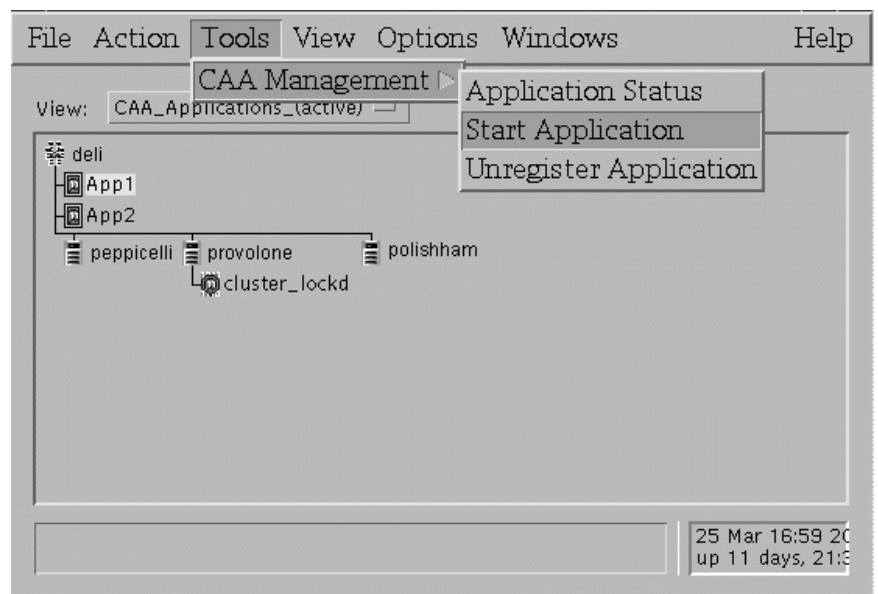
ZK-1758U-AI

## 8.8.2 SysMan Station での CAA 管理

SysMan Station を使用して CAA リソースを管理することができます。図 8-5 に SysMan Station の CAA\_Applications\_(active) ビューを示します。図 8-6 に SysMan Station の CAA\_Applications\_(all) ビューを示します。ウィンドウの上端にある [View] メニューを使用して、これらのビューのいずれかを選択します。クラスタ・アイコンまたはクラスタ・メンバ・アイコンを選択すると、CAA 固有の作業を含む [Tools] メニューの下のすべての SysMan Menu が使用可能になります。

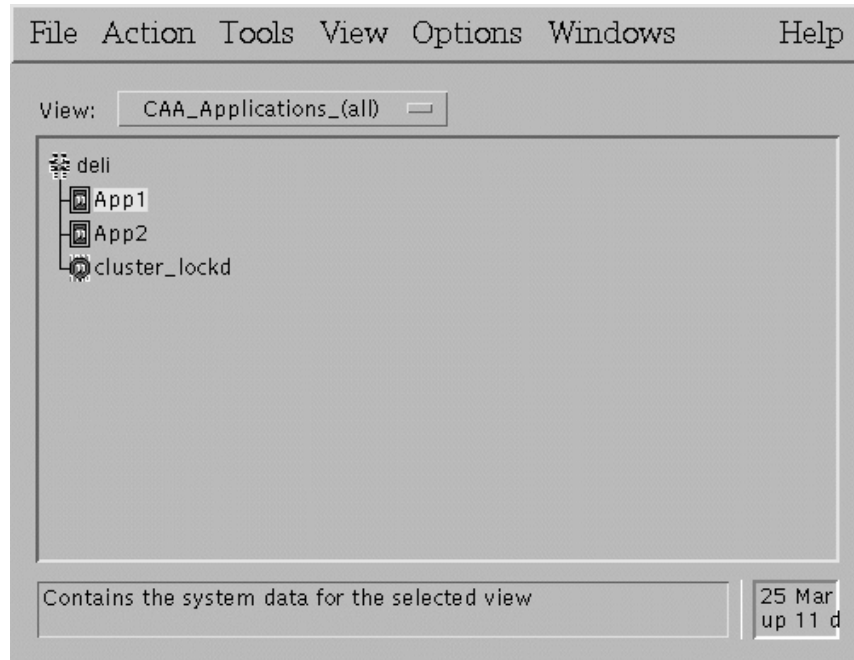
アプリケーション・リソースのアイコンは、リソースの状態を示します。これら 2 つの図では、App1 と App2 が現在オフラインで、cluster\_lockd がオンラインになっています。

図 8-5: SysMan Station の CAA\_Applications\_(active) ビュー



ZK-1753U-AI

図 8-6: SysMan Station の CAA\_Applications\_(all) ビュー



ZK-1752U-AI

#### 8.8.2.1 SysMan Station でのアプリケーションの起動

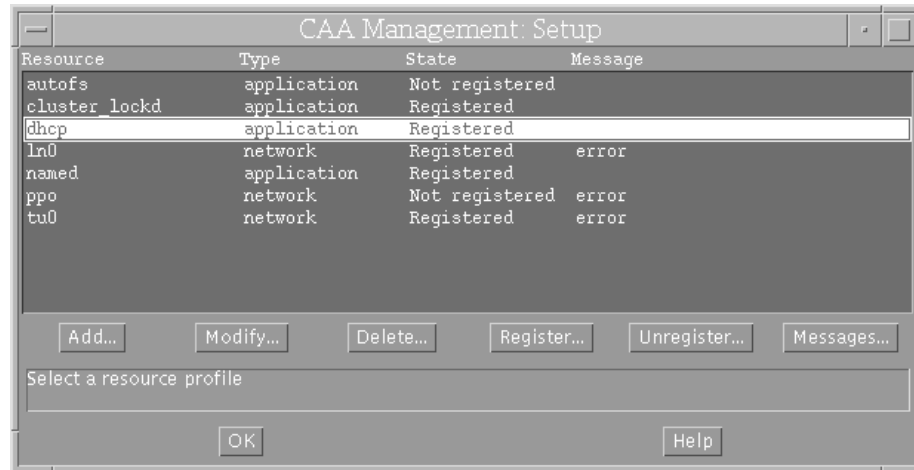
CAA\_Applications\_(active) ビュー (図 8-5) または CAA\_Applications\_(all) ビュー (図 8-6) のいずれかでアプリケーションを起動するには、[Tools] メニューでマウスの右ボタンをクリックして、[CAA Management Start Application] を選択します。

#### 8.8.2.2 SysMan Station でのリソース設定

SysMan Station を使用してリソースを設定するには、クラスタ・アイコンかクラスタ・メンバ・アイコンのどちらかを選択します。マウスの右ボタンをクリックするか、[Tool] メニューをクリックして、[CAA Management CAA Setup] を選択します。図 8-7 を参照してください。その後の手順は SysMan Menu の手順と同じで、オンライン・ヘルプの「使い方」の項で説明しています。



図 8-7: SysMan Station の [CAA Setup] 画面



ZK-1754U-AI

## 8.9 起動時およびシャットダウン時の CAA に関する考慮事項

CAA デーモンは、データベースから全リソースの情報を読み取る必要があります。そのため、多数のリソースが登録されている場合は、そのクラスタ・メンバはブートするのに時間がかかる場合があります。

CAA は、メンバのブート時に次のようなメッセージを表示する場合があります。

```
Cannot communicate with the CAA daemon.
```

このメッセージの前に次のようなメッセージが表示されるときもあります。

```
Error: could not start up CAA Applications
Cannot communicate with the CAA daemon.
```

これらのメッセージは、TrueCluster Server ライセンスを登録していませんを示しています。メンバのブートが終了したら、次のコマンドを入力します。

```
# lmf list
```

TCS-UA ライセンスが動作していない場合は、『クラスタ・インストール・ガイド』に記述されている方法によって登録してから、次のように CAA デーモン (caad) を起動します。

```
#!/usr/sbin/caad
```

クラスタのシャットダウン時、CAA デーモンは各アプリケーション・リソースが ONLINE であるか OFFLINE であるかを記録します。クラスタの再起動時に、ONLINE だったアプリケーションは再起動され、OFFLINE だったアプリケーションは再起動されません。UNKNOWN だったアプリケーションは中断状態とみなされます。クラスタの再起動によって解決されるような問題でアプリケーションが停止している場合は、caa\_start コマンドを使って、それらのアプリケーションを起動します。AUTO\_START が 1 に設定されているアプリケーションは、クラスタが再編成されたときに起動されます。

クラスタのメンバをシャットダウンする前にアプリケーションの配置を行う場合は、アプリケーション・リソースの状態を調べ、シャットダウンするメンバ上のあらゆるアプリケーションを別のメンバに再配置します。アプリケーションを再配置する方法は、8.3 節にリストされています。

## 8.10 CAA デーモンの管理

CAA デーモン (caad) の管理は、通常、自動的に行われます。CAA デーモンは、クラスタのあらゆるメンバ上で、ブート時に起動され、シャットダウン時に停止されます。ただし、このデーモンに問題が発生した場合は、作業が必要になります。

コマンド caa\_stat, caa\_start, caa\_stop, または caa\_relocate を実行したときに、「Cannot communicate with the CAA daemon!」というメッセージが表示された場合、caad デーモンはおそらく動作していません。CAA デーモンでエラーが発生した場合には、Essential Services Monitor デーモンが CAA デーモンの再起動を試みます。数秒たってから、CAA コマンドの実行を試みます。成功した場合には、デーモンが再起動され、CAA のすべての機能が正常に動作しています。デーモンが動作しているかどうかを手動で調べるには、8.10.1 項を参照してください。

### 8.10.1 ローカルの CAA デーモンの状態の確認

ローカルの CAA デーモンの状態を確認するには、次のコマンドを実行します。

```
# ps ax | grep -v grep | grep caad
```

caad が動作している場合、たとえば次のような出力が表示されます。

```
545317 ??          S          0:00.38 caad
```

何も表示されない場合は、caad が実行されていません。

クラスタの別のメンバにログインし、`ps ax | grep -v grep | grep caad` コマンドを実行することによって、他の `caad` デーモンの状態も確認できます。

マシン上で `caad` デーモンが動作していない場合、そのマシン上で起動されたアプリケーション・リソースは CAA で管理されなくなります。`caa_stop` を使ってアプリケーションを停止することはできません。8.10.2 項の説明に従って `caad` デーモンを再起動すると、そのマシン上のリソースは CAA サブシステムによって完全に管理されるようになります。

### 8.10.2 CAA デーモンの再起動

`caad` デーモンがクラスタの 1 つのメンバ上で強制終了されても、そのメンバ上のすべてのアプリケーション・リソースは動作し続けますが、それらのリソースは CAA サブシステムを使って一時的に管理できなくなります。Essential Services Monitor デーモン `esmd` は自動的に `caad` を再起動し、管理機能が元に戻ります。Essential Services Monitor デーモンが `caad` デーモンを再起動できない場合には、`syslog` に高い優先度のイベントをポストします。詳細は `esmd(8)` を参照してください。

`/usr/sbin/caad` コマンドを手動で入力することによって、このデーモンの再起動を試みることができます。

`caad` デーモンを再起動するために起動スクリプト `/sbin/init.d/clu_caa` を使用しないでください。このスクリプトは、クラスタ・メンバをブートするときに、`caad` デーモンを起動するためにだけ使用します。

### 8.10.3 CAA デーモンのメッセージのモニタ

リソースの状態変化に関する情報を見るには、CAA デーモンによって EVM にポストされたイベントを表示します。EVM のメッセージについての詳細は、8.11 節を参照してください。

## 8.11 EVM による CAA のイベントの表示

CAA デーモンは、EVM (イベント・マネージャ) にイベントをポストします。これらのイベントは、CAA サブシステムで発生したエラーの解決に役立つ場合があります。

---

## 注意

---

いくつかの CAA 動作は、syslog を介して /var/cluster/members/{member}/adm/syslog.dated/[date]/daemon.log にログが取られます。daemon.log と EVM の両方の情報を取得するのが問題を判別するのに有益です。EVM はクラスタ全体の情報の単一ソースであるという利点があり、daemon.log 情報は各メンバ固有です。daemon.log のみに存在する情報がいくつかあります。

---

SysMan Station を使用するか、コマンド行から EVM のコマンドを使用することによって、EVM にポストされたイベントにアクセスできます。SysMan Station の使用方法についての詳細は、Tru64 UNIX 『システム管理ガイド』を参照してください。特定のタスクの実行方法についての詳細は、オンライン・ヘルプを参照してください。

CAA デーモンが生成する多くのイベントは、EVM の構成ファイル /usr/share/evm/templates/clu/caa/caa.evt に定義されています。これらのイベントにはすべて sys.unix.clu.caa.\* という形式の名前が付きます。

CAA デーモンが生成する一部のイベントには、sys.unix.syslog.daemon という名前が付きます。ただし、他のデーモンによってポストされるイベントにもこの名前が付きます。したがって、EVM によって表示されるのは CAA のイベントだけではありません。

EVM Event Management System からの情報の取得についての詳細は、EVM(5)、evmget(1)、または evmshow(1) を参照してください。

### 8.11.1 CAA のイベントの表示

EVM にポストされた CAA のイベントを表示するには、次のコマンドを入力します。

```
# evmget -f "[name *.caa.*]" | evmshow
CAA cluster_lockd was registered
CAA cluster_lockd is transitioning from state ONLINE to state OFFLINE
CAA resource sbtest action script /var/cluster/caa/script/foo.scr (start): success
CAA Test2002_Scale6 was registered
CAA Test2002_Scale6 was unregistered
```

EVM の詳細なイベント情報を取得するには、オプション -d を次のように指定します。

```
# evmget -f "[name *.caa.*]" | evmshow -d | more
===== EVM Log event =====
EVM event name: sys.unix.clu.caa.app.registered

This event is posted by the Cluster Application Availability
subsystem (CAA) when a new application has been registered.

=====

Formatted Message:
    CAA a was registered

Event Data Items:
    Event Name      : sys.unix.clu.caa.app.registered
    Cluster Event   : True
    Priority        : 300
    PID            : 1109815
    PPID           : 1103504
    Event Id       : 4578
    Member Id      : 2
    Timestamp       : 18-Apr-2001 16:56:17
    Cluster IP address: 16.69.225.123
    Host Name       : provolone.zk4.dec.com
    Cluster Name    : deli
    User Name       : root
    Format          : CAA $application was registered
    Reference       : cat:evmexp_caa.cat

Variable Items:
    application (STRING) = "a"

=====
```

テンプレート・スクリプト /var/cluster/caa/template/template.scr は、CAA がアプリケーションを起動、停止、チェックするときにイベントがポストされるスクリプトを生成するように変更されました。caa\_profile または SysMan で新たに作成された処理スクリプトはどれも、EVM ヘイイベントをポストするようになります。これらのイベントのみを表示するには、次のコマンドを入力します。

```
# evmget -f "[name sys.unix.clu.caa.action_script]" | evmshow -t "@timestamp @@"
```

CAA のイベントは SysMan Station を使っても表示できます。そのためには、SysMan Station Monitor Window の [Status Light or Label Box for Applications] をクリックします。

他のデーモンと同様に caad デーモンによってログに記録されたその他のイベントを表示するには、次のコマンドを入力します。

```
# evmget -f "[name sys.unix.syslog.daemon]" | \
evmshow -t "@timestamp @@"
```

### 8.11.2 CAA のイベントのモニタ

CAA のイベントにタイム・スタンプを付けてコンソール上で監視するには、次のコマンドを入力します。

```
# evmwatch -f "[name *.caa.*]" | evmshow "@timestamp @@"
```

CAA のイベントが EVM にポストされると、このコマンドの実行元の端末上にそれらのイベントが表示されます。たとえば、次のようなメッセージが表示されます。

```
CAA cluster_lockd was registered
CAA cluster_lockd is transitioning from state ONLINE to state OFFLINE
CAA Test2002_Scale6 was registered
CAA Test2002_Scale6 was unregistered
CAA xclock is transitioning from state ONLINE to state OFFLINE
CAA xclock had an error, and is no longer running
CAA cluster_lockd is transitioning from state ONLINE to state OFFLINE
CAA cluster_lockd started on member polishham
```

syslog 機能を用いて、CAA デーモンとその他のデーモンがログをとったその他のイベントを表示するには、次のコマンドを入力します。

```
# evmwatch -f "[name sys.unix.syslog.daemon]" | evmshow | grep CAA
```

## 8.12 イベントに関するトラブルシューティング

この節で示すエラー・メッセージは、次のコマンドを入力して、CAA デーモンからのイベントを示すときに表示される可能性があります。

```
# evmget -f "[name sys.unix.syslog.daemon]" | evmshow | grep CAA
```

### 時間切れの処理スクリプト

```
CAAD[564686]: RTD #0: Action Script \
/var/cluster/caa/script/[script_name].scr(start) timed out! (timeout=60)
```

最初に、`/var/cluster/caa/script/[script_name].scr start` を実行して、処理スクリプトでアプリケーションが正しく起動されるかどうかを確認してください。処理スクリプトが正しく動作してエラーなしに正常に戻っても、`SCRIPT_TIMEOUT` の値より時間がかかる場合は、`SCRIPT_TIMEOUT` の値を増やしてください。スクリプトで実行されたアプリケーションの終了に時間がかかる場合は、スクリプト内のアプリケーションを起動する行にアンパサンド (&) を追加して、スクリプト内のタスクをバックグラウンドで実行してください。ただし、この処理はコマンドが常に 0 の状態を返し、CAA はささいな理由 (たとえば、コマンド・パスのスペルの誤り) によるコマンドの失敗を検出できません。

処理スクリプトの停止エントリ・ポイントでのリターン値 **0** 以外

```
CAAD[524894]: 'foo' on member 'provolone' has experienced an unrecoverable failure.
```

このメッセージは、停止エントリ・ポイントが 0 以外の値を返すときに出力されます。リソースは UNKNOWN 状態になります。アプリケーションを停止するためには、0 を返すように停止処理スクリプトを修正して `caa_stop` または `caa_stop -f` を実行する必要があります。いずれにしても、停止処理スクリプトが 0 を返すように修正してから、アプリケーション・リソースを再起動してください。

ネットワーク障害

```
CAAD[524764]: 'tu0' has gone offline on member 'skiing'
```

ネットワーク・リソース `tu0` のこのようなメッセージは、ネットワークが停止したことを示します。ネットワーク・カードが正しく接続されているかどうかを確認してください。必要であれば、カードを交換してください。

CAA デーモンの起動を妨げるロック

```
CAAD[526369]: CAAD exiting; Another caad may be running, could not obtain \
lock file /var/cluster/caa/locks/.lock-provolone.dec.com
```

これと同様のメッセージは、2 つ目の `caad` を起動しようとしたときに表示されます。8.10.1 項で説明したように `caad` が実行されているかどうかを確認してください。このデーモンが実行されていない場合は、メッセージでリストされているロック・ファイルを削除して、8.10.2 項で説明したように `caad` を再起動してください。

## 8.13 コマンド行メッセージのトラブルシューティング

以下のようなメッセージは、ユーザが登録しようとしたリソースのプロファイルが CAA で検出できないことを示しています。

```
Cannot access the resource
profile file_name
```

たとえば、`clock` のプロファイルがない場合は、`clock` を登録しようとしても失敗して、次のようになります。

```
# caa_register clock
Cannot access the resource profile '/var/cluster/caa/profile/clock.cap'.
```

リソース・プロファイルの場所が正しくないか、プロファイルがありません。プロファイルが、メッセージ中に示された場所に存在するかどうかを確認してください。



## ファイル・システムとデバイスの管理

この章では、TruCluster Server システムでのストレージ・デバイスの管理に特有な情報を記載しています。ここでは、次の項目について説明します。

- CDSL の使用 (9.1 節)
- デバイスの管理 (9.2 節)
- クラスタ・ファイル・システムの管理 (9.3 節)
- デバイス要求ディスクパッチャの管理 (9.4 節)
- クラスタでの AdvFS の管理 (9.5 節)
- 新しいファイル・システムの作成 (9.6 節)
- CDFS ファイル・システムの管理 (9.7 節)
- ファイルのバックアップとリストア (9.8 節)
- スワップ領域の管理 (9.9 節)
- ブート・パラメータによる問題の修正 (9.10 節)
- クラスタでの `verify` の使用 (9.11 節)

その他のデバイス管理に関する情報は、表 9-1 に示す Tru64 UNIX バージョン 5.1B のマニュアルに記載されています。

表 9-1: ストレージ・デバイス管理の参考マニュアル

内容	Tru64 UNIX マニュアル
デバイスの管理	『ハードウェア管理ガイド』
ファイル・システムの管理	『システム管理ガイド』
アーカイブ・サービスの管理	『システム管理ガイド』
AdvFS の管理	『AdvFS 管理ガイド』

LSM (Logical Storage Manager) とクラスタについては、第 10 章を参照してください。

## 9.1 CDSL の使用

コンテキスト依存シンボリック・リンク (CDSL) には、クラスタ・メンバを識別する変数があります。この変数は、実行時にターゲット内で解決されます。

CDSL は、次のように構造化されます。

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

CDSL パス名を解決する場合は、カーネルにより、文字列 {memb} が文字列 member $n$  に置き換えられます。ここで、 $n$  は現在のメンバ ID です。たとえば、メンバ ID が 2 のクラスタ・メンバでは、パス名 /cluster/members/{memb}/etc/rc.config は、/cluster/members/member2/etc/rc.config に解決されます。

CDSL を利用すれば、単一ファイル名で複数のファイルのうちの 1 つを示すことができます。クラスタでこの CDSL を利用すれば、メンバ固有のファイルをクラスタ単位で単一ファイル名によって指定できるようになります。システム・データと構成ファイルで CDSL が利用される傾向にあります。このようなファイルは、ルート (/)、/usr、および /var のディレクトリにあります。

### 9.1.1 CDSL の作成

mkcdsl コマンドには、CDSL の作成や設定を行うための簡単なツールが用意されています。たとえば、/usr/accounts/usage-history ファイルの新規 CDSL を作成するには、次のコマンドを入力してください。

```
# mkcdsl /usr/accounts/usage-history
```

結果のリストを表示させると、次のように出力されます。

```
# ls -l /usr/accounts/usage-history
```

```
... /usr/accounts/usage-history -> cluster/members/{memb}/accounts/usage-history
```

CDSL usage-history は、/usr/accounts に作成されます。どのメンバの /usr/cluster/members/{memb} ディレクトリにもファイルは作成されません。

CDSL にファイルを移動するには、次のコマンドを使用してください。

```
# mkcdsl -c targetname
```

コピー (-c) オプションを使用して既存のファイルを置き換える場合は、強制 (-f) オプションも使用しなければなりません。

-c オプションを使用すれば、クラスタ・メンバのメンバ固有の領域にソース・ファイルがコピーされます。この領域で、mkcdsl コマンドが実行され、ソース・ファイルが CDSL で置き換えられます。ソース・ファイルをすべてのクラスタ・メンバのメンバ固有の領域にコピーしてから、ソース・ファイルを CDSL に置き換えるには、次のように、コマンドに -a オプションを使用してください。

```
# mkcdsl -a filename
```

CDSL の削除は、他のシンボリック・リンクを削除する場合と同様に、rm コマンドを使用します。

/var/adm/cdsl\_admin.inv ファイルには、クラスタの CDSL の記録が格納されます。mkcdsl コマンドを使用して CDSL を追加すると、コマンドにより、/var/adm/cdsl\_admin.inv が更新されます。ln -s コマンドを使用して CDSL を作成した場合には、/var/adm/cdsl\_admin.inv は更新されません。

/var/adm/cdsl\_admin.inv を更新するには、次のコマンドを入力します。

```
# mkcdsl -i targetname
```

CDSL を削除する場合、または ln -s を使用して CDSL を作成した場合は、インベントリを更新します。

詳細は、mkcdsl(8) を参照してください。

## 9.1.2 CDSL の保守

次のツールは、CDSL の保守に役立ちます。

- clu\_check\_config(8)
- cdslinvchk(8)
- mkcdsl(8) (-i オプションを指定)

次の例は、clu\_check\_config で不正な CDSL や不明な CDSL が検出された場合の出力 (およびエラーが記述されたログ・ファイルへのポインタ) を示しています。

```
# clu_check_config -s check_cdsl_config
Starting Cluster Configuration Check...
check_cdsl_config : Checking installed CDSLs
check_cdsl_config : CDSLs configuration errors : See /var/adm/cdsl_check_list
clu_check_config : detected one or more configuration errors
```

通常、ファイルを移動する前に、コピー先が CDSL でないことを確認します。誤って該当するクラスタ・メンバの CDSL を上書きした場合は、`mkcdsl -c filename` コマンドを使用してファイルをコピーし、CDSL を再作成してください。

### 9.1.3 カーネル構築と CDSL

クラスタでカーネルを構築する場合は、コマンド `cp` を使って、新しいカーネルを `/sys/HOSTNAME/vmunix` から `/vmunix` へコピーします。カーネルを `/vmunix` へ移動させると、`/vmunix` CDSL を上書きします。そのため、次にクラスタ・メンバをブートすると、`/sys/HOSTNAME/vmunix` 内の古い `vmunix` を使うことになります。

### 9.1.4 CDSL のエクスポートとマウント

CDSL は、1 つのファイル名で異なるクラスタ・メンバのさまざまな内容を表す必要がある場合に使用されます。このようなことから、CDSL はエクスポート用には指定されていません。

クラスタ別名で CDSL をマウントすると、マウント要求を取得するクラスタ・システムによってファイルの内容が異なるため、問題が生じます。ただし、CDSL のエクスポートを防止することはできません。ディレクトリ全体が CDSL になっている場合は、マウント要求を受けるノードから、そのノードのディレクトリに対応するファイル・ハンドルが得られます。CDSL が、エクスポートされたクラスタ単位のディレクトリ内に含まれている場合は、要求を受けるネットワーク・ファイル・システム (NFS) サーバが拡張します。通常のシンボリック・リンクと同様に、クライアントがファイルまたはディレクトリを読み込むことはできません。ただし、該当する領域もクライアントでマウントされている場合は除きます。

## 9.2 デバイスの管理

クラスタでのデバイス管理は、次の例外を除けば、スタンドアロン・システムにおける管理と同様です。

- デバイス・スペシャル・ファイルの管理用の `dsfmgr` コマンドで、クラスタの特殊オプションが解釈される場合
- クラスタ内で共用バスとプライベート・バスが混在するため、デバイス・トポロジがさらに複雑になる場合

- クラスタ内でサーバとして機能するクラスタ・メンバ，およびアクセス・ノードとして機能するメンバを指定できる場合

以降の各項で，これらの違いについて説明します。

## 9.2.1 デバイス・スペシャル・ファイルの管理

クラスタ内で `dsfmgr` (デバイス・スペシャル・ファイル管理ユーティリティ) を使用する場合は，次のことに留意してください。

- `-a` オプションには，`entry_type` として `c` (クラスタ) を使用する必要があります。
- `-o` と `-O` のオプションは，古い形式のデバイス・スペシャル・ファイルを作成するので，クラスタでは有効ではありません。
- `-s` オプションでの出力では，最初のテーブルの `class scope` の欄は，`c` (クラスタ) を使用してデバイスの有効範囲を示します。

詳細は，`dsfmgr(8)` を参照してください。デバイス，デバイスの命名，およびデバイス管理については，Tru64 UNIX 『ハードウェア管理ガイド』を参照してください。

## 9.2.2 デバイス位置の確定

Tru64 UNIX の `hwmgr` コマンドを使用すれば，プライベート・バス上のハードウェア・デバイスをはじめとするクラスタ内のすべてのハードウェア・デバイス，バス-ターゲット-LUN の名前と `/dev/disks/dsk*` の名前との関連を示すことができます。たとえば，次のようになります。

```
# hwmgr -view devices -cluster
HWID: Device Name      Mfg      Model      Hostname      Location
-----
3: kevm                pepicelli
28: /dev/disk/floppy0c  3.5in floppy pepicelli fdi0-unit-0
40: /dev/disk/dsk0c     DEC      RZ28M      (C) DEC pepicelli bus-0-targ-0-lun-0
41: /dev/disk/dsk1c     DEC      RZ28L-AS   (C) DEC pepicelli bus-0-targ-1-lun-0
42: /dev/disk/dsk2c     DEC      RZ28       (C) DEC pepicelli bus-0-targ-2-lun-0
43: /dev/disk/cdrom0c   DEC      RRD46      (C) DEC pepicelli bus-0-targ-6-lun-0
44: /dev/disk/dsk3c     DEC      RZ28M      (C) DEC pepicelli bus-1-targ-1-lun-0
44: /dev/disk/dsk3c     DEC      RZ28M      (C) DEC provolone bus-1-targ-1-lun-0
45: /dev/disk/dsk4c     DEC      RZ28L-AS   (C) DEC pepicelli bus-1-targ-2-lun-0
45: /dev/disk/dsk4c     DEC      RZ28L-AS   (C) DEC polishham bus-1-targ-2-lun-0
45: /dev/disk/dsk4c     DEC      RZ28L-AS   (C) DEC provolone bus-1-targ-2-lun-0
46: /dev/disk/dsk5c     DEC      RZ29B      (C) DEC pepicelli bus-1-targ-3-lun-0
46: /dev/disk/dsk5c     DEC      RZ29B      (C) DEC polishham bus-1-targ-3-lun-0
46: /dev/disk/dsk5c     DEC      RZ29B      (C) DEC provolone bus-1-targ-3-lun-0
47: /dev/disk/dsk6c     DEC      RZ28D      (C) DEC pepicelli bus-1-targ-4-lun-0
47: /dev/disk/dsk6c     DEC      RZ28D      (C) DEC polishham bus-1-targ-4-lun-0
47: /dev/disk/dsk6c     DEC      RZ28D      (C) DEC provolone bus-1-targ-4-lun-0
```

```

48: /dev/disk/dsk7c DEC RZ28L-AS (C) DEC pepicelli bus-1-targ-5-lun-0
48: /dev/disk/dsk7c DEC RZ28L-AS (C) DEC polishham bus-1-targ-5-lun-0
48: /dev/disk/dsk7c DEC RZ28L-AS (C) DEC provolone bus-1-targ-5-lun-0
49: /dev/disk/dsk8c DEC RZ1CF-CF (C) DEC pepicelli bus-1-targ-8-lun-0
49: /dev/disk/dsk8c DEC RZ1CF-CF (C) DEC polishham bus-1-targ-8-lun-0
49: /dev/disk/dsk8c DEC RZ1CF-CF (C) DEC provolone bus-1-targ-8-lun-0
50: /dev/disk/dsk9c DEC RZ1CB-CS (C) DEC pepicelli bus-1-targ-9-lun-0
50: /dev/disk/dsk9c DEC RZ1CB-CS (C) DEC polishham bus-1-targ-9-lun-0
50: /dev/disk/dsk9c DEC RZ1CB-CS (C) DEC provolone bus-1-targ-9-lun-0
51: /dev/disk/dsk10c DEC RZ1CF-CF (C) DEC pepicelli bus-1-targ-10-lun-0
51: /dev/disk/dsk10c DEC RZ1CF-CF (C) DEC polishham bus-1-targ-10-lun-0
51: /dev/disk/dsk10c DEC RZ1CF-CF (C) DEC provolone bus-1-targ-10-lun-0
52: /dev/disk/dsk11c DEC RZ1CF-CF (C) DEC pepicelli bus-1-targ-11-lun-0
52: /dev/disk/dsk11c DEC RZ1CF-CF (C) DEC polishham bus-1-targ-11-lun-0
52: /dev/disk/dsk11c DEC RZ1CF-CF (C) DEC provolone bus-1-targ-11-lun-0
53: /dev/disk/dsk12c DEC RZ1CF-CF (C) DEC pepicelli bus-1-targ-12-lun-0
53: /dev/disk/dsk12c DEC RZ1CF-CF (C) DEC polishham bus-1-targ-12-lun-0
53: /dev/disk/dsk12c DEC RZ1CF-CF (C) DEC provolone bus-1-targ-12-lun-0
54: /dev/disk/dsk13c DEC RZ1CF-CF (C) DEC pepicelli bus-1-targ-13-lun-0
54: /dev/disk/dsk13c DEC RZ1CF-CF (C) DEC polishham bus-1-targ-13-lun-0
54: /dev/disk/dsk13c DEC RZ1CF-CF (C) DEC provolone bus-1-targ-13-lun-0
59: kevm polishham
88: /dev/disk/floppy1c 3.5in floppy polishham fdi0-unit-0
94: /dev/disk/dsk14c DEC RZ26L (C) DEC polishham bus-0-targ-0-lun-0
95: /dev/disk/cdrom1c DEC RRD46 (C) DEC polishham bus-0-targ-4-lun-0
96: /dev/disk/dsk15c DEC RZ1DF-CB (C) DEC polishham bus-0-targ-8-lun-0
99: /dev/kevm provolone
127: /dev/disk/floppy2c 3.5in floppy provolone fdi0-unit-0
134: /dev/disk/dsk16c DEC RZ1DF-CB (C) DEC provolone bus-0-targ-0-lun-0
135: /dev/disk/dsk17c DEC RZ1DF-CB (C) DEC provolone bus-0-targ-1-lun-0
136: /dev/disk/cdrom2c DEC RRD47 (C) DEC provolone bus-0-targ-4-lun-0

```

drdmgr devicename コマンドでは、デバイスに対するサービスを提供するメンバが報告されます。複数のサーバに対応するディスクは共用 SCSI バス上にあります。数少ない例外を除けば、1 台のサーバのみに対応しているディスクは、そのサーバのプライベート・ディスクです。例外についての詳しい説明は、9.4.1 項を参照してください。

クラスタ・メンバのハードウェア構成を確認するには、次のコマンドを入力してください。

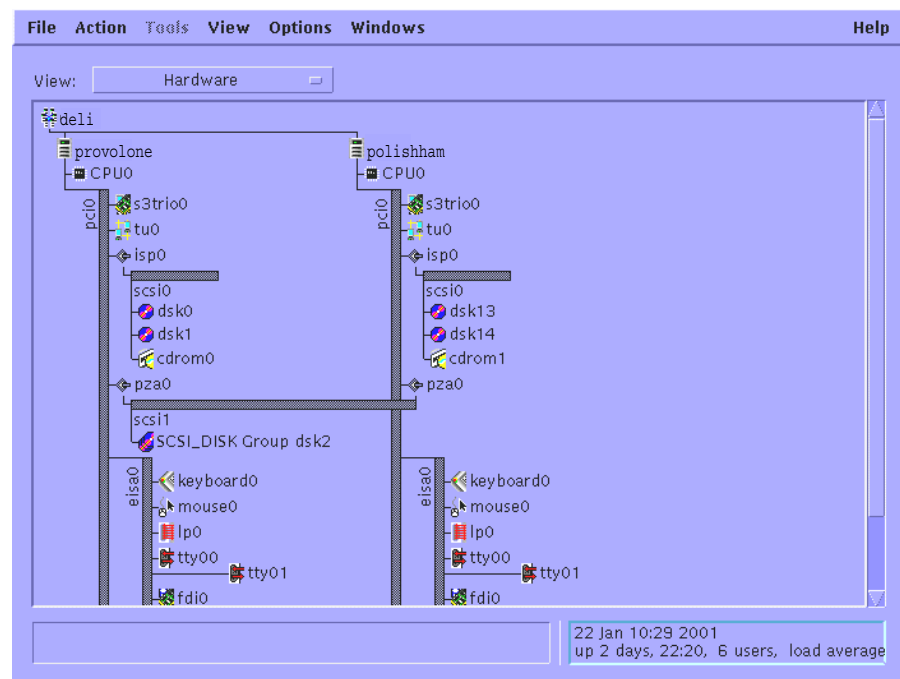
```
# hwmgr -view hierarchy -member membername
```

メンバが共用バス上にある場合は、このコマンドで、共用バス上のデバイスが報告されます。このコマンドでは、他のメンバのプライベート・デバイスは報告されません。

動作中のメンバ、バス、共用ストレージ・デバイスとプライベート・ストレージ・デバイス、およびそれらの接続をはじめとするクラスタ・ハードウェア構成をグラフィック表示するには、sms コマンドを使用して SysMan Station のグラフィック・インタフェースを起動し、次に [View] メニューから [Hardware] を選択してください。

図 9-1 は、SysMan Station で表示した 2 メンバ・クラスタを示しています。

図 9-1: SysMan Station でのハードウェア構成の表示



ZK-1700U-AI

### 9.2.3 クラスタへのディスクの追加

SCSI ハードウェア・デバイスの物理的な設置については、TruCluster Server 『クラスタ・ハードウェア構成ガイド』を参照してください。新しいディスクを設置した後は、以下の手順に従ってください。

1. すべてのメンバが新しいディスクを認識するように、各メンバ上で次のコマンドを実行します。

```
# hwmgr -scan comp -cat scsi_bus
```

## 注意

ディスクにアクセスする必要があるすべてのクラスタ・メンバ上で `hwmgr -scan comp -cat scsi_bus` コマンドを実行しなければなりません。

すべてのメンバに新しいディスクの情報が登録されるまでしばらく待ちます。

2. 追加するディスクがモデル RZ26, RZ28, RZ29, または RZ1CB-CA の場合, 各クラスタ・メンバ上で次のコマンドを実行します

```
# /usr/sbin/clu_disk_install
```

クラスタに多数のストレージ・デバイスがある場合は, このコマンドが完了するのに数分かかります。

3. 新しいディスクの名前を確認するには, 次のコマンドを入力します。

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

SysMan Station コマンドを実行し, [Views] メニューから [Hardware] を選択して, 新しいディスク名を確認することもできます。

ディスク上でのファイル・システムの作成については, 9.6 節を参照してください。

### 9.2.4 他社製ストレージの管理

クラスタ・メンバがクォーラムを失うと, その入出力はすべて中断され, 残りのメンバは, クラスタから削除されたノードに対して入出力バリアを形成します。この入出力バリアが働くと, 非クラスタ・メンバは共用ストレージ・デバイスに対して入出力できなくなります。

入出力バリアを形成する方法は, クラスタ・メンバが共用しているストレージ・デバイスのタイプによって異なります。場合によっては, Target\_Reset というタスク管理の機能を使って, 今までメンバであったノードとの間で行うすべての入出力を停止します。このタスク管理の機能は, 次のいずれかの状況で使用されます。

- 共用している SCSI デバイスが SCSI Persistent Reserve コマンド・セットをサポートしておらず, Fibre Channel インターコネクトを使用する場合



- 共用している SCSI デバイスが 4 つの条件，つまり，(1) SCSI Persistent Reserve コマンド・セットをサポートしていない，(2) SCSI Parallel インターコネクトを使用する，(3) マルチポートのデバイスである，および (4) SCSI Target\_Reset シグナルを伝えない，にすべて当てはまる場合

どちらの状況でも，Target\_Reset を送ってから，デバイスと前メンバとの間で保留されていたすべての入出力がクリアされるまでに，遅延が生じます。この遅延の長さは，デバイスとクラスタの構成によって異なります。遅延が発生している間に，前メンバから入出力が発生することもあります。Target\_Reset の後で送られたこの入出力は，他のノードから影響を受けることなく，通常どおり行われます。

この遅延待ち時間は，`drd_target_reset_wait` カーネル属性で構成可能です。デバイス要求ディスパッチャは，この時間が経過するまで，共用デバイスに対する新しい入出力をすべて中断します。Target\_Reset を受信した後，前メンバから入出力要求を受け取ったデバイスでも，この間にクリアできます。この時間が経過すると，入出力バリアが働き始めます。

`drd_target_reset_wait` の省略時の値は 30 秒です。通常は，この値で十分です。それでも，クラスタ内に他社製デバイスがあって不安のある場合は，そのデバイスの製造元に仕様を問い合わせ，Target\_Reset を受信してから入出力をクリアするまでにどの程度時間がかかるかを確認してください。

`drd_target_reset_wait` はブート時と実行時に設定することもできます。

クォーラムの喪失とシステムの分断についての詳細は，TruCluster Server 『クラスタ概要』の接続マネージャに関する章を参照してください。

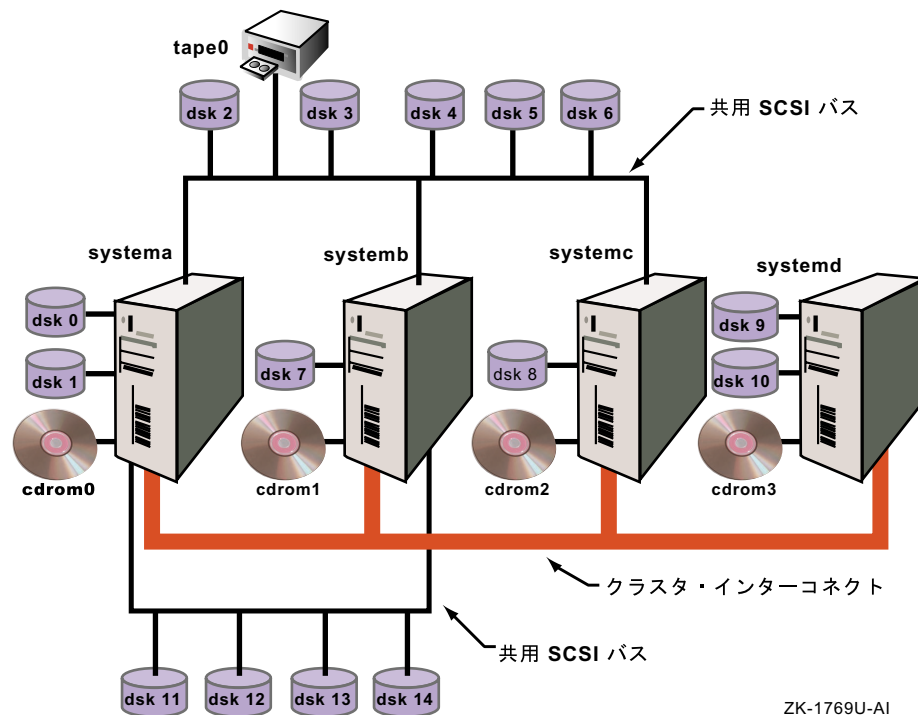
## 9.2.5 テープ装置

テープ装置がメンバのプライベート・バス上，共用バス上，または別メンバのプライベート・バス上にあっても，クラスタ上のそのテープ装置にはどのメンバからでもアクセスできます。

複数のメンバが装置へ直接アクセスする共用バス上に，テープ装置を配置します。共用バス上のテープ装置の性能も考慮します。テープ装置のある共用バス上のシステムへ接続されたストレージのバックアップは，クラスタ・インターコネクトを介すより高速です。たとえば，図 9-2 では，`dsk9` と `dsk10` をテープ・ドライブへバックアップするにはクラスタ・インターコネクトを介して行います。セミ・プライベート・ディスク `dsk11`，`dsk12`，

dsk13 , dsk14 を含むその他のディスクをバックアップする場合 , データ転送速度が速くなります。

図 9-2: セミ・プライベート・ストレージを持つクラスタ



ZK-1769U-AI

テープ装置が共用バス上に配置されている場合 , 装置へアクセスするアプリケーションは , 共用 SCSI バス上の特定のイベントに対して適切に対応するように記述しなければなりません (たとえば , バス , デバイスのリセット)。バスとデバイスのリセット (たとえば , クラスタ・メンバシップ遷移による) は , 共用 SCSI バス上のすべてのテープ装置を巻き戻します。

テープ・サーバ・アプリケーションの `read()` または `write()` では , `errno` が返されます。I/O 呼出しから戻されたエラー情報を使用するようにテープ・サーバ・アプリケーションを明示的に設定し , テープを再位置付けします。`read()` または `write()` の操作が失敗したときは , `ioctl()` を `MTIOCGGET` コマンド・オプションと一緒に使用して , アプリケーションでテープを再位置付けするために必要なエラー情報を持つ構造体を取得します。構造体の説明は , `/usr/include/sys/mtio.h` を参照してください。

通常使うユーティリティ `tar` , `cpio` , `dump` , `vdump` はこのように設計されていません。そのため、クラスタ内の共用バスに存在するテープ装置上で使うときには、予告なく停止する可能性があります。

## 9.2.6 クラスタでのディスクットのフォーマット

9.3.7 項で説明するように、TruCluster Server には UFS (UNIX ファイル・システム) ファイル・システムの読み取り/書き込み機能があるので、TruCluster Server を用いてディスクットをフォーマットできます。

バージョン 5.1A より前の TruCluster Server では、UFS ファイル・システムの読み取り/書き込みはできません。旧バージョンの TruCluster Server は UFS ファイル・システムの読み取り/書き込みをサポートしておらず、また、AdvFS メタデータがディスクットの容量を超えるので、クラスタ内では一般的な方法でディスクットをフォーマットできません。

バージョン 5.1A より前のバージョンの TruCluster Server でクラスタ内のディスクットをフォーマットしなければならない場合は、`mttools` または `dxmtools` のツール・セットを使用してください。詳細は、`mttools(1)` および `dxmtools(1)` を参照してください。

## 9.2.7 CD-ROM と DVD-ROM

CD-ROM 装置と DVD-ROM 装置は常時サービスされる装置です。この種の装置はローカル・バスへ接続する必要があります。つまり、共用バスへ接続できません。

クラスタ上で CD-ROM ファイル・システム (CDFS) を管理する方法については、9.7 節を参照してください。

## 9.3 クラスタ・ファイル・システムの管理

クラスタ・ファイル・システム (CFS) を利用すれば、クラスタ内のどこに格納されたファイルでも透過的にアクセスできます。ユーザやアプリケーションは、1 つのシステム・イメージでファイルにアクセスできます。アクセス方法は、アクセス要求元のクラスタ・メンバがどこにあるかや、ファイルを格納しているディスクがクラスタ内のどこに接続されているかにかかわらず、同じです。CFS では、サーバ/クライアント・モデルに従って、クラスタ・メンバが各ファイル・システムに対するサービスを提供します。クラスタ・メンバは、クラスタ内のどのデバイス上のファイル・システムに対する

サービスも提供できます。ファイル・システムを提供しているメンバが利用できなくなった場合は、CFS サーバにより利用可能なクラスタ・メンバに自動的にフェイルオーバーされます。

クラスタ・ファイル・システム管理用の基本ツールは `cfsmgr` コマンドです。このコマンドの使用例についてはこの節で説明します。`cfsmgr` コマンドについての詳しい説明は、`cfsmgr(8)` を参照してください。

TruCluster Server バージョン 5.1B では、`mount` コマンドの `-o` オプションが用意され、起動時に特定のクラスタ・メンバでファイル・システムがサービスされます。このオプションについては 9.3.4 項で説明します。

TruCluster Server バージョン 5.1B では、負荷モニタ・デーモン `/usr/sbin/cfsd` が提供されており、ファイル・システム関連のメンバやクラスタのアクティビティを監視したり、そのレポートを作成したり、応答したりします。`cfsd` デーモンについては、9.3.3 項で説明します。

CFS についての統計情報を収集するには、コマンド `cfsstat` またはコマンド `cfsmgr -statistics` を使います。`cfsstat` を使って直接入出力についての情報を取得する例が 9.3.6.2 項に示されています。このコマンドについての詳細は、`cfsstat(8)` を参照してください。

共用バス上のデバイスのファイル・システムでは、入出力性能は、バスの負荷とファイル・システムを提供しているメンバの負荷によって異なります。負荷分散を単純化するために、CFS では、サーバを別のメンバに容易に再配置できます。メンバがプライベート・デバイスのファイル・システムに対するサービスを提供している場合、そのメンバは、ファイル・システムに高速にアクセスできます。

### 9.3.1 ファイル・システムがフェイルオーバーできない場合

ほとんどの場合、CFS を利用すれば、クラスタ内のファイル・システムを問題なくフェイルオーバーできます。ファイル・システムに対するサービスを提供しているクラスタ・メンバが利用できなくなった場合、CFS では、利用可能なメンバにサーバがフェイルオーバーされます。ただし、次のような場合には、ファイル・システムへのパスが存在しないので、ファイル・システムがフェイルオーバーできません。

- ファイル・システムのストレージが、メンバに直接接続されたプライベート・バス上にあり、そのメンバが利用できなくなった場合

- ストレージが共用バス上にあり、共用バス上のすべてのメンバが利用できなくなった場合

どちらの場合も、`cfsmgr` コマンドは、ファイル・システム (またはドメイン) について次の状態を返します。

```
Server Status : Not Served
```

ファイル・システムにアクセスしようとする、次のメッセージが返されます。

```
filename I/O error
```

ストレージに接続されたクラスタ・メンバが利用可能になると、ファイル・システムに対するサービスが再び提供されて、ファイル・システムにアクセスできるようになります。メンバを利用可能にする以外には、処置の必要はありません。

### 9.3.2 ダイレクト・アクセス・キャッシュド・リード

TruCluster Server には ダイレクト・アクセス・キャッシュド・リード が実装されており、AdvFS ファイル・システムの性能が向上しています。CFS は ダイレクト・アクセス・キャッシュド・リード を使用することで、同時に複数のクラスタ・メンバに代わってストレージから読み取ることができます。

読み取りを発行したクラスタ・メンバがファイル・システムのあるストレージに直接接続されていると、ダイレクト・アクセス・キャッシュド・リード はそのストレージに直接アクセスし、CFS サーバとの間にあるクラスタ・インターコネクトを経由しません。

CFS クライアントは、自分がファイル・システムのあるストレージに直接接続されていない場合 (たとえば、ストレージがクラスタ・メンバに対してプライベートである場合) でも、読み取り要求をそのデバイスに直接発行します。ただし、この場合、デバイス要求ディスパッチャの階層はクラスタ・インターコネクトを通して読み取り要求をデバイスに送ります。

ダイレクト・アクセス・キャッシュド・リード と CFS がサービスを行う既存のファイル・システム・モデルとの間には整合性があり、CFS サーバは、読み取り操作に関するメタデータとログの更新を今までどおり行います。

ダイレクト・アクセス・キャッシュド・リード は AdvFS ファイルにのみ実装されています。また、ダイレクト・アクセス・キャッシュド・リード は、サ

イズが 64K 以上のファイルに対してのみ実行されます。小さなファイル処理する場合は、通常の入出力方法の方が効率的です。

ダイレクト・アクセス・キャッシュド・リードは特に指定しなくても有効になっており、ユーザが設定したり調節したりすることはできません。ただし、アプリケーションが 9.3.6.2 項で述べるように直接入出力を使用する場合は、そちらの方が優先され、そのアプリケーションに対してダイレクト・アクセス・キャッシュド・リードは実行されません。

cfsstat directio コマンドを使用して直接入出力の統計情報を表示させることができます。ダイレクト・アクセス・キャッシュド・リードの統計情報は direct i/o reads フィールドに表示されます。これらのフィールドについての詳細は、9.3.6.2.3 項を参照してください。

```
# cfsstat directio
Concurrent Directio Stats:
  941 direct i/o reads
    0 direct i/o writes
    0 aio raw reads
    0 aio raw writes
    0 unaligned block reads
  29 fragment reads
  73 zero-fill (hole) reads
    0 file-extending writes
    0 unaligned block writes
    0 hole writes
    0 fragment writes
    0 truncates
```

### 9.3.3 CFS の負荷分散

クラスタのブート時に、TruCluster Server ソフトウェアにより、各ファイル・システムは、それぞれに対するサービスを提供するメンバに直接接続されます。メンバのローカル・バスに接続されたデバイスのファイル・システムは、そのメンバがサービスを提供します。共用 SCSI バス上のデバイスのファイル・システムは、その SCSI バスに直接接続されたメンバのいずれかがサービスを提供します。

AdvFS の場合は、CFS サーバに割り当てられた最初のファイルセットによって、そのドメイン内の他のすべてのファイルセットに CFS サーバと同じクラスタ・メンバが設定されていると判断されます。

クラスタのブート時には、通常、共用 SCSI バスに接続されている最初に起動したメンバが、最初に共用バス上のデバイスを検出します。その後、こ

のメンバが、共用バス上のすべてのデバイスに対するすべてのファイル・システムの CFS サーバになります。このため、ほとんどのファイル・システムは 1 つのメンバで取り扱われるため、そのメンバは他のメンバよりも負荷が重くなります。したがって、そのメンバのリソース (CPU、メモリ、I/O、など) の使用量は大きくなります。このような場合には、ファイル・システムを他のクラスタ・メンバに再配置して、負荷の分散と性能の向上を図ることをお奨めします。

TruCluster Server バージョン 5.1B には、負荷モニタ・デーモン `/usr/sbin/cfsd` が用意されており、ファイル・システム関連のメンバとクラスタのアクティビティを監視したり、そのレポートを作成したり、応答したりします。`cfsd` は、省略時の構成では無効なので、明示的に有効にする必要があります。有効にされた後、`cfsd` は次の機能を実行します。

- 目的やストレージの接続性に基づいてファイル・システムを配置することにより、ファイル・システムの管理を支援します。`cfsd` を構成して、ストレージの接続性が変化したり、CFS のメモリ使用量が大きくなった場合、メンバをクラスタに参加させたり切り離したりするときに、自動的にファイル・システムを再配置できます。
- ファイル・システムの使用性やシステム負荷のさまざまな統計情報を収集します。このデータによって、クラスタのファイル・システムの使用方法を把握できます。
- 収集した統計情報を分析し、システム性能の改善やクラスタ単位でのファイル・システムの負荷分散を図るために、ファイル・システムの再配置を推奨します。
- クラスタ・ノード上で CFS のメモリ使用量を監視し、メンバが CFS メモリの使用限度に近づいたときに警告します。

`cfsd` デーモンのインスタンスはクラスタの各メンバ上で実行します。すなわち、`cfsd` がクラスタで動作するときには、各メンバ上で実行する必要があります。適切に動作する各メンバ上のデーモンに依存します。`cfsd` をクラスタで動作させたくない場合は、どのメンバにもその実行を許可しないでください。

デーモンの各インスタンスはそのメンバ上で統計情報を収集し、CFS メモリが少ないというようなメンバ固有のイベントを監視します。クラスタのデーモンは、自動的に“マスタ”デーモンとしてサービスされ、収集したすべての統計情報を分析し、推奨案を作成して、自動的な再配置を開始しま

す。このデーモンは、クラスタ単位の `/etc/cfsd.conf` 構成ファイルを紹介して構成されます。

`cfsd` デーモンは、定期的にメンバをポーリングして情報を入手し、ファイル・システムの性能とリソースの使用量を監視します。このポーリングは、指定されたポリシーに対応した `/etc/cfsd.conf` 構成ファイルの `polling_schedule` 属性に従って行われます。

`cfsd` デーモンは、各メンバの各ファイル・システムの使用量、各ファイル・システムのメモリ要求、各メンバのシステム・メモリ要求、メンバの物理的なストレージ接続性などの情報を収集しています。各デーモンは、メンバ固有のバイナリ・ファイル `/var/cluster/cfs/stats.member` に統計情報を格納します。このファイルのデータは、`cfsd` 固有のフォーマットであり、直接のユーザ・アクセスを意図していません。`cfsd` デーモンはこれらのファイルを更新して、保守します。定期的に削除したり、保存したりする必要はありません。

次のデータは、各クラスタ・メンバによって収集されます。

- `svrcfstok` 構造体の限度値 (この構造体については、9.3.6.3 項を参照)
- `svrcfstok` 構造体の使用数
- 総バイト数
- ラインのバイト数

次のデータは、メンバごと、ファイル・システムごとに集められます。

- 読み取り回数
- 書き込み回数
- 検索回数
- `getattr` 操作回数
- `readlink` 操作回数
- アクセス回数
- その他の操作回数
- 読み取りバイト数
- 書き込みバイト数
- `svrcfstok` 構造体の使用数 (9.3.6.3 項で説明)



また、`cfstd` デーモンは EVM イベントを処理し、クラスタ・メンバシップ、マウントされたファイル・システムおよびデバイス接続性のような、一般的なクラスタとクラスタのファイル・システムの状態についての情報を監視します。

---

#### 注意

---

`cfstd` は AdvFS ドメインに必要なエンティティとみなします。AdvFS ファイル・システムの再配置は、対象のファイル・システムばかりでなく同ドメインのすべてのファイル・システムに影響を及ぼします。ドメイン全体が再配置されます。

NFS クライアントおよび MFS (メモリ・ファイル・システム) タイプのファイル・システムは再配置できません。さらに、メンバ・ブート・パーティション、サーバだけのファイル・システム (読み書きアクセス用にマウントされた UFS ファイルなど)、HSM (hierarchical storage manager) で管理されるファイル・システムも、再配置できません。

共用バス上のダイレクト・アクセス入出力デバイスはバス上のすべてのクラスタ・メンバによって使用されます。共用バス上、またはクラスタ・メンバに直接接続された単一サーバ・デバイスは、単一メンバによって使用されます。2 つ以上のファイル・システムが同一の単一サーバ・デバイスを使用している場合、`cfstd` はファイル・システムが別のメンバによってサービスされるときに起こる性能上の理由から再配置しません。

#### 9.3.3.1 `cfstd` の起動と停止

`/sbin/init.d/cfstd` ファイルは各クラスタ・メンバ上で `cfstd` デーモンのインスタンスを起動します。ただし、デーモンの起動では `cfstd` をアクティブにしません。`cfstd` デーモンの動作は、スタンザ・フォーマット・ファイル `/etc/cfstd.conf` の `active` フィールドを介して制御されます。`active` フィールドに 1 が設定されると、`cfstd` をアクティブにします。

省略時の設定では `cfstd` デーモンは無効 (0 を設定) です。使用したい場合は、明示的に有効にする必要があります。

cfstd デーモンは、起動時にクラスタ単位の `/etc/cfstd.conf` ファイルを読み取ります。次のように、cfstd に SIGHUP シグナルを送って構成ファイルの再読み取りを強制的に行うことができます。

```
# kill -HUP `cat /var/run/cfstd.pid`
```

`/etc/cfstd.conf` を変更する場合は、cfstd に SIGHUP シグナルを送って変更したファイルを再度読み取ります。クラスタ内の 1 つの cfstd デーモン・プロセスに SIGHUP を送信した場合、クラスタ内のすべての cfstd デーモンがこのファイルを再度読み取ります。複数の SIGHUP シグナルを送る必要はありません。

### 9.3.3.2 EVM イベント

cfstd デーモンは、最新の分析結果に有益な情報が含まれていることを警告するため、EVM の `sys.unix.clu.cfstd.anlys.relocsuggested` イベントをポストします。evmwatch コマンドと evmshow コマンドを使って、これらのイベントを監視します。

```
# evmget | evmshow -t "@name [@priority]" | grep cfstd
sys.unix.clu.cfstd.anlys.relocsuggested [200]
```

次のコマンドは、cfstd EVM イベントの追加情報を提供します。

```
# evmwatch -i -f "[name sys.unix.clu.cfstd.*]" | evmshow -d | more
```

### 9.3.3.3 `/etc/cfstd.conf` 構成ファイルの変更

`/etc/cfstd.conf` 構成ファイルは (cfstd.conf(4) で説明)、cfstd の一般的なパラメータを指定し、cfstd がクラスタのファイル・システムを管理して分析する場合に従うファイル・システム配置ポリシーのセットを定義します。クラスタ内のすべてのファイル・システムには、それらに関連付けられた配置ポリシーがあります。このポリシーでは、各ファイル・システムをメンバに割り当てる方法を指定し、必要であれば cfstd がファイル・システムを自動的に再配置するかどうかを決定します。

このファイルを変更する場合は、次の点に注意してください。

- ファイル・システムに明示的にポリシーを指定しない場合、省略時のポリシーを引き継ぎます。
- cfstd デーモンは、たとえブート・パーティションが再配置を行う `active_placement` ポリシーに属していたとしても、デーモンはクラスタ・メンバのブート・パーティションを決して再配置しません。

cfstd デーモンはブート・パーティション用の active\_placement を無視します。

- active\_placement キーワードは、自動的な再配置を発生させるイベントを定義します。hosting\_members オプションは、ファイル・システムを再配置するメンバを定義します。
- hosting\_members オプションは、優先リストでなく制限するリストです。placement 属性は、hosting\_members 属性で指定されたメンバが利用できないときに、ファイル・システムを配置する方法を制御します。
- cfstd デーモンは、1 つだけの hosting\_members エントリで定義されたすべてのクラスタ・メンバを平等に扱います。リストの位置による優先順位はありません。

メンバの優先順位を指定するには、複数の hosting\_members 行を使用します。cfstd デーモンは、一番最初の hosting\_members 行にリストされたメンバに優先権を与え、次の hosting\_members 行のメンバに次の優先権を与えます。以下、このように優先順位を付与します。

- 作成可能なポリシーの数には制限がありません。
- file systems 行には、ファイル・システムを任意の数だけリストできます。複数のポリシーに同一のファイル・システムが出現すると、最後に使用されたファイル・システムの優先順位が採用されます。

/etc/cfstd.conf ファイルの例を例 9-1 に示します。詳細については、cfstd.conf(4) を参照してください。

#### 例 9-1: /etc/cfstd.conf ファイル

```
# Use this file to configure the CFS load monitoring daemon (cfstd)
# for the cluster. cfstd will read this file at startup and on receipt
# of a SIGHUP. After modifying this file, you can apply your changes
# cluster-wide by issuing the following command from any cluster
# member: "kill -HUP `cat /var/run/cfstd.pid`". This will force the
# daemon on each cluster member to reconfigure itself. You only need
# to send one SIGHUP.
#
# Any line whose first non-whitespace character is a '#' is ignored
# by cfstd.
#
# If cfstd encounters syntax errors while processing this file, it will
# log the error and any associated diagnostic information to syslog.
#
# See cfstd(8) for more information.

# This block is used to configure certain daemon-wide features.
#
# To enable cfstd, set the "active" attribute to "1".
```

## 例 9-1: /etc/cfsd.conf ファイル(続き)

---

```
# To disable cfsd, set the "active" attribute to "0".
#
# Before enabling the daemon, you should review and understand the
# configuration in order to make sure that it is compatible with how
# you want cfsd to manage the cluster's file systems.
#
# cfsd will analyze load every 12 hours, using the past 24 hours worth
# of statistical data.
#
cfsd:
  active = 1
  reloc_on_memory_warning = 1
  reloc_stagger = 0
  analyze_samplesize = 24:00:00
  analyze_interval = 12:00:00

# This block is used to define the default policy for file systems that
# are not explicitly included in another policy. Furthermore, other
# policies that do not have a particular attribute explicitly defined
# inherit the corresponding value from this default policy.
#
# Collect stats every 2 hours all day monday-friday.
# cfsd will perform auto relocations to maintain server preferences,
# connectivity, and acceptable memory usage, and will provide relocation
# hints to the kernel for preferred placement on failover.
# No node is preferred over another.
#
defaultpolicy:
  polling_schedule = 1-5, 0-23, 02:00:00
  placement = favored
  hosting_members = *
  active_placement = connectivity, preference, memory, failover

# This policy is used for file systems that you do NOT want cfsd to
# ever relocate. It is recommended that cfsd not be allowed to relocate
# the /, /usr, or /var file systems.
#
# It is also recommended that file systems whose placements are
# managed by other software, such as CAA, also be assigned to
# this policy.
#
policy:
  name = PRECIOUS
  filesystems = cluster_root#, cluster_usr#, cluster_var#
  active_placement = 0

# This policy is used for file systems that cfsd should, for the most
# part, ignore. File systems in this policy will not have statistics
# collected for them and will not be relocated.
#
# Initially, this policy contains all NFS and MFS file systems that
# are not explicitly listed in other policies. File systems of these
# types tend to be temporary, so collecting stats for them is usually
# not beneficial. Also, CFS currently does not support the relocation
# of NFS and MFS file systems.
#
policy:
  name = IGNORE
```

### 例 9-1: /etc/cfsd.conf ファイル (続き)

---

```
filesystems = %nfs, %mfs
polling_schedule = 0
active_placement = 0

# Policy for boot file systems.
#
# No stats collection for boot file systems. Boot partitions are never
# relocated.
#
policy:
  name = BOOTFS
  filesystems = root1_domain#, root2_domain#, root3_domain#
  polling_schedule = 0

# You can define as many policies as necessary, using this policy block
# as a template. Any attributes that you leave commented out will be
# inherited from the default policy defined above.
#
policy:
  name = POLICY01
  #filesystems =
  #polling_schedule = 0-6, 0-23, 00:15:00
  #placement = favored
  #hosting_members = *
  #active_placement = preference, connectivity, memory, failover
```

---

#### 9.3.3.4 cfsd 分析の理解と実装に関する推奨事項

cfsd デーモンは、メンバ固有のファイル `/var/cluster/cfs/stats.member` に統計情報を収集します。これらのデータ・ファイルは、cfsd 固有のフォーマットであり、直接のユーザ・アクセスを意図していません。cfsd デーモンはこれらのファイルを更新し、維持します。定期的に削除したり、保存したりする必要はありません。

cfsd は、収集された統計情報を分析した後に、分析結果を `/var/cluster/cfs/analysis.log` ファイルに格納します。`/var/cluster/cfs/analysis.log` ファイルは、最新の `/var/cluster/cfs/analysis.log.dated` ファイルへのシンボリック・リンクです。`/var/cluster/cfs/analysis.log.dated` ファイルは 24 時間後に、新しいバージョンが作られてシンボリック・リンクが更新されます。古いバージョンの `/var/cluster/cfs/analysis.log.dated` ファイルは、削除されます。cfsd デーモンは、最新の分析結果に有益な情報が含まれていることを警告するため、EVM イベントをポストします。

/var/cluster/cfs/analysis.log ファイルは、通常のテキスト文で、次のような形式です。

```
Cluster Filesystem (CFS) Analysis Report
(generated by cfsd[525485])

Recommended
relocations:

none

Filesystem usage summary:
cluster      reads      writes      req'd svr mem
            24 KB/s      0 KB/s      4190 KB

node         reads      writes      req'd svr mem
rye          4 KB/s      0 KB/s      14 KB
swiss        19 KB/s      0 KB/s      4176 KB

filesystem
node         reads      writes      req'd svr mem
test_one#    2 KB/s      0 KB/s      622 KB
rye          0 KB/s      0 KB/s
@swiss       2 KB/s      0 KB/s

test_two#    4 KB/s      0 KB/s      2424 KB
rye          1 KB/s      0 KB/s
@swiss       3 KB/s      0 KB/s
:
:
Filesystem placement evaluation results:

filesystem
node         conclusion  observations
test_one#
  rye        considered (hi conn, hi pref, lo use)
  @swiss     recommended (hi conn, hi pref, hi use)
test_two#
  rye        considered (hi conn, hi pref, lo use)
  @swiss     recommended (hi conn, hi pref, hi use)
:
:
```

各ファイル・システムの現行の CFS サーバは、“at” シンボル (@) で示されます。前にも説明していますが、cfsd は、AdvFS ドメインに必要なエンティティとみなしており、分析は AdvFS ドメイン・レベルで報告されます。AdvFS タイプのファイル・システムの再配置は、同ドメインのすべてのファイル・システムに影響を及ぼします。

この分析結果を使用して、ファイル・システムの CFS サーバを別のクラスター・メンバにするのかどうかを判断できます。現行の CFS サーバが、cfsd の分析に基づき、このファイル・システムにとって推奨されるサーバでない

場合、`cfsmgr` コマンドを使用して推奨されるサーバへファイル・システムを再配置することができます。

たとえば、`swiss` が `test_two` ドメインの現行の CF サーバであり、`rye` が推奨される CFS サーバであると仮定します。この分析結果を受け入れて実装する場合は、次の `cfsmgr` コマンドを実行して、CFS サーバを `rye` に変更します。

```
# cfsmgr -a server=rye -d test_two
# cfsmgr -d test_two
Domain or filesystem name = test_two
Server Name = rye
Server Status : OK
```

### 9.3.3.5 自動再配置

`cfstd` デーモは、独自の統計情報の分析だけに基づいて自動再配置を行いません。むしろ、レポートを生成し、ユーザ環境に応じて採用の判断が可能な推奨案を提供します。

ただし、条件を設定すると、`cfstd` はファイル・システム・ポリシーの `active_placement` オプションで指定されたキーワードに基づいて、自動的にファイル・システムを再配置できます。`active_placement` キーワードでは、自動再配置を行うイベントを定義します。`hosting_members` オプションは、ファイル・システムを再配置するメンバとその優先順位を定義します。

`active_placement` オプションの動作と指定できる値については、`cfstd.conf(4)` に説明されています。次にその概要を示します。

- メモリ

CFS メモリ使用量は、9.3.6.3 項で説明されている

`svrcfstok_max_percent` カーネル属性によって制限されます。クラスタ・メンバがこの制限に達すると、そのメンバによってサービスされているファイル・システム上でのファイル処理は、「file table overflow」エラーで失敗します。メンバが CFS メモリの使用限度に近づくと、カーネルが警告として EVM イベントをポストします。このようなイベントがポストされると、`cfstd` はサービスしているファイル・システムのいくつかを再配置して、そのメンバ上のメモリを開放しようとします。

- 優先順位

メンバがクラスタに参加したり切り離されたりするとき、`cfstd` は希望するメンバにファイル・システムを再配置することができます。クラスタ・メンバのサブセットでの基本的なファイル・システムを望む場合もあります。

- フェイルオーバー

メンバがクラスタに参加したり切り離されたりすると、`cfstd` は希望するメンバにファイル・システムを再配置することができます。特定のファイル・システムのサービスを、主にクラスタ・メンバのサブセットで行いたい場合もあります。

- 接続性

メンバがサービスしているファイル・システムによって要求されたデバイスへの直接的な物理接続がない場合、サーバの性能は低下します。`cfstd` デーモンは、現行のサーバがファイル・システム下のデバイスへの接続を失ったというイベントで自動的にファイル・システムを再配置します。

### 9.3.3.6 CAA リソースとの関連性

`cfstd` デーモンは、CAA リソースの情報を持ちません。CAA では、`hosting_members`、`placement_policy`、`required_resources` オプションを使って、特定の CAA リソースを実行できるメンバを優先的に指定したり、制限したりすることができます。

CAA リソースがアプリケーション固有のファイル・システムやリソース固有のファイル・システムを使う場合、関連する CAA 処理スクリプトを使って、ファイル・システムを配置したり再配置します。リソースが再配置された場合、処理スクリプトでも `cfsmgr` を使って、同様にファイル・システムを移動すべきです。処理スクリプトを通して `cfsmgr` コマンドを使うと、CAA リソースとファイル・システムを直接、容易に同期させられるようになります。

### 9.3.3.7 `cfstd` 以外での CFS の負荷分散

`cfstd` デーモンは、クラスタ上で CFS の負荷を分析し、分散させる推奨方法です。`cfstd` デーモンは、メンバとクラスタ・アクティビティを監視したり、そのレポートを作成したり、ファイル・システムに関連するメンバに応答したりします。ただし、既にファイル・システムの負荷を分散させ



る手段がある場合、または単に自分で負荷分散の分析を実行する場合には、当然そうすることもできます。

`cfsmgr` コマンドを使用して、CFS サーバの再配置に最適な候補を確定してください。`cfsmgr` コマンドでは、メンバ単位にファイル・システムの使用状況に関する統計情報が表示されます。たとえば、性能を改善するために `/accounts` のサーバを再配置した方がよいかどうかを判断したいとします。最初に、次のようにして `/accounts` の現在の CFS サーバを確認します。

```
# cfsmgr /accounts
```

```
Domain or filesystem name = /accounts
Server Name = systemb
Server Status : OK
```

次に、現在のサーバと候補サーバの CFS 統計情報を得るために、次のコマンドを入力します。

```
# cfsmgr -h systemb -a statistics /accounts
```

```
Counters for the filesystem /accounts:
  read_ops = 4149
  write_ops = 7572
  lookup_ops = 82563
  getattr_ops = 408165
  readlink_ops = 18221
  access_ops = 62178
  other_ops = 123112
```

```
Server Status : OK
```

```
# cfsmgr -h systema -a statistics /accounts
```

```
Counters for the filesystem /accounts:
  read_ops = 26836
  write_ops = 3773
  lookup_ops = 701764
  getattr_ops = 561806
  readlink_ops = 28712
  access_ops = 81173
  other_ops = 146263
```

```
Server Status : OK
```

```
# cfsmgr -h systemc -a statistics /accounts
```

```
Counters for the filesystem /accounts:
  read_ops = 18746
  write_ops = 13553
  lookup_ops = 475015
  getattr_ops = 280905
```

```
readlink_ops = 24306
access_ops = 84283
other_ops = 103671
```

Server Status : OK

```
# cfsmgr -h systemd -a statistics /accounts
```

Counters for the filesystem /accounts:

```
read_ops = 98468
write_ops = 63773
lookup_ops = 994437
getattr_ops = 785618
readlink_ops = 44324
access_ops = 101821
other_ops = 212331
```

Server Status : OK

この例では、/accounts の読み取りと書き込みの動作の大半は systemd メンバからで、現在このファイル・システムに対するサービスを提供している systemb メンバからではありません。systemd が /accounts のストレージに物理的に接続されているとすると、systemd は /accounts の CFS サーバとして最適な選択肢となります。

systemd と /accounts のストレージが物理的に接続されているかどうかを次のようにして判断してください。

1. /accounts がマウントされている場所を調べてください。/etc/fstab を調べるか、または mount コマンドを使用できます。マウントされているファイル・システムが多数ある場合は、次のように grep を使用することをお勧めします。

```
# mount | grep accounts
accounts_dmn#accounts on /accounts type advfs (rw)
```

2. AdvFS ドメイン accounts\_dmn がマウントされているデバイスを確認するために、次のようにして /etc/fdmns/accounts\_dmn ディレクトリを調べます。

```
# ls /etc/fdmns/accounts_dmn
dsk6c
```

3. dsk6 のサーバを確認するために、次のように drdmgr コマンドを入力します。

```
# drdmgr -a server dsk6
Device Name: dsk6
Device Type: Direct Access IO Disk
```

```
Device Status: OK
Number of Servers: 4
  Server Name: membera
  Server State: Server
  Server Name: memberb
  Server State: Server
  Server Name: memberc
  Server State: Server
  Server Name: memberd
  Server State: Server
```

dsks6 に複数のサーバが存在するので、そのデバイスは共用バス上にあります。systemd がサーバの 1 つになっているので、物理的に接続されています。

4. 次のように /accounts の CFS サーバを systemd に再配置します。

```
# cfsmgr -a server=systemd /accounts
```

CFS 統計情報の示す負荷がそれほど不均衡になっていない場合でも、共用バスに接続されている利用可能なメンバ間で CFS サーバを分散するようお勧めします。そうすれば、クラスタの性能を全体的に向上できます。

#### 9.3.3.8 cfsmgr による CFS サーバの負荷分散

特定のクラスタ・メンバを自動的にファイル・システムまたはドメインの CFS サーバとして機能させるには、cfsmgr コマンドを呼び出すスクリプトを /sbin/init.d に設定すれば、ファイル・システムまたはドメインのサーバを目的のクラスタ・メンバに再配置できます。この方法は CFS の負荷を分散させますが、平衡はとりません。

たとえば、クラスタ・メンバ alpha で accounting ドメインに対するサービスを提供したい場合は、起動スクリプトに次のように cfsmgr コマンドを設定してください。

```
# cfsmgr -a server=alpha -d accounting
```

そしてスクリプトを編集し、再配置が正常に終了したかどうかを確認して、再配置が失敗した場合は、処理を再試行させる必要があります。cfsmgr コマンドは、失敗した場合にゼロ以外の値を返しますが、この値では、不正な終了値でスクリプトを再試行するのに不十分です。フェイルオーバーまたは再配置がすでに進行しているために、再配置に失敗したとも考えられます。

そのため、スクリプトでは、再配置に失敗した場合に、次の 2 つのメッセージを検索するようにします。

```
Server Status : Failover/Relocation in Progress
```

```
Server Status : Cluster is busy, try later
```

このどちらのメッセージが見つかってても再配置を再試行するようなスクリプトにします。その他のエラーの場合には、スクリプトが適切なメッセージを出力して終了するようにします。

### 9.3.4 mount -o コマンドによるファイル・システムの分散

共用 SCSI バス上のデバイスのファイル・システムは、その SCSI バスに直接接続されているメンバの 1 つによってサービスされます。クラスタのブート時、通常は、共用 SCSI バスに接続されている最初にアクティブにされるメンバが共用バス上のデバイスを最初に使うメンバとなります。このメンバが、その共用バス上のすべてのデバイスのすべてのファイル・システムの CFS サーバとなります。CFS では、ファイル・システムの負荷をよりよく分散するために、9.3.3 項で説明されているように、ファイル・システムの再配置を行うことができます。

もう 1 つの方法は、`mount -o server=name` コマンドを使用して、起動時にファイル・システムをどのクラスタ・メンバがサービスするかを指定することです。`-o server=name` オプションは、再配置ができない NFS、MFS、および読み取り/書き込み用 UFS ファイル・システムのようなファイル・システムには、特に有効です。

```
# mount -t nfs -o server=rye smooch:/usr /tmp/mytmp
# cfsmgr -e
Domain or filesystem name = smooch:/usr
Mounted On = /cluster/members/member1/tmp/mytmp
Server Name = ernest
Server Status : OK
```

`mount -o server=name` コマンドによってマウントが成功すると、指定したクラスタ・メンバがそのファイル・システムの CFS サーバになります。ただし、指定されたメンバがそのクラスタのメンバでないかファイル・システムをサービスできない場合、そのマウントの試みは失敗します。

`mount -o server=name` コマンドは、ファイル・システムを最初にどこにマウントするかを決定します。ファイル・システムを後で再配置したりフェイルオーバーしたりするクラスタ・メンバの制限や定義は行いません。

`-o server_only` オプションと `-o server=name` を組み合わせると、ファイル・システムは指定されたクラスタ・メンバのみにマウントされ、パーティショニングされたファイル・システムとして扱われます。つまり、その

ファイル・システムは、それをマウントしたメンバだけからは、読み取り専用アクセスも、読み取り書き込みアクセスも、どちらも可能です。他のクラスタ・メンバは、ファイル・システムからの読み取りも、ファイル・システムへの書き込みもできません。リモート・アクセスは許可されず、フェイルオーバーは発生しません。-o `server_only` オプションは、AdvFS、MFS、および UFS ファイル・システムにだけ適用できます。

---

#### 注意

---

-o `server=name` オプションは、通常のサーバ選択プロセスを介さないため、ファイル・システムのデバイスへの接続性が低いメンバになる可能性があります。さらに、指定したメンバが使用可能でない場合、ファイル・システムは他のクラスタ・メンバでもマウントできません。

-o `server=name` と -o `server_only` オプションの組み合わせでは、CFS ファイル・システムの多くの高可用性機能が使えません。ファイル・システムは、指定されたクラスタ・メンバにだけマウントでき、そのメンバだけがアクセスでき、また他のメンバにフェイルオーバーすることはできません。したがって、この組み合わせを使用する際には、注意が必要です。

---

-o `server=name` オプションは、1 つのクラスタ内でだけ、かつ AdvFS、UFS、MFS、NFS、CDFS および DVDFS ファイル・システムだけで有効です。MFS ファイル・システムの場合、-o `server=name` オプションは、限定された方法でサポートされています。このファイル・システムは、指定されたサーバがローカル・ノードの場合だけマウントできます。

-o `server=name` オプションで `/etc/fstab` ファイルがあるメンバを指定すると、クラスタ・メンバ固有の `fstab` エントリを作成できます。

その他の使い方については、`mount(8)` を参照してください。

### 9.3.5 クローン作成前のデーモンのフリーズ

マルチボリューム・ドメイン構成で一貫性のあるハードウェアのスナップショットを可能にするため、ファイル・システムのメタデータは、それぞれのボリュームのクローンが作成されるとき、すべてのボリュームにわたって整合性がとれていなければなりません。メタデータの整合性を保つために、

Tru64 UNIX バージョン 5.1B では `freezefs` コマンドが用意されました。このコマンドについては、`freezefs(8)` で説明されています。`freezefs` コマンドは AdvFS ドメインをメタデータの整合性がとれたフリーズ状態にし、指定されたフリーズ期限が切れるか、`thawfs` コマンドで明示的に解凍されるまで、その状態を継続することを保証します。複数ボリュームまたは論理単位 (LUN) に渡るすべてのメタデータは、ディスクにフラッシュされて、フリーズ期間中は変更されません。

`freezefs` によって、AdvFS ファイル・システムをマウントしたディレクトリをいくつか指定することが要求されますが、AdvFS ドメインのすべてのファイルセットに影響があります。`freezefs` コマンドは、AdvFS ドメインを必要不可欠なエンティティとみなしています。ドメインのファイル・システムをフリーズすることは、ドメイン全体をフリーズすることになります。

クラスタ構成でファイル・システムをフリーズする場合、その時実行中のすべてのファイル・システム操作は完了するまで実行を継続できます。メタデータの更新を必要としないいくつかのファイル・システム操作は、たとえば、ターゲット・ファイル・システムがフリーズされていても、通常通り動作します。たとえば、`read` および `stat` が該当します。

単一システムかクラスタかによって `freezefs` の機能の仕方に幾分違いがありますが、両方とも、フリーズされたドメイン上でのメタデータの変更は許されません。クラスタでのコマンドの動作で、最も注目される単一システムとの違いは、次のとおりです。

- クラスタ・メンバをシャットダウンすると、クラスタのすべてのフリーズされたファイル・システムが解凍される。
- クラスタ・メンバに障害が発生した場合、クラスタのすべてのフリーズされたファイル・システムが解凍される。

#### 9.3.5.1 ドメインがフリーズされているかどうかの判断

省略時の設定では、`freezefs` はファイル・システムを 60 秒間フリーズします。ただし、`-t` オプションを使用して、さらに短い、またはより長いタイムアウト値を秒単位で指定したり、`thawfs` によって解凍されるまでドメインがフリーズ状態のままであることを指定したりすることができます。

`freezefs` コマンドの `-q` オプションを使って、ファイル・システムにフリーズ状態かどうかを問い合わせることができます。

```
# freezeefs -q /mnt
/mnt is frozen
```

また、freezeefs コマンドは、ファイル・システムがフリーズされるか解凍されるときに EVM イベントをポストします。evmwatch および evmshow コマンドを使用して、クラスタのドメインがフリーズされているか解凍されているかを判断します。次に例を示します。

```
# /usr/sbin/freezeefs -t -1 /freezetest
freezeefs: Successful

# evmget -f "[name sys.unix.fs.vfs.freeze]" | evmshow -t "@timestamp @@"
14-Aug-2002 14:16:51 VFS: filesystem test2_domain#freeze mounted on
/freezetest was frozen

# /usr/sbin/thawfs /freezetest
thawfs: Successful

# evmget -f "[name sys.unix.fs.vfs.thaw]" | evmshow -t "@timestamp @@"
14-Aug-2002 14:17:32 VFS: filesystem test2_domain#freeze mounted on
/freezetest was thawed
```

## 9.3.6 CFS 性能の最適化

CFS 性能を次に示す事項で調整できます。

- 先読みおよび後書きスレッド数の変更 (9.3.6.1 項)
- 直接入出力の利用 (9.3.6.2 項)
- CFS メモリ使用量の調整 (9.3.6.3 項)
- メモリ・マップ・ファイルの使用 (9.3.6.4 項)
- フル・ファイル・システムの回避 (9.3.6.5 項)
- その他の方法 (9.3.6.6 項)

### 9.3.6.1 先読みおよび後書きスレッド数の変更

CFS は、ファイルの順次アクセスを検出すると、先読みスレッドを使用して、データの次の入出力ブロック・サイズ分を読み取ります。また、CFS は後書きスレッドを使用して、ディスクの書き込みが予想される次のデータ・ブロックをバッファに入れます。cfs\_async\_biod\_threads カーネル属性を使用して、非同期先読みと後書きを実行する入出力スレッド数を設定してください。先読みおよび後書きスレッドは、CFS クライアントから行われる読み取りと書き込みのみに適用されます。

`cfs_async_biod_threads` の省略時のサイズは 32 です。一度に 32 を超える数の大規模なファイルを順次にアクセスするような環境では、`cfs_async_biod_threads` を大きくすると、特にそれらのファイルを使用するアプリケーションで待ち時間が短縮される可能性がある場合は、CFS の性能を改善できます。

先読みスレッドおよび後書きスレッドの数は、0 ~ 128 の範囲で調整できます。使用中以外は、スレッドはシステム・リソースをほとんど消費しません。

### 9.3.6.2 直接入出力の利用

アプリケーションで `open` システム・コールにフラグ `O_DIRECTIO` を指定して AdvFS ファイルをオープンした場合、データ入出力はストレージを対象とします。システム・ソフトウェアは、ファイル・システム・レベルでファイルのデータ・キャッシングを行いません。クラスタでは、クラスタのメンバからのファイルへの同時直接入出力が可能です。つまり、どのメンバの要求でもファイルへの入出力は CFS サーバへのクラスタ・インターコネクトを介して行われません。データベース・アプリケーションでは、`raw` 非同期入出力 (クラスタでもサポート) と合わせて頻繁に直接入出力を行って、入出力性能を向上させます。

直接入出力用にオープンされたファイル上で最高の性能を引き出すには、次のような条件があります。

- ファイルの既存位置から読み取る
- ファイルの既存位置へ書き込む
- 読み取られる、または書き込まれるデータのサイズがディスク・セクタ・サイズ (512 バイト) の倍数

次の条件により、直接入出力性能が低くなります。

- メタデータがファイルに変更される操作。ファイル・システムの CFS サーバ以外のメンバで直接入出力を行うアプリケーションが実行されるとき、この操作は、ファイル・システムの CFS サーバへクラスタ・インターコネクトを介して行われます。このような操作には次のものがあります。
  - ファイル内の散在する位置への変更
  - ファイルに追加する変更
  - ファイルを切り捨てる変更



- 8K 未満のフラグメントだけで構成されるファイルでの読み取りまたは書き込み、または大きいファイルの末尾のフラグメント部分に対する読み書き
- ファイルの既存位置以外への非ブロック境界の読み取りまたは書き込み。要求がブロック境界で開始または終了しない場合は、複数の入出力が実行されます。
- 直接入出力用にファイルがオープンされたときには、ドメイン上の AdvFS 移行操作 (たとえば、migrate、rmvol、defragment、balance) は、全メンバ上で実行中の入出力が完了するまでブロックされます。逆に、直接入出力は AdvFS 移行操作が完了するまでブロックされます。

直接入出力を使用するアプリケーションは、それ自身のキャッシングを管理する責任があります。1 つのクラスタ・メンバまたは複数のメンバ上でマルチスレッドの直接入出力を実行する場合は、アプリケーションで、どの場合もセクタの読み取りまたは書き込みが確実に単一のスレッドのみで行われるように同期をとる必要があります。

直接入出力プログラミングについては、Tru64 UNIX 『プログラミング・ガイド』の最適化技術の章を参照してください。

#### 9.3.6.2.1 クラスタとスタンドアロン AdvFS 直接入出力の違い

スタンドアロン・システムと異なるクラスタ上の直接入出力の動作を次に示します。

- 直接入出力ブロック用に既にオープン済みのファイルでの移行操作は、実行中の入出力がすべてのメンバ上で完了するまで実行される。続く入出力は、移行操作が完了するまでブロックされる。
- スタンドアロン・システムの AdvFS では、セクタ・レベル (複数のスレッドがファイルの同じセクタに書き込もうとする場合) で保証する。この保証はクラスタでは提供されていない。

#### 9.3.6.2.2 直接入出力モードでオープンしたファイルを持つファイルセットのクローンを作成する

9.3.6.2 項で説明したように、アプリケーションが open システム・コールで O\_DIRECTIO フラグを立ててファイルをオープンした場合、そのファイルの入出力は、CFS サーバとの間にあるクラスタ・インターコネクトを経

由しません。ただし、直接入出力モードでオープンしたファイルを持つファイルセットのクローンを作成すると、入出力がこのモデルに従わないので、性能がかなり低下するおそれがあります (読み取り性能は、このクローン作成から影響を受けません)。

`clonefsset(8)` で説明されている `clonefsset` ユーティリティは、クローン・ファイルセットという AdvFS ファイルセットの読み取り専用コピーを作成します。クローン・ファイルセットは、ファイルセット・データ構造 (メタデータ) の読み取り専用のスナップショットです。つまり、ファイルセットのクローンを作成すると、このユーティリティは元になるファイルセットの構造のみをコピーし、データはコピーしません。元になっているファイルセットのファイルを変更すると、その書き込みのたびに、元になるデータがまだコピーされていないかがチェックされ、コピーされていなければ、データの書き込みに同期してクローンにもコピーされます。このようにして、クローン・ファイルセットの内容は、最初に作成したときのように元データと同じ内容になるよう維持されます。

直接入出力モードでオープンしたファイルがそのファイルセットにある場合にファイルを変更すると、AdvFS は元データをクローンのストレージにコピーします。AdvFS はこのコピー操作をクラスタ・インターコネクトを経由して送りません。しかし、CFS は、直接入出力モードを使用するアプリケーションが自分と同じノードで動作している場合を除き、ファイルセット内のデータの変更に伴う書き込み操作を、クラスタ・インターコネクトを経由して CFS サーバへ送ります。書き込み操作をクラスタ・インターコネクトを経由して送ると、直接入出力モードでファイルをオープンする利点がなくなってしまいます。

直接入出力モードの利点を活かすためには、バックアップが完了した時点ですぐにクローンを削除し、次の書き込みもストレージに直接書き込まれるようにします。そうすれば、書き込みはクラスタ・インターコネクトを経由して送られません。

#### 9.3.6.2.3 直接入出力に関する統計情報の収集

直接入出力を使うアプリケーションの性能向上が期待より低い場合は、コマンド `cfsstat` を使ってノードごとにグローバルな直接入出力の統計情報を調べます。

最初に `cfsstat` を使用して、アプリケーションを実行せずにグローバルな直接入出力の統計情報を調べます。次に、アプリケーションを実行して統計情報を再度確認し、直接入出力の動作を最適化しないパスがあるかどうかを調べます。

次の例は、コマンド `cfsstat` を使って直接入出力の統計情報を取得する方法を示します。

```
# cfsstat directio
Concurrent Directio Stats:
    160 direct i/o reads
    160 direct i/o writes
      0 aio raw reads
      0 aio raw writes
      0 unaligned block reads
      0 fragment reads
      0 zero-fill (hole) reads
    160 file-extending writes
      0 unaligned block writes
      0 hole writes
      0 fragment writes
      0 truncates
```

個々の統計情報には次の意味があります。

- `direct i/o reads`  
通常の直接入出力読み取り要求回数。要求を発行したメンバ上で処理された読み取り要求で、CFS サーバ上の AdvFS 層へ送信されなかった要求です。
- `direct i/o writes`  
通常の直接入出力書き込み要求実行回数。要求を発行したメンバ上で処理された書き込み要求で、CFS サーバ上の AdvFS 層へ送信されなかった要求です。
- `aio raw reads`  
通常の直接入出力非同期読み取り要求回数。要求を発行したメンバ上で処理された読み取り要求で、CFS サーバ上の AdvFS 層へ送信されなかった要求です。
- `aio raw writes`  
通常の直接入出力非同期書き込み要求回数。要求を発行したメンバ上で処理された書き込み要求で、CFS サーバ上の AdvFS 層へ送信されなかった要求です。

- `unaligned block reads`

ディスク・セクタのサイズ (現在は 512 バイト) の倍数でない読み取り回数。セクタ境界から開始しない要求またはセクタ境界で終了しない要求を加えた回数です。非ブロック境界の読み取り操作は、セクタの読み取りと、セクタの適切な位置からのユーザ・データのコピーから構成されます。

入出力要求がファイルの既存位置を対象とするものであり、フラグメントを含まない場合、この操作は CFS サーバへ送られません。

- `fragment reads`

要求がフラグメントを含むファイルの一部に対してなされたため、CFS サーバへ送信する必要のある読み取り要求回数。

140K 未満のファイルには、8K の倍数でないフラグメントを最後に含むことがあります。また、サイズが 8K 未満の小さいファイルは、フラグメントだけのことがあります。

8K 未満のファイルにフラグメントを含まないようにするには、常に直接入出力用にのみファイルをオープンします。そうしなければ、通常にオープンされたファイルをクローズする場合、フラグメントがファイルに作成されます。

- `zero-fill (hole) reads`

直接入出力でオープンされたファイルの散在する部分に発生する読み取り回数。この要求は CFS サーバへ送られません。

- `file-extending writes`

データをファイルに追加するため、CFS サーバへ送信する書き込み要求回数。

- `unaligned block writes`

ディスク・セクタのサイズ (現在は 512 バイト) の倍数でない書き込み回数。セクタ境界から開始しない要求またはセクタ境界で終了しない要求を加えた回数です。非ブロック境界の書き込み操作は、セクタの読み取り、ブロックの適切な位置へのユーザ・データのコピー、およびその後のデータの書き込みから構成されます。

入出力要求がファイルの既存位置を対象にするものであり、フラグメントを含まない場合、この操作は CFS サーバへ送られません。

- `hole writes`

CFS サーバで AdvFS に送る必要のあるファイル内の散在する位置への書き込み要求回数。

- `fragment writes`

要求がフラグメントを含むファイルの一部に対してなされたため、CFS サーバへ送信する必要のある書き込み要求回数。

140K 未満のファイルには、8K の倍数でないフラグメントを最後に含むことがあります。また、サイズが 8K 未満の小さいファイルは、フラグメントだけのことがあります。

8K 未満のファイルにフラグメントを含まないようにするには、常に直接入出力用にのみファイルをオープンします。そうしなければ、通常にオープンされたファイルをクローズする場合、フラグメントがファイルに作成されます。

- `truncates`

直接入出力でオープンされたファイルの切り捨て要求回数。この要求は CFS サーバへ送られます。

### 9.3.6.3 CFS メモリ使用量の調整

1 つのクラスタ・メンバが多数のファイル・システムに対する CFS サーバになっている場合には、そのクライアント・メンバでは、サービスが提供されるファイル・システムから莫大な数の `vnode` がキャッシュに書き込まれる可能性があります。`vnode` が積極的に使用されなくても、クライアントのキャッシュに書き込まれた `vnode` ごとに、CFS サーバでは、CFS 層でのファイルの記録に必要な CFS トークン構造体用に 800 バイトのシステム・メモリを割り当てなければなりません。このほかに、一般に CFS トークン構造体には、対応する AdvFS アクセス構造体と `vnode` が必要なため、ほぼ 2 倍のメモリ容量が使用されます。

省略時には、各クライアントでは、メモリの最大 4 パーセントを `vnode` のキャッシュに使用できます。複数のクライアントのキャッシュが CFS サーバからの `vnode` で限界に達すると、サーバ上のシステム・メモリが不足して、システムがハングするおそれがあります。

`svrcfstok_max_percent` カーネル属性は、そのようなシステム・ハングを防止するように設計されています。この属性を使用して、CFS サーバに割り

当てられるメモリ容量の上限を設定し、クライアント上の vnode キャッシングを管理します。省略時の値は 25 パーセントです。メモリが使用されるのは、サーバの負荷のためにメモリが必要になった場合に限ります。事前にメモリが割り当てられることはありません。

サーバ上の `svrconfstok_max_percent` の制限に達した後、メンバによって管理されるファイルにアクセスするアプリケーションは、`EMFILE` エラーになります。`perror()` を使用して現在の `errno` 設定を確認するアプリケーションは、アプリケーションで使用される標準エラー・ストリーム `stderr`、制御 TTY またはログ・ファイルに対して `too many open files` というメッセージを返します。`EMFILE` エラー・メッセージを調べても、キャッシュに書き込まれたデータは失われません。

アプリケーションで `EMFILE` エラーの取得を開始する場合は、以下の手順に従ってください。

1. CFS クライアントが vnode の範囲外かどうかを次のように判断します。

- a. `max_vnodes` カーネル属性の現在の値を取得します。

```
# sysconfig -q vfs max_vnodes
```

- b. `dbx` を使用して `total_vnodes` と `free_vnodes` の値を取得します。

```
# dbx -k /vmunix /dev/mem
dbx version 5.0
Type 'help' for help.
(dbx) pd total_vnodes
total_vnodes_value
```

`max_vnodes` の値を取得します。

```
(dbx) pd max_vnodes
max_vnodes_value
```

`total_vnodes` が `max_vnodes` と等しく、`free_vnodes` が 0 の場合、そのメンバは vnode の範囲外です。このような場合には、`max_vnodes` カーネル属性の値を増やすことができます。`sysconfig` を使用すれば、動作中のメンバの `max_vnodes` を変更できます。たとえば、vnode の最大数を 20000 に設定するには、次のように入力します。

```
# sysconfig -r vfs max_vnodes=20000
```

2. CFS クライアントが `vnode` の範囲外でない場合は、CFS サーバでトークン構造体 (`svrcfstok_max_percent`) に利用可能なメモリすべてが使用されたかどうかを次のように調べてください。

- a. CFS サーバにログインします。
- b. `dbx` を使用して、`svrtok_active_svrcfstok` の現在の値を取得します。

```
# dbx -k /vmunix /dev/mem
dbx version 5.0
Type 'help' for help.
(dbx)pd svrtok_active_svrcfstok
active_svrcfstok_value
```

- c. `cfs_max_svrcfstok` の値を取得します。

```
(dbx)pd cfs_max_svrcfstok
max_svrcfstok_value
```

`svrtok_active_svrcfstok` が `cfs_max_svrcfstok` 以上になっている場合は、CFS サーバでトークン構造体に利用可能なメモリすべてが使用されています。

このような場合に、ファイル・システムを再び使用できるようにする理想的な解決策は、ファイル・システムの一部を他のクラスタ・メンバーに再配置することです。それが無理な場合は、以下のような解決策が適しています。

- `cfs_max_svrcfstok` の値を増やす。

`sysconfig` コマンドで `cfs_max_svrcfstok` を変更することはできません。ただし、`dbx assign` コマンドを使用すれば、カーネルの動作中に `cfs_max_svrcfstok` の値を変更できます。たとえば、CFS サーバ・トークン構造体の最大数を 80000 に設定するには、次のコマンドを入力します。

```
(dbx)assign cfs_max_svrcfstok=80000
```

`dbx assign` コマンドで設定する値は、システムがリブートされると失われます。

- CFS サーバ上のトークン構造体に利用可能なメモリ容量を増やす。  
このオプションは、メモリ容量の少ないシステムには好ましくありません。

`svrcfstok_max_percent`を増やすには、サーバにログインして、`dxkerneltuner` コマンドを実行します。メイン・ウィンドウで、`cfs` カーネル・サブシステムを選択します。`cfs` ウィンドウで、`svrcfstok_max_percent` に適切な値を入力します。この変更は、クラスタ・メンバがリブートされるまで有効になりません。

一般に、CFS サーバが `svrcfstok_max_percent` の制限に達した場合は、ファイル・システムのサービス提供の負荷がクラスタ・メンバに分散されるように、CFS ファイル・システムの一部を再配置します。起動スクリプトを使用すれば、`cfsmgr` を実行して、メンバの起動時にファイル・システムをクラスタのあちこちに自動的に再配置できます。

`svrcfstok_max_percent` を省略時の値より小さく設定する方法を推奨できるのは、省略時の値の 25 パーセントでも大きすぎてメモリ不足になるような小容量のメモリのシステムに限ります。

#### 9.3.6.4 メモリ・マップ・ファイルの使用

クラスタ単位で読み取り専用アクセス以外にメモリを共用するためにメモリ・マッピングを行うと、性能が低下する可能性があります。複数のメンバが同時にデータを変更しているときには、ファイルへの CFS I/O はうまく動作しません。この状況では、全ノードでいつでも同じデータをアクセスするように、早期キャッシュ・フラッシュが必要となります。

#### 9.3.6.5 フル・ファイル・システムの回避

ファイル・システムの空きスペースが 50 MB 未満の場合、または空きスペースがファイル・システムのサイズの 10 パーセント未満の場合 (どちらが少ない場合であっても)、CFS クライアントからのファイル・システムへの書き込み性能に問題が生じます。これは、フル・ファイル・システムに近い状態でのすべての書き込みが、正常な ENOSPC (“not enough space: 十分な容量がない”) の処理を行えるように即座にサーバへ送信されるためです。

#### 9.3.6.6 その他の方法

次のような方法で CFS の性能を改善できます。

- ・ クラスタ・メンバでシステム・メモリが不足しないように確認します。
- ・ 一般に、クラスタ・メンバの間でファイルを共用して読み書きすると、キャッシュがすべて無効になるため、性能が低下するおそれがありま



す。CFS を使うファイルの入出力は、複数のメンバが同時にデータを変更する場合は正しく動作しません。この状況では、キャッシュが頻繁にフラッシュされるため、すべてのノードに対して、どの時点でもデータが同じように見えるようにすることは困難です。

- 個々のメンバ上の分散型アプリケーションで読み取りと書き込みを行う場合は、書き込みを行うメンバ上にアプリケーションの CFS サーバを配置します。書き込みは、リモート入出力に対して読み取りよりも慎重に処理する必要があります。
- 複数のアプリケーションが単一の AdvFS ドメイン内のさまざまなデータ・セットにアクセスする場合は、そのデータを複数のドメインに分割することを検討します。このようにすれば、単一の CFS サーバより多くのサーバに負荷を分散できます。単一のメンバにすべてをロードせずに、各アプリケーションを、そのアプリケーションのデータ用の CFS サーバと同じ配置にすることもできます。

### 9.3.7 MFS および UFS ファイル・システムのサポート

TruCluster Server では、MFS (メモリ・ファイル・システム) と UFS (UNIX ファイル・システム) に対する読み取り/書き込みをサポートしています。

クラスタ内で UFS ファイル・システムを読み取り/書き込みアクセスでマウントする場合と、クラスタ内で MFS ファイル・システムを読み取り専用または読み取り/書き込みアクセスでマウントする場合は、特に指定しなくても `mount` コマンドで `server_only` 引数が使用されます。これらのファイル・システムは、9.3.8 項で説明するように、パーティショニングされたファイル・システムとして扱われます。つまり、これらのファイル・システムは、それをマウントしたメンバであれば、読み取り専用または読み取り/書き込みアクセスのいずれでもアクセスすることができます。他のクラスタ・メンバは、MFS または UFS ファイル・システムのいずれに対しても書き込みだけでなく読み取りもできません。リモート・アクセスは許可されず、フェイルオーバーはできません。

すべてのクラスタ・メンバに対して読み取り専用で UFS ファイル・システムをマウントするには、明示的に読み取り専用でマウントしなければなりません。

### 9.3.8 ファイル・システムのパーティショニング

CFS はすべてのファイルをすべてのクラスタ・メンバに対してアクセス可能にします。全クラスタ・メンバに接続された装置または単一メンバにプライベートな装置にファイルが格納されているかどうかにかかわらず、各クラスタ・メンバはファイルを同様にアクセスできます。ただし、CFS は AdvFS ファイル・システムのマウントを可能にして、単一クラスタ・メンバのみがアクセスできるようにします。このことをファイル・システムのパーティショニングといいます。

TruCluster Server 製品の初期のバージョンである ASE (Available Server Environment) では、ファイル・システムのパーティショニングのような機能を提供していました。TruCluster Server バージョン 5.1 ではファイルのパーティショニング機能が用意され、ASE から簡単に移植できます。TruCluster Server のファイル・システムのパーティショニングは、ファイル・システムのアクセスを単一メンバに制限するための汎用的な手段としては提供されていません。

パーティショニングされたファイル・システムをマウントするには、ファイル・システムへの排他的アクセスを持つメンバにログオンします。コマンド `mount` をオプション `server_only` を使って実行します。これで、コマンド `mount` を実行するメンバ上でファイル・システムがマウントされ、そのメンバにファイル・システムへの排他的アクセスが与えられます。マウントしたメンバのみがファイル・システムへのアクセス権を持っていても、すべてのメンバがクラスタ単位でファイル・システムをマウントすることができます。

オプション `server_only` は AdvFS、MFS、および UFS ファイル・システムのみに適用できます。

パーティショニングされたファイル・システムには次のような制限があります。

- Tru64 UNIX バージョン 5.1B からは、マウントされるファイル・システムもまたパーティショニングされたファイル・システムであり、同じクラスタのメンバでサービスされる場合には、ファイル・システムはパーティショニングされたファイル・システムにマウントできます。
- CFS を介したフェイルオーバー機能を持たない

パーティショニングされたファイル・システムをサービスするクラスタ・メンバに障害が発生した場合、ファイル・システムはアンマウントされます。

パーティショニングされたファイル・システムを使用するアプリケーションが CAA の制御下にあることでこれを回避できます。パーティショニングされたファイル・システムがマウントされているメンバ上でアプリケーションを実行する必要があるため、メンバに障害が発生すると、ファイル・システムとアプリケーションの両方に障害が発生します。CAA の制御下のアプリケーションは、実行中のクラスタ・メンバにフェイルオーバーします。アプリケーションの CAA 処理スクリプトにパーティショニングされたファイル・システムを新規メンバでマウントするように記述できます。

- NFS エクスポート

パーティショニングされたファイル・システムをエクスポートする最適な方法は、パーティショニングしたそのファイル・システムを提供するノードに対して単一のノード・クラスタ別名を作成し、その別名を `/etc/exports.aliases` ファイルに入れることです。  
`/etc/exports.aliases` ファイルの利用方法についての詳細は、3.15 節を参照してください。

クラスタの提供する NFS マウント・ファイル・システムへアクセスするときに省略時のクラスタ別名を使用すると、NFS 要求がファイル・システムへアクセスできないメンバに送られて失敗することがあります。

パーティショニングされたファイル・システムをエクスポートするもうひとつの方法は、パーティショニングされたファイル・システムを提供するメンバにクラスタ内の一番高いクラスタ別名選択優先順位 (`selp`) を付けることです。これを行うと、メンバはすべての NFS 接続要求に対してサービスします。ただし、メンバはクラスタに指示された全種類のネットワーク・トラフィックも処理する必要があります。これは、ほとんどの環境では容認できません。

接続要求の分散についての詳細は、3.10 節を参照してください。

- 同じドメイン内にパーティショニングされたファイルセットと通常のファイルセットが混在しない

オプション `server_only` は、ドメイン内のすべてのファイル・システムに適用されます。マウントされた最初のファイルセットのタイプが、ドメイン内の全ファイルセットのタイプを決定します。

- ファイルセットがオプション `server_only` なしでマウントされた場合、ドメイン `server_only` 内の別のファイルセットのマウントは失敗します。
- ドメイン内のファイルセットが `server_only` でマウントされた場合、ドメイン内でマウントされたそれ以降のファイルセットは `server_only` になる必要があります。

- 手動の再配置機能を持たない

パーティショニングされたファイル・システムを別の CFS サーバへ移動するには、ファイル・システムをアンマウントし、対象とするメンバ上で再度マウントする必要があります。同時に、ファイル・システムを使うアプリケーションを移動させる必要があります。

- オプション `server_only` を使ったマウント更新機能を持たない

ファイル・システムの通常のマウントの後、ファイル・システム上でオプション `server_only` でコマンド `mount -u` を使用することはできません。たとえば、`file_system` がフラグ `server_only` なしで既にマウント済みの場合、次のコマンドは失敗します。

```
# mount -u -o server_only file_system
```

### 9.3.9 ブロック・デバイスとキャッシュの整合性

単一のブロック・デバイスには、複数の別名を設定できます。この場合には、ファイル・システムのネームスペースにおける複数のブロック・デバイス・スペシャル・ファイルに同じ `dev_t` が含まれます。このような別名は、ネームスペース内の複数のドメインまたはファイル・システムの至るところで見つかる可能性があります。

スタンドアロン・システムでは、キャッシュの整合性は、もともになる共通のブロック・デバイスの `open()` 呼び出しに使用された別名にかかわらず、オープンしたブロック・デバイスすべての間で保証されます。ただし、クラスタでキャッシュの整合性が得られるのは、すべてのブロック・デバイス・ファイルの別名が同じドメインまたはファイル・システム上に存在する場合に限られます。

たとえば、クラスタ・メンバ `mutt` がブロック・デバイスを備えたドメインを提供し、メンバ `jeff` が同じ `dev_t` で別のブロック・デバイス・ファイルを備えたドメインを提供する場合、この2つの別名で同時に入出力が実行されれば、キャッシュの整合性は得られません。

### 9.3.10 CFS の制限

クラスタ・ファイル・システム (CFS) は、ネットワーク・ファイル・システム (NFS) クライアントを読み取り/書き込みアクセスでサポートします。

ファイル・システムをクラスタで NFS マウントした場合、CFS はすべてのクラスタ・メンバで読み取り/書き込みアクセスを可能にします。実際にマウントしたメンバが、他のクラスタ・メンバにファイル・システムを提供します。

NFS ファイル・システムをマウントしたメンバが、シャットダウンするか、または障害が発生すると、ファイル・システムは自動的にアンマウントされ、CFS はマウント・ポイントのクリーンアップを開始します。クリーンアップ処理の間、これらのマウント・ポイントにアクセスするメンバはクリーンアップの進展の度合いに応じてさまざまな状態になります。

- メンバがそのファイル・システム上でまだファイルをオープンしている場合は、書き込みデータは実際の NFS マウント・ファイル・システムの代わりに、ローカル・キャッシュに送られます。
- そのファイル・システム上ですべてのファイルがクローズされている場合は、ファイル・システムが再びマウントされるまで、`EIO` エラーで失敗します。アプリケーションには、「Stale NFS handle」メッセージが送信されます。これは、スタンドアロン・システムでも、クラスタと同様に、通常の動作です。

CFS クリーンアップが完了するまでの間、メンバは NFS ファイル・システムのローカル・マウント・ポイント(または、マウント・ポイントの下にローカルに作られたディレクトリ)で、新しいファイルを作成することができます。

NFS ファイル・システムは自動的に別のメンバにフェイルオーバーしません。手動でファイル・システムをマウントして — 同じマウント・ポイントまたは別のマウント・ポイント — 他のクラスタのメンバから再びアクセスできるようにする必要があります。もう1つの方法は、クラスタ・メンバをブートして `/etc/fstab` ファイルにリストされたファイル・システムをマウントすることです。そのリスト上のファイル・システムは、現在マウントされていない

く、クラスタ内でサービスされていないファイル・システムです (AutoFS または automount を使用している場合、マウントは自動的に行われます)。

## 9.4 デバイス要求ディスパッチャの管理

デバイス要求ディスパッチャ・サブシステムは、クラスタ内の物理的なストレージの位置に関係なく、すべてのクラスタ・メンバが物理的なディスク・ストレージとテープ・ストレージを透過的に利用できるようにします。アプリケーションがファイルへのアクセスを要求すると、CFS から AdvFS に要求が渡され、そこからデバイス要求ディスパッチャに渡されます。ファイル・システム階層においては、デバイス要求ディスパッチャは、デバイス・ドライバの真上になります。

デバイス要求ディスパッチャの管理用の基本ツールは `drdmgr` コマンドです。このコマンドの使用例については、この節で説明しています。詳細は、`drdmgr(8)` を参照してください。

### 9.4.1 ダイレクト・アクセス入出力と単一サーバ・デバイス

デバイス要求ディスパッチャはクライアント/サーバ・モデルに従います。つまり、メンバがディスク、テープ、CD-ROM ドライブなどのデバイスに対するサービスを提供します。

クラスタ内のデバイスは、ダイレクト・アクセス入出力デバイスまたは単一サーバ・デバイスのどちらかです。ダイレクト・アクセス入出力デバイスには、複数のクラスタ・メンバから同時にアクセスできます。単一サーバ・デバイスには、単一メンバのみからアクセスできます。

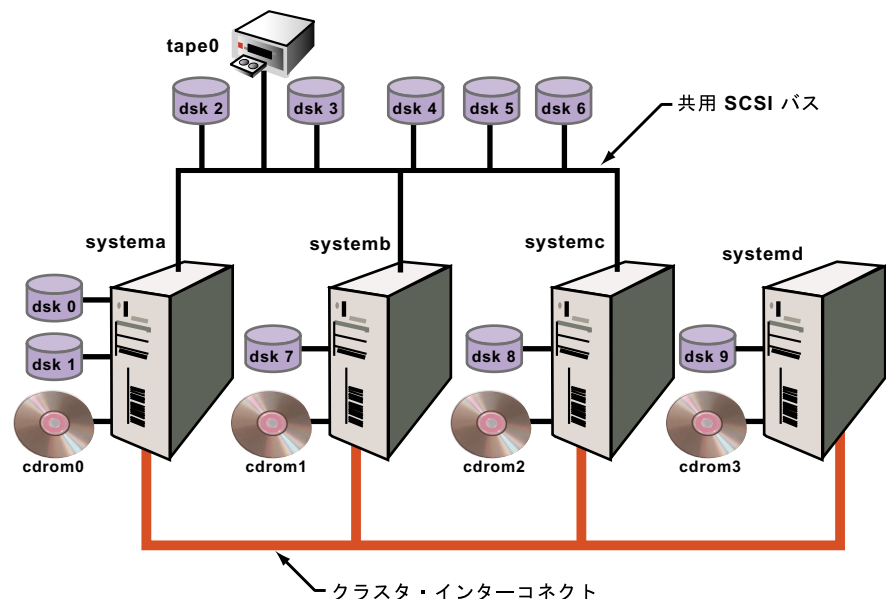
共用バス上のダイレクト・アクセス入出力デバイスは、そのバスのすべてのクラスタ・メンバによってサービスが提供されます。共用バス上か、クラスタ・メンバに直接接続されているかにかかわらず、単一サーバ・デバイスは単一のメンバによってサービスが提供されます。他のすべてのメンバは、サービスを提供しているメンバを介してサービス対象のデバイスにアクセスします。ダイレクト・アクセス入出力デバイスは、デバイス要求ディスパッチャ・サブシステムの一部であり、CFS で処理される直接入出力 (`open` システム・コールに `O_DIRECTIO` フラグを指定したファイルのオープン) とは関係がありません。直接入出力と CFS についての詳しい説明は、9.3.6.2 項を参照してください。

通常、共用バス上のディスクはダイレクト・アクセス入出力デバイスですが、場合によっては、共用バス上の一部のディスクが単一サーバ・デバイスになることがあります。このような例外が発生するのは、設定済みのクラスタに RZ26, RZ28, RZ29, RZ1CB-CA のディスクのうちいずれか 1 つを追加する場合です。最初は、こうしたデバイスは単一サーバ・デバイスです。詳細は、9.4.1.1 項を参照してください。テープ・デバイスは、常に単一サーバ・デバイスです。

共用バス上の単一サーバ・ディスクはサポートされますが、メンバのブート・ディスクまたはスワップ・ファイルとして、あるいはコア・ダンプの取得に使用するにはきわめて低速です。したがって、このような場合にはダイレクト・アクセス入出力ディスクを使用することをお勧めします。

図 9-3 は、共用バス上に 5 台のディスクと 1 台のテープ・ドライブを備えた 4 ノード・クラスタを示しています。systemd は共用バスに接続されていないことに注意してください。このノードからクラスタ・ストレージへのアクセスは、クラスタ・インターコネクトを介して行われます。

図 9-3: 4 ノード・クラスタ



ZK-1546U-AI

共用バス上のディスクは、バスのすべてのクラスタ・メンバによってサービスが提供されます。これを確認するには、次のようにして dsk3 のデバイス要求ディスパッチャ・サーバを検索してください。

```
# drdmgr -a server dsk3
      Device Name: dsk3
      Device Type: Direct Access IO Disk
      Device Status: OK
      Number of Servers: 3
        Server Name: systema
        Server State: Server
        Server Name: systemb
        Server State: Server
        Server Name: systemc
        Server State: Server
```

dsk3 は共用バス上のダイレクト・アクセス入出力デバイスなので、バス上の 3 つのシステムすべてがそのディスクに対するサービスを提供します。共用バス上の任意のメンバがディスクにアクセスする場合は、そのメンバから直接デバイスにアクセスします。

プライベート・バス上のディスクは、接続元のローカル・システムによってサービスが提供されます。たとえば、次に示すように、dsk7 のサーバは systemb です。

```
# drdmgr -a server dsk7
      Device Name: dsk7
      Device Type: Direct Access IO Disk
      Device Status: OK
      Number of Servers: 1
        Server Name: systemb
        Server State: Server
```

テープ・ドライブは、常に単一サーバ・デバイスです。tape0 は共用バス上にあるので、そのバス上のどのメンバもテープ・ドライブのサーバとして機能できます。クラスタの起動時には、テープ・ドライブにアクセスする最初のアクティブ・メンバが、テープ・ドライブのサーバになります。

ディスクに付けられた番号は、クラスタのブート時に systema が最初に起動したことを示します。このサーバでは、最初にプライベート・ディスクを検出してそのディスクにラベルが設定され、次に、共用バスのディスクを検出してそのディスクにラベルが設定されます。systema は最初に起動したので、tape0 のサーバにもなります。これを確認するには、次のコマンドを入力してください。



```
# drdmgr -a server tape0
      Device Name: tape0
      Device Type: Served Tape
      Device Status: OK
      Number of Servers: 1
      Server Name: systema
      Server State: Server
```

tape0 のサーバを systemc に変更するには、drdmgr コマンドを次のように入力してください。

```
# drdmgr -a server=systemc /dev/tape/tape0
```

単一サーバ・デバイスの場合は、サービスを提供しているメンバがアクセス・ノードにもなります。次のコマンドでこれを確認してください。

```
# drdmgr -a accessnode tape0
      Device Name: tape0
      Access Node Name: systemc
```

指定したデバイスに対する、デバイス要求ディスパッチャの SERVER 属性はすべてのクラスタ・メンバで同じですが、ACCESSNODE 属性の値はクラスタ・メンバに固有です。

共用バス上のシステムは、常に同じ共用バス上のダイレクト・アクセス入出力デバイスに対する固有のアクセス・ノードになります。

systemd には共用バスが接続されていないので、共用バス上のダイレクト・アクセス入出力デバイスごとに、デバイスへのアクセス時に systemd が利用するアクセス・ノードを指定できます。アクセス・ノードは、共用バス上のいずれかのメンバでなければなりません。

次のコマンドを実行すると、systemc が、systemd と dsk3 の間のあらゆるデバイス要求ディスパッチャの処理を引き受けるようになります。

```
# drdmgr -h systemd -a accessnode=systemc dsk3
```

#### 9.4.1.1 ダイレクト・アクセス入出力をサポートするデバイス

RAID 対応のディスクはダイレクト・アクセス入出力機能を備えています。RAID (Redundant Array of Independent Disks) コントローラの一部を、以下に示します。

- HSZ80
- HSG60

- HSG80
- RA3000 (HSZ22)
- Enterprise Virtual Array (HSV110)

`clu_create` または `clu_add_member` のコマンドによってシステムがクラスタ・メンバになる時点ですでに設置されている RZ26, RZ28, RZ29, および RZ1CB-CA のディスクは、ダイレクト・アクセス入出力ディスクとして自動的に使用可能になります。これらのディスクのうちいずれか 1 つを後からダイレクト・アクセス入出力ディスクとして追加するには、9.2.3 項の手順を使用しなければなりません。

#### 9.4.1.2 ダイレクト・アクセス入出力ディスクとしての RZ26, RZ28, RZ29, RZ1CB-CA のうちいずれか 1 つの交換

RZ26, RZ28, RZ29, RZ1CB-CA のうちいずれか 1 つのダイレクト・アクセス入出力ディスクを同じタイプのディスクと交換 (たとえば, RZ28-VA を別の RZ28-VA に交換) する場合は、次の手順を実行して、新しいディスクをダイレクト・アクセス入出力ディスクにしてください。

1. ディスクをバスに物理的に接続します。
2. 次のように `hwmgr` コマンドを使用して新しいディスクを走査します。  
  
# `hwmgr -scan comp -cat scsi_bus`  
  
走査が完了するまで 1, 2 分かかります。
3. 交換した旧ディスクと同じデバイス名を新規ディスクに付ける場合は、`hwmgr -redirect scsi` コマンドを実行します。詳細は、`hwmgr(8)` および Tru64 UNIX 『ハードウェア管理ガイド』の「障害の発生した SCSI デバイスの交換」の項を参照してください。
4. 各クラスタ・メンバで、次のように `clu_disk_install` コマンドを入力します。

```
# clu_disk_install
```

---

#### 注意

---

クラスタに多数のストレージ・デバイスが存在する場合は、コマンド `clu_disk_install` の完了まで数分かかる場合があります。

---

#### 9.4.1.3 共用バス上の HSZ ハードウェアのサポート

共用バス上でサポートされているハードウェアの一覧については、TruCluster Server バージョン 5.1B 『*QuickSpecs*』を参照してください。

共用バス上で適切なファームウェア・リビジョンを持たない HSZ を使用した場合は、HSZ に対して同時に複数のアクセスが発生すると、クラスタがハングするおそれがあります。

## 9.5 クラスタでの AdvFS の管理

クラスタでの AdvFS (Advanced file system) は、大部分がスタンドアロン・システムと同様の特徴を示します。ただし、この節で次のようなクラスタ固有のいくつかの注意事項を説明します。

- 新たに追加されたメンバからの AdvFS ファイルの統合 (9.5.1 項)
- クラスタ・ルート・ドメインにファイルセットを 1 つだけ作成する (9.5.2 項)
- メンバのブート・パーティションへファイルセットを追加する (9.5.3 項)
- メンバのルート・ドメインへボリュームを追加しない (9.5.4 項)
- `addvol` および `rmvol` コマンドの使用 (9.5.5 項)
- ユーザとグループのファイル・システムに対するクォータの使用 (9.5.6 項)
- ストレージの接続性と AdvFS ボリュームの理解 (9.5.7 項)

### 9.5.1 新たに追加されたメンバからの AdvFS ファイルの統合

新規メンバをクラスタに追加しますが、その新規メンバには、スタンドアロン・システムとして稼働していたときから AdvFS ボリュームとファイルセットが用意されているとします。このようなボリュームとファイルセットをクラスタに統合するには、次の操作を行う必要があります。

1. クラスタに統合したい `domains#filesets` を列挙している `/etc/fstab` ファイルを変更します。
2. 新規ドメインをクラスタに登録するために、ドメイン情報を手動で `/etc/fdmns` に入力するか、または `advscan` コマンドを実行します。

advscan コマンドについては、advscan(8) を参照してください。  
/etc/fdmns の再構築の例は、Tru64 UNIX 『AdvFS 管理ガイド』の AdvFS  
ファイル・システムのリストアに関する節を参照してください。

### 9.5.2 クラスタ・ルート・ドメインにファイルセットを 1 つだけ作成する

ルート・ドメインである cluster\_root には、ファイルセットが 1 つだけが必要です。cluster\_root にファイルセットを 2 つ以上作成すると (作成自体は可能)、cluster\_root ドメインにフェイルオーバが必要となった場合にパニックが発生することがあります。

このような状況が発生する例として、クローン・ファイルセットをとりあげます。advfs(4) で説明されているように、クローン・ファイルセットは既存のファイルセットを読み取り専用でコピーしたものであり、他のファイルセットと同じようにマウントできます。クラスタ単位のルート (/) のクローンを作成してマウントすると、そのクローン・ファイルセットが cluster\_root ドメインに追加されます。クローン・ファイルセットがマウントされている間に cluster\_root ドメインにフェイルオーバの必要が生じると、クラスタはパニックに陥ります。

---

#### 注意

---

クローン・ファイルセットからクラスタ単位のルートのバックアップを作成する場合は、クローンのマウント時間をできるだけ短くしてください。できるだけ速く、クローン・ファイルセットをマウントし、バックアップを実行して、クローンをアンマウントしてください。

---

### 9.5.3 メンバのブート・パーティションへファイルセットを追加する (非推奨)

メンバのブート・パーティションへのファイルセットの追加は禁止されていませんが、推奨しません。メンバがクラスタから切り離された場合、メンバのブート・パーティションからマウントされたすべてのファイルセットは、強制的にアンマウントされ、再配置できません。

#### 9.5.4 メンバのルート・ドメインへボリュームを追加しない

addvol コマンドを使用してメンバのルート・ドメイン (rootmemberID\_domain#root) にボリュームを追加することはできません。追加したい場合は、まずクラスタからメンバを削除し、次に diskconfig または SysMan を用いて適切にディスクを構成したのちに、そのメンバをクラスタに追加するかたちで戻さなければなりません。メンバのブート・ディスクを構成する場合の要件については、『クラスタ・インストール・ガイド』を参照してください。

#### 9.5.5 クラスタでの addvol コマンドと rmvol コマンドの使用

AdvFS ドメインがローカル・メンバ上またはリモート・メンバ上にマウントされているかどうかにかかわらず、任意のクラスタ・メンバからそのドメインを管理できます。ただし、管理対象ドメインの CFS サーバでないメンバから addvol コマンドまたは rmvol コマンドを使用する場合、コマンドは、rsh を利用してドメインの CFS サーバになっているメンバ上でリモートに実行されます。これにより、次のような結果になります。

- ドメインのサーバでないメンバから addvol または rmvol が入力され、ドメインに対するサービスを提供するメンバに障害が発生した場合、コマンドは、実行されたシステム上で TCP のタイムアウトまで、場合によっては 1 時間もの間ハングするおそれがあります。

このような状況になった場合は、コマンドとコマンドに関連する rsh プロセスを強制終了させて、次のようにしてコマンドを繰り返すことができます。

- ps コマンドでプロセス ID (PID) を取得して、more によって出力をパイプ処理し、addvol または rmvol を (どちらが適切であっても) 検索します。たとえば、次のようになります。

```
# ps -el | more +/addvol
80808001 I + 0 16253977 16253835 0.0 44 0 451700 424K wait pts/0
0:00.09 addvol
80808001 I + 0 16253980 16253977 0.0 44 0 1e6200 224K event pts/0
0:00.02 rsh
808001 I + 0 16253981 16253980 0.0 44 0 a82200 56K tty pts/0
0:00.00 rsh
```

- プロセス ID (この例では、16253977、16253980、および 16253981 の PID) と親プロセス ID (16253977 と 16253980 の PPID) を使用して、addvol または rmvol と rsh プロセスとの関連を確認します。2 つの rsh プロセスは addvol プロセスに関連

付けられています。3つのプロセスすべてを強制終了しなければなりません。

3. 該当するプロセスを強制終了します。この例では、次のようになります。

```
# kill -9 16253977 16253980 16253981
```

4. `addvol` コマンドまたは `rmvol` コマンドを再入力します。`addvol` の場合は、`-F` オプションを入力しなければなりません。`-F` オプションを使用する必要があるのは、ハングした `addvol` コマンドでディスク・ラベルがすでに AdvFS に変更されている可能性があるためです。

もう1つの方法として、ドメインで `addvol` または `rmvol` コマンドを使用する前に、次の操作が実行できます。

1. 次のように `cfsmgr` コマンドを使用して、ドメインの CFS サーバの名前を確認します。

```
# cfsmgr -d domain_name
```

あらゆる CFS ドメインのサーバのリストを取得するには、`cfsmgr` コマンドのみを入力します。

2. サービスを提供しているメンバにログインします。
3. `addvol` または `rmvol` のコマンドを使用します。

- ボリュームの CFS サーバが、`addvol` または `rmvol` の操作の途中で別のメンバにフェイルオーバーした場合は、新しいサーバが部分的な操作をすべて取り消すため、コマンドの再入力が必要になる可能性があります。このコマンドでは、サーバに障害が発生したことを示すメッセージは返されないで、操作を繰り返す必要があります。

コマンドが戻った後に `addvol` または `rmvol` コマンドのターゲット・ドメインに対する `showfdmn` コマンドを入力することをお勧めします。

コマンドが実行されたメンバがドメインのサーバでない場合、`rmvol` および `addvol` コマンドは `rsh` を使用します。`rsh` が機能するためには、省略時のクラスタ別名が `/.rhosts` ファイルに登録されていなければなりません。`/.rhosts` のクラスタ別名のエントリは、完全に修飾されたホスト名または修飾されていないホスト名の形式をとります。ホスト名の代わりに、す

すべてのホストのアクセスを許可する正符合 (+) を指定することもできますが、セキュリティ上の理由からお勧めしません。

`clu_create` コマンドは、クラスタ別名を自動的に `/.rhosts` に登録して、ユーザが介入しなくても `rsh` が動作するようにします。`rsh` が失敗したために、`rmvol` または `addvol` コマンドが失敗した場合には、次のメッセージが返されます。

```
rsh failure, check that the /.rhosts file allows cluster alias access.
```

### 9.5.6 ユーザとグループのファイル・システムに対するクォータのサポート

TruCluster Server では、クォータをサポートしています。この機能を使用することにより、ユーザやグループに代わって、AdvFS ファイル・システムに置けるファイルの数と使用できるディスク・スペースの総量の両方を制限できます。

TruCluster Server 環境でサポートしているクォータは、Tru64 UNIX システムでサポートしているクォータと似ていますが、次の点が異なります。

- CFS (クラスタ・ファイル・システム) はキャッシュされているデータを独自の基準 (タイミングと方法) に基づいて書き込みます。そのため、ハード限界値は絶対ではありません。
- ソフト限界値と猶予期間はサポートされていますが、ユーザがソフト限界値を超過したクライアント・ノードからメッセージを受け取れない場合があります、そのようなメッセージがタイムリーに到達しない場合もあります。
- `quota` コマンドは、クラスタ単位で有効です。ただし、各クラスタ・メンバで `/sys/conf/NAME` システム構成ファイルを編集して、システムにクォータ・サブシステムを組み入れる必要があります。クラスタ・メンバでこの手順を実行しなくても、クォータはそのメンバで有効になりますが、そのメンバから `quota` コマンドを入力することはできません。
- TruCluster Server がサポートしているファイル・システムは、AdvFS ファイル・システムだけです。
- ユーザとグループはクラスタ単位で管理されます。つまり、ユーザおよびグループのクォータも、クラスタ単位で管理されます。

ここでは、TruCluster Server 環境で行うディスク・クォータ管理に特有な事項を説明します。クォータ管理についての一般的な説明は、Tru64 UNIX『システム管理ガイド』を参照してください。

#### 9.5.6.1 クォータのハード限界値

Tru64 UNIX システムでは、それぞれのユーザまたはグループがファイル・システムに置けるファイルの数や使用できるディスクのスペースは、ハード限界値によってその絶対的な上限が制限されます。ハード限界値に達すると、ディスク・スペースの割り当てやファイルの作成ができなくなります。ハード限界値を超えるようなシステム・コールは、クォータ違反となって失敗します。

TruCluster Server 環境でも、スタンドアロンの Tru64 UNIX システムと同じように、ファイル数のハード限界値が強制されます。

ただし、ディスク・スペース全体に対するハード限界値は、それほど厳密には適用されません。CFS では、性能上の理由から、クライアント・ノードでユーザまたはグループのデータをそのデータを提供するメンバと連絡することなく、ある程度キャッシュに置けるようになっています。その量は構成できます。そのため、あるクライアント・ノードで書き込み操作を行ったときに、そのデータがキャッシュに書き込まれるだけでサーバのディスクに書き込まれない状況が発生しますが、CFS は、そのノードに障害が発生した場合を除いて、そのデータが最終的にサーバのディスクに書き込まれることを保証します。

キャッシュされているデータの書き込みは、ディスク・クォータの強制より優先します。つまり、クォータ違反が発生しても、キャッシュ内のデータは違反を無視してディスクに書き込まれます。その後このグループまたはユーザが書き込みを行っても、クォータ違反が修正されるまでキャッシュされません。

クォータ違反が発生している間はデータがキャッシュに追加されないので、全クラスタ・メンバの `quota_excess_blocks` を総計した値以上にハード限界値を超えることはありません。したがって、ユーザまたはグループに対する実際のディスク・スペース・クォータは、ハード限界値に全クラスタ・メンバの `quota_excess_blocks` を足した値になります。

ユーザまたはグループごとにキャッシュできるデータ量は、各メンバの `etc/sysconfigtab` ファイルにある `quota_excess_blocks` の値で決ま



ります。quota\_excess\_blocks の値は 1024 バイトのブロックを 1 単位として表します。省略時の値は 1024 で、1 MB のディスク・スペースです。quota\_excess\_blocks の値は、クラスタ・メンバごとに異なってもかまいません。データのほとんどが置かれるクラスタ・メンバでは quota\_excess\_blocks の値を大きくし、その他のクラスタ・メンバには省略時の quota\_excess\_blocks の値を使用することもできます。

#### 9.5.6.2 quota\_excess\_blocks の値の設定

quota\_excess\_blocks の値は、/etc/sysconfigtab ファイルの cfs スタンザに記述されています。

このファイルは、手動で変更しないでください。変更は、sysconfigdb コマンドで行ってください。このユーティリティは、自動的にすべての変更をカーネルが使用できるようにするとともに、ファイルの構造を保存して、将来アップグレードしたときに正しく組み込めるようにします。

ユーザまたはグループによっては、性能が quota\_excess\_blocks の影響を受けることがあります。この値を低くしすぎると、CFS はキャッシュを有効に使えません。quota\_excess\_blocks の値を 64K より小さくすると、性能に大きく影響します。逆に、quota\_excess\_blocks を高く設定しすぎると、ユーザまたはグループが消費するディスク・スペースの割合が増加します。

quota\_excess\_blocks の値としては、省略時の 1 MB を使用するか、潜在的な上限値がディスク・ブロックの利用に与える影響を考慮して、妥当と思われる分だけ増やすことをお勧めします。設定する値を決める際は、最悪の上限値を考慮してください。最悪の上限値は次の式で求められます。

(admin の指定するハード上限値) +  
(各クライアント・ノードの quota\_excess\_blocks の合計)

CFS はできる限りの処理を行って、ハード・クォータ限界値を超えないようにしています。したがって、最悪の上限に到達することはほとんどありません。

### 9.5.7 ストレージの接続性と AdvFS ボリューム

フェイルオーバ機能が要求される場合は、AdvFS ドメイン内のすべてのボリュームが同じ接続性を備えていなければなりません。次の条件のどちらかに当てはまる場合は、ボリュームの接続性が同じです。

- AdvFS ドメイン内のすべてのボリュームが同じ共用 SCSI バス上にある。
- AdvFS ドメイン内のボリュームは別々の共用 SCSI バス上にあるが、それぞれのバスすべてが同じクラスタ・メンバに接続されている。

`drdmgr` および `hwmgr` コマンドを使用すれば、システムとディスクのサービスの関係がわかります。動作中のメンバ、バス、ストレージ・デバイス、およびそれらの接続をはじめとするクラスタ・ハードウェアの構成をグラフィカルに表示するには、`sms` コマンドを使用して SysMan Station のグラフィック・インタフェースを起動し、次に [Views] メニューから [Hardware] を選択してください。

## 9.6 新しいファイル・システムを作成するときの注意事項

新しいファイル・システムの作成は、そのほとんどがクラスタとスタンドアロンの環境で同じです。Tru64 UNIX 『*AdvFS 管理ガイド*』では、スタンドアロン環境での AdvFS ファイル・システムの作成方法を幅広く説明しています。

クラスタへのディスクの追加については、9.2.3 項を参照してください。

以下に示すのは、新しいファイル・システムを作成する場合の重要なクラスタ固有の注意事項です。

- 最高の可用性が得られるように、AdvFS ドメインでボリュームに使用されるすべてのディスクが同じ接続性を持つことを確実にしてください。  
できるだけ、AdvFS ドメイン内のすべて LSM ボリュームの接続性を同じにしてください。LSM ボリュームと接続性の詳細については、Tru64 UNIX 『*Logical Storage Manager*』を参照してください。
- ディスクが使用中かどうかを調べるときは、そのディスクが以下のいずれかとして使用されていないかどうかを確認してください。
  - クラスタ・クォーラム・ディスク  
データ用にはクォーラム・ディスク上のどのパーティションも使用しないでください。
  - クラスタ単位のルート・ファイル・システム、クラスタ単位の `/var` ファイル・システム、またはクラスタ単位の `/usr` ファイル・システム
  - メンバのブート・ディスク

メンバ・ブート・ディスクの説明とその構成方法については、  
11.1.5 項を参照してください。

- 1 つの /etc/fstab ファイルがクラスタのすべてのメンバに適用されます。

### 9.6.1 ディスクの接続性の確認

最高の可用性が得られるように、AdvFS ドメインでボリュームに使用されるすべてのディスクの接続性が同じであることを確認する必要があります。

ディスクは、次のどちらかの条件を満たす場合は接続性が同じです。

- AdvFS ドメイン内のボリュームに使用されるすべてのディスクが同じ共用 SCSI バス上にある。
- AdvFS ドメイン内のボリュームに使用されるディスクが異なる共用 SCSI バス上にあっても、それぞれのバスがすべて同じクラスタ・メンバに接続されている。

ディスクの接続性を確認する最も簡単な方法は、`sms` コマンドを使用して SysMan Station のグラフィック・インタフェースを起動し、[Views] メニューから [Hardware] を選択することです。

たとえば、図 9-1 では、`pza0` に接続されている SCSI バスは、3 つのクラスタ・メンバすべてによって共用されています。そのバス上のディスクはすべて接続性が同じになります。

`hwmgr` コマンドを使用してクラスタ上のすべてのデバイスを表示してから、さまざまなメンバに接続されているために複数回表示されるディスクを選別することもできます。たとえば、次のようになります。

```
# hwmgr -view devices -cluster
HWID: Device Name      Mfg      Model      Hostname  Location
-----
3: kevm
28: /dev/disk/floppy0c  DEC      3.5in floppy pepicelli fdi0-unit-0
40: /dev/disk/dsk0c     DEC      RZ28M      (C) DEC pepicelli bus-0-targ-0-lun-0
41: /dev/disk/dsk1c     DEC      RZ28L-AS   (C) DEC pepicelli bus-0-targ-1-lun-0
42: /dev/disk/dsk2c     DEC      RZ28       (C) DEC pepicelli bus-0-targ-2-lun-0
43: /dev/disk/cdrom0c   DEC      RRD46      (C) DEC pepicelli bus-0-targ-6-lun-0
44: /dev/disk/dsk13c    DEC      RZ28M      (C) DEC pepicelli bus-1-targ-1-lun-0
44: /dev/disk/dsk13c    DEC      RZ28M      (C) DEC polishham bus-1-targ-1-lun-0
44: /dev/disk/dsk13c    DEC      RZ28M      (C) DEC provolone bus-1-targ-1-lun-0
45: /dev/disk/dsk14c    DEC      RZ28L-AS   (C) DEC pepicelli bus-1-targ-2-lun-0
45: /dev/disk/dsk14c    DEC      RZ28L-AS   (C) DEC polishham bus-1-targ-2-lun-0
45: /dev/disk/dsk14c    DEC      RZ28L-AS   (C) DEC provolone bus-1-targ-2-lun-0
46: /dev/disk/dsk15c    DEC      RZ29B      (C) DEC pepicelli bus-1-targ-3-lun-0
46: /dev/disk/dsk15c    DEC      RZ29B      (C) DEC polishham bus-1-targ-3-lun-0
46: /dev/disk/dsk15c    DEC      RZ29B      (C) DEC provolone bus-1-targ-3-lun-0
```

この出力では、`dsk0`、`dsk1`、および `dsk2` が `pepicelli` のローカル・バスに接続されているプライベート・ディスクです。これらのディスクはどれもフェイルオーバー機能を必要とするファイル・システムに適切ではないので、LSM (Logical Storage Manager) ボリュームとして選択しない方が賢明です。

ディスク `dsk13` (HWID 44)、`dsk14` (HWID 45)、および `dsk15` (HWID 46) は、`pepicelli`、`polishham`、および `provolone` に接続されています。この3つのディスクは、接続性がまったく同じです。

## 9.6.2 利用可能なディスクの調査

ディスクがすでに使用中かどうかを判断したいときは、クォラム・ディスク、クラスタ単位のファイル・システムが含まれているディスク、およびメンバ・ブート・ディスクとスワップ領域を調べてください。

### 9.6.2.1 クォラム・ディスクの場所の調査

クォラム・ディスクの場所を調べるには、`clu_quorum` コマンドを使用します。次の例では、コマンドからの出力の一部は、`dsk10` がクラスタ・クォラム・ディスクであることを示しています。

```
# clu_quorum
Cluster Quorum Data for: deli as of Wed Apr 25 09:27:36 EDT 2001

Cluster Common Quorum Data
Quorum disk:   dsk10h
.
.
.
```

`disklabel` コマンドを使用して、クォラム・ディスクを調べることもできます。クォラム・ディスク内のすべてのパーティションは未使用です。ただし、`fstype cnx` が設定されている `h` パーティションは除きます。

### 9.6.2.2 メンバ・ブート・ディスクとクラスタ単位の AdvFS ファイル・システムの場所の調査

メンバ・ブート・ディスクとクラスタ単位の AdvFS ファイル・システムの場所を確認するには、`/etc/fdmns` ディレクトリ内のファイル・ドメイン・エントリを調べてください。この作業に `ls` コマンドを使用できます。たとえば、次のようになります。

```
# ls /etc/fdmns/*
/etc/fdmns/cluster_root:
dsk3c

/etc/fdmns/cluster_usr:
dsk5c

/etc/fdmns/cluster_var:
dsk6c

/etc/fdmns/projects1_data:
dsk9c

/etc/fdmns/projects2_data:
dsk11c

/etc/fdmns/projects_tools:
dsk12c

/etc/fdmns/root1_domain:
dsk4a

/etc/fdmns/root2_domain:
dsk8a

/etc/fdmns/root3_domain:
dsk2a

/etc/fdmns/root_domain:
dsk0a

/etc/fdmns/usr_domain:
dsk0g
```

このような `ls` コマンドからの出力で、以下のことがわかります。

- ディスク `dsk3` は、クラスタ単位のルート・ファイル・システム (`/`) で使用されます。このディスクは使用できません。
- ディスク `dsk5` は、クラスタ単位の `/usr` ファイル・システムで使用されます。このディスクは使用できません。
- ディスク `dsk6` は、クラスタ単位の `/var` ファイル・システムで使用されます。このディスクは使用できません。
- ディスク `dsk4` , `dsk8` , および `dsk2` は、メンバ・ブート・ディスクです。これらのディスクは使用できません。

disklabel コマンドを使用してメンバ・ブート・ディスクを識別することもできます。それぞれのディスクにはパーティションが3つあり、a パーティションには fstypeAdvFS、b パーティションには fstypeswap、h パーティションには fstypecnx が設定されています。

- ディスク dsk9、dsk11、および dsk12 は、データおよびツールに使用されていると思われます。
- ディスク dsk0 は非クラスタの Tru64 UNIX ベース・オペレーティング・システム用のブート・ディスクです。

非クラスタ・カーネルをブートして修復しなければならない場合に備えて、このディスクは変更しないでください。

### 9.6.2.3 メンバ・スワップ領域の調査

メンバの1次スワップ領域は、常にメンバ・ブート・ディスクのbパーティションです。メンバ・ブート・ディスクの詳細は、11.1.5 項を参照してください。ただし、メンバにさらにスワップ領域を設定する可能性があります。メンバが停止している場合は、メンバのスワップ領域を使用しないように注意してください。ディスクにスワップ領域が設定されているかどうかを確認するには、disklabel -r コマンドを使用します。fstype swap のパーティションに関する出力で「fstype」の列を調べてください。

次の例では、dsk11 のパーティション b がスワップ・パーティションです。

```
# disklabel -r dsk11
.
.
.
8 partitions:
#      size      offset  fstype  [fsize bsize cpgr] # NOTE: values not exact
a:    262144         0   AdvFS          0      0      # (Cyl. 0 - 165*)
b:    401408    262144    swap          0      0      # (Cyl. 165*- 418*)
c:    4110480         0  unused          0      0      # (Cyl. 0 - 2594)
d:    1148976    663552  unused          0      0      # (Cyl. 418*- 1144*)
e:    1148976    1812528  unused          0      0      # (Cyl. 1144*- 1869*)
f:    1148976    2961504  unused          0      0      # (Cyl. 1869*- 2594)
g:    1433600    663552   AdvFS          0      0      # (Cyl. 418*- 1323*)
h:    2013328    2097152   AdvFS          0      0      # (Cyl. 1323*- 2594)
```

### 9.6.3 /etc/fstab の編集

SysMan Station GUI (グラフィカル・ユーザ・インタフェース) を利用して AdvFS ボリュームの作成や構成を行うことができます。ただし、/etc/fstab の編集にコマンド行を使用することにした場合、そのファイルは一度だけ編集すればよく、任意のクラスタ・メンバから編集できます。

/etc/fstab ファイルは、CDSLではありません。1つのファイルがすべてのクラスタ・メンバで使用されます。

## 9.7 CDFS ファイル・システムの管理

クラスタでは、CD-ROM は必ずサービスされるデバイスです。ドライブはローカル・バスに接続されていなければならない、共用バスに接続することはできません。次に、クラスタで CDFS (CD-ROM ファイル・システム) を管理する場合の制約を示します。

- `cddevsuppl` コマンドは、クラスタではサポートされていません。
- 次のコマンドは、CDFS ファイル・システムの CFS サーバになっているクラスタ・メンバから実行された場合にのみ機能します。
  - `cddrec(1)`
  - `cdptrec(1)`
  - `cdsuf(1)`
  - `cdvd(1)`
  - `cdxar(1)`
  - `cdmntsuppl(8)`

CD-ROM をマウントするメンバにかかわらず、ドライブに接続されたメンバは CDFS ファイル・システムの CFS サーバです。

CDFS ファイル・システムを管理するには、次の手順を実行します。

1. `cfsmgr` コマンドを入力して、現在 CDFS に対するサービスを提供しているメンバを確認します。

```
# cfsmgr
```

2. サービスを提供しているメンバにログインします。
3. 適切なコマンドを使用して管理タスクを実行します。

CDFS を操作するライブラリ関数の使用方法については、TruCluster Server 『クラスタ高可用性アプリケーション・ガイド』を参照してください。

## 9.8 ファイルのバックアップとリストア

クラスタ内のユーザ・データのバックアップとリストアは、スタンドアロン・システムの場合と同様です。他のシンボリック・リンクと同様に CDSL

のバックアップとリストアを行います。CDSL のすべてのターゲットをバックアップするには、`/cluster/members` 領域をバックアップしてください。

使用する予定のすべてのリストア・ソフトウェアが、最初にクラスタ・メンバになったシステムの Tru64 UNIX ディスク上で利用できることを確認してください。このディスクをクラスタの緊急修復ディスクとして扱ってください。クラスタからルート・ドメイン `cluster_root` が失われた場合は、Tru64 UNIX ディスクから最初のクラスタ・メンバをブートすれば、`cluster_root` をリストアできます。

`bttape` ユーティリティは、クラスタではサポートされていません。

`clonefs` ユーティリティは(`clonefs(8)` を参照)、アクティブなファイルセットの読み取り専用のコピー (クローン) を作ることで、アクティブなファイルのオンライン・バックアップを行えるようにします。クローン・ファイルセットを作成しマウントした後、`vdump` コマンドや他のサポートされているバックアップ・ユーティリティを使って、クローンをバックアップできます(`dump` コマンドは、AdvFS でサポートされていません)。`clonefs` は、クラスタ・ファイル・システムのバックアップに適しています。クローン・ファイルセットからのクラスタ単位のルートのバックアップは、クローンがマウントされている時間を短縮します。クローン・ファイルセットをマウントし、バックアップを実行して、クローンをアンマウントする時間を可能な限り短くしてください。詳細は、9.5.2 項を参照してください。

### 9.8.1 バックアップするファイルに対する推奨事項

データ・ファイルと以下のファイル・システムを定期的にバックアップします。

- クラスタ単位のルート・ファイル・システム  
ユーザ・データに使用するのと同じバックアップとリストア方式を採用してください。
- クラスタ単位の `/usr` ファイル・システム  
ユーザ・データに使用するのと同じバックアップとリストア方式を採用してください。
- クラスタ単位の `/var` ファイル・システム  
ユーザ・データに使用するのと同じバックアップとリストア方式を採用してください。



TruCluster Server をインストールする前に、AdvFS を使用しており、`/usr (usr_domain#var)` に `/var` が配置されている場合は、インストール処理により、`/var` は自身のドメイン (`cluster_var#var`) に移動されます。

この移動のため、`/var` を `/usr` から独立したファイル・システムとしてバックアップする必要があります。

- メンバ・ブート・ディスク

メンバ・ブート・ディスクのバックアップとリストアの特別な注意事項は、11.1.5 項を参照してください。

## 9.9 スワップ領域の管理

スワップ・エントリを `/etc/fstab` に設定しないでください。Tru64 UNIX バージョン 5.0 では、スワップ・デバイスのリストが `/etc/fstab` ファイルから `/etc/sysconfigtab` ファイルに移動されました。また、スワップ割り当てを示すために `/sbin/swapdefault` ファイルを使用できなくなりました。したがって、このような用途にも `/etc/sysconfigtab` ファイルを使用してください。スワップ・デバイスとスワップ割り当てモードは、ベース・オペレーティング・システムのインストール中に自動的に `/etc/sysconfigtab` ファイルに設定されます。詳細は、Tru64 UNIX 『システム管理ガイド』および `swapon(8)` を参照してください。

各メンバのスワップ情報は、該当するメンバの `sysconfigtab` ファイルに設定してください。スワップ情報はクラスタ単位の `/etc/fstab` ファイルには設定しないでください。

`sysconfigtab` におけるスワップ情報は、`swapdevice` 属性で識別されます。スワップ情報の形式は、次のとおりです。

```
swapdevice=disk_partition,disk_partition,...
```

たとえば、次のようになります。

```
swapdevice=/dev/disk/dsk1b,/dev/disk/dsk3b
```

`/etc/fstab` でスワップ・エントリを指定しても、`/etc/fstab` がメンバ固有のファイルではなく、クラスタ単位のファイルなので、そのエントリはクラスタでは機能しません。スワップが `/etc/fstab` で指定されている場合は、ブートしてクラスタを形成する最初のメンバが、`/etc/fstab` 内の

すべてのファイル・システムを読み取り，マウントします。他のメンバには，そのスワップ領域はまったくわかりません。

/etc/sysconfigtab ファイルはコンテキスト依存シンボリック・リンク (CDSL) なので，各メンバはそれぞれ固有のスワップ・パーティションを見つけてマウントできます。インストレーション・スクリプトでは，メンバごとに 1 つのスワップ・デバイスを自動的に構成して，swapdevice= エントリが，該当するメンバの sysconfigtab ファイル内に設定されます。

さらにスワップ領域を追加したい場合は，swapon で新規パーティションを指定してから，リブート後にパーティションが利用可能になるように，エントリを sysconfigtab 内に設定してください。たとえば，スワップ用に dsk1b をすでに使用しているメンバの 2 次スワップ・デバイスとして使用できるように dsk3b を構成するには，次のコマンドを入力してください。

```
swapon -s /dev/disk/dsk3b
```

次に，該当するメンバの /etc/sysconfigtab を編集し，/dev/disk/dsk3b を追加してください。/etc/sysconfigtab 内の最後のエントリは，次のようになります。

```
swapdevice=/dev/disk/dsk1b,/dev/disk/dsk3b
```

### 9.9.1 性能向上のためのスワップ・デバイスの設定

共用バス上のデバイスにメンバのスワップ領域を設定すると，バスの入出力トラフィックがさらに増加します。これを回避するために，メンバのローカル・バス上のディスクにスワップを設定できます。

メンバに対してローカルのスワップを設定した際の唯一のマイナス面は，アダプタの障害時に稀に発生するおそれのある，スワップ・ディスクへのメンバのパスが失われるという問題です。パスが失われた場合は，メンバに障害が発生したことになります。スワップ・ディスクが共用バス上にある場合は，少なくとも 1 つのメンバにそのディスクへのパスが設定されている限り，メンバはそのスワップ・パーティションを使用できます。

### 9.10 ブート・パラメータによる問題の修正

クラスタ・メンバが，メンバのルート・ドメイン (rootN\_domain) におけるパラメータの問題のためにブートに失敗する場合は，そのドメインを動作中のメンバ上にマウントして，パラメータに必要な変更を加えることができます。ただし，その起動していないメンバをブートする前に，動作中の

クラスタ・メンバから新たに更新されたメンバのルート・ドメインをアンマウントしなければなりません。

アンマウントしなければ、クラッシュして、次のメッセージが表示される場合があります。

```
cfs_mountroot: CFS server already exists for node boot
partition.
```

詳細は、11.1.10 項を参照してください。

## 9.11 クラスタでの verify ユーティリティの使用

verify ユーティリティでは、AdvFS ファイル・システムのオンディスク・メタデータ構造が検査されます。このユーティリティを使用する前に、検査対象のファイル・ドメイン内のファイルセットすべてをアンマウントしなければなりません。

verify ユーティリティを実行し、実行先のクラスタ・メンバに障害が発生した場合は、外部マウントが残るおそれがあります。このようになるのは、verify ユーティリティにより、検査対象ドメイン内のファイルセットの一時的なマウントが作成されるためです。単一システムでは、この一時的なマウントは、ユーティリティの実行中にシステムに障害が発生すれば削除されますが、クラスタでは一時的なマウントは、別のクラスタ・メンバにフェイルオーバーされます。このように一時的なマウントがフェイルオーバーされるため、そのマウントを削除するまでは、ファイルセットもマウントできなくなります。

verify を実行すると、ドメイン内のファイルセットごとにディレクトリが作成され、対応するディレクトリで各ファイルセットがマウントされます。ディレクトリは、`/etc/fdmns/domain/set_verify_XXXXXX` のように名前が付けられます。ここで、XXXXXX は固有の ID です。

たとえば、ドメイン名が `dom2` で、`dom2` 内のファイルセットが `fset1`、`fset2`、および `fset3` の場合は、次のようにコマンドを入力してください。

```
# ls -l /etc/fdmns/dom2
total 24
lrwxr-xr-x 1 root system 15 Dec 31 13:55 dsk3a -> /dev/disk/dsk3a
lrwxr-xr-x 1 root system 15 Dec 31 13:55 dsk3d -> /dev/disk/dsk3d
drwxr-xr-x 3 root system 8192 Jan 7 10:36 fset1_verify_aacTxa
drwxr-xr-x 4 root system 8192 Jan 7 10:36 fset2_verify_aacTxa
drwxr-xr-x 3 root system 8192 Jan 7 10:36 fset3_verify_aacTxa
```

フェイルオーバーされたマウントをクリーンアップするには、次の手順を実行します。

1. 次のようにして `/etc/fdmns` 内のファイルセットをすべてアンマウントします。

```
# umount /etc/fdmns/**/*.verify_*
```

2. 次のコマンドで、フェイルオーバーされたマウントをすべて削除します。

```
# rm -rf /etc/fdmns/**/*.verify_*
```

3. `verify` ユーティリティが正常に完了した後に、ファイルセットを再マウントします。

`verify` についての詳しい説明は、`verify(8)` を参照してください。

### 9.11.1 クラスタ・ルートでの `verify` ユーティリティの使用

`verify` ユーティリティは、稼働状態のドメインで実行できるように変更されています。`-a` オプションを使用して、クラスタ・ルート・ファイル・システム `cluster_root` を検査してください。

検査対象のドメインに対するサービスを提供しているメンバ上で `verify -a` ユーティリティを実行しなければなりません。`cfsmgr` コマンドを使用して、ドメインに対するサービスを提供しているメンバを確定してください。

`-a` オプションを指定して `verify` を実行すると、ドメインの検査のみが実行されます。稼働状態のドメインに対して修正を行うことはできません。`-f` と `-d` のオプションを `-a` オプションとともに使用することできません。

---

## クラスタでの LSM (Logical Storage Manager) の使用

クラスタで単一システムの場合と同様に LSM を使用できます。クラスタ構成とスタンドアロン構成で同じ LSM ソフトウェア・サブセットが使用されます。

LSM についての詳しい説明は、Tru64 UNIX 『*Logical Storage Manager*』を参照してください。また、LSM ソフトウェアのインストールについては、Tru64 UNIX 『インストレーション・ガイド』を参照してください。

クラスタでは、LSM は次のような特長を備えています。

- 高可用性

クラスタ・メンバに障害が発生しても、クラスタ自体が継続的に動作してストレージへの物理パスが利用できる限り、LSM の動作は継続します。

- 性能

- クラスタ環境における入出力では、LSM ボリュームにさらに LSM 入出力オーバーヘッドが生じることはありません。

LSM は完全対称の共用入出力モデルに従っています。このモデルではすべてのメンバが LSM 構成を共用し、各メンバにはプライベートなダーティ・リージョン・ロギング機能が設定されます。

- ディスク・グループは、すべてのクラスタ・メンバが同時に使用できます。
- 1 つの共用ディスク・グループ `rootdg` があります。
- どのメンバも LSM 入出力をすべて直接処理できるので、別のクラスタ・メンバに入出力を渡して処理してもらう必要がありません。

- 管理の容易さ

LSM 構成は、どのメンバからでも管理できます。

## 10.1 クラスタとスタンドアロン・システムにおける LSM の管理の違い

クラスタでの LSM には、次の制約が適用されます。

- LSM ボリュームは、個々のメンバのブート・パーティションには使用できません。
- LSM を使用してクォーラム・ディスクまたはそのディスク上のパーティションをミラー化することはできません。
- LSM Redundant Array of Independent Disks (RAID) 5 ボリュームは、クラスタではサポートされません。
- `cluster_root`、`cluster_usr`、または `cluster_var` ドメインを LSM 制御下に配置するには、`volmigrate` コマンドを使用しなければなりません。

クラスタにおける次の LSM の動作は、単一システム・イメージ・モデルと異なります。

- `volstat` コマンドの返す統計情報は、コマンドの実行対象のメンバのみが該当します。
- LSM の一部でないディスク（つまり、`autoconfig` ディスク）に対して `voldisk list` コマンドを実行する場合は、メンバによっては結果が異なる可能性があります。

このような結果の違いは、多くの場合、使用不能ディスク・グループに限定されます。たとえば、あるメンバは使用不能ディスク・グループを示すが、別のメンバは同じディスク・グループをまったく示さないという場合があります。また、クラスタでは、`voldisk list` コマンドはコマンドを実行したメンバに直接接続された非 LSM ディスクのみを表示します。

## クラスタのトラブルシューティング

この章では次の項目について説明します。

- クラスタに関する問題の解決方法の提案 (11.1 節)
- クラスタの構成と管理のヒント (11.2 節)

### 11.1 問題の解決

ここでは、クラスタの日常の運用時に生じるおそれのある問題の解決策について説明します。

#### 11.1.1 ライセンスなしのシステムのブート

TruCluster Server ライセンスのないシステムをブートできます。このシステムはクラスタに参加し、マルチユーザ・モードでブートしますが、root のみ (最大 2 ユーザ) がログインできます。CAA (Cluster Application Availability) デーモン `caad` は起動されません。システムから、ライセンスの登録を通知するライセンス・エラー・メッセージが表示されます。このような方法でライセンス・チェックが実施され、緊急時のシステムに対するブート、ライセンス設定、および修復を行うことができるようになります。

#### 11.1.2 メンバが動作したままになるシャットダウン

クラスタ・シャットダウン (`shutdown -c`) で 1 つ以上のメンバが動作したままになる場合があります。このような場合は、すべてのメンバをシャットダウンしてクラスタ・シャットダウンを完了しなければなりません。

各メンバに対するポートが 1 でクォーラム・ディスクなしの 3 メンバ・クラスタが構成されているとします。クラスタ・シャットダウンの間に、最後から 2 番目のメンバが停止すると、クォーラムを喪失します。クォーラム・チェックが有効になっている場合は、動作中の最後のメンバは処理がすべて中断され、クラスタ・シャットダウンが完了しません。

このような状況での手詰まりを回避するために、クラスタ・シャットダウン・プロセスの起動時にクォーラム・チェックが無効にされます。クラスタ・シャットダウンの間にメンバがシャットダウンできなかった場合、そのメンバは正常に機能しているクラスタ・メンバのように見えますが、そうではありません。この理由は、クォーラム・チェックが無効になっているためです。したがって、手動でシャットダウン・プロセスを完了しなければなりません。

このシャットダウン手順は、引き続き稼働しているシステムの状態によって次のように異なります。

- システムがハングしており、コンソールからのコマンドを処理できない場合は、システムを停止してクラッシュ・ダンプを生成します。
- システムがハングしていない場合は、`/sbin/halt` を使用してシステムを停止します。

### 11.1.3 環境のモニタリングとクラスタのシャットダウン

`envconfig` コマンド (`envconfig(8)` を参照) には、`ENVMON_SHUTDOWN_SCRIPT` 変数が含まれています。この変数には、シャットダウンする状況になったとき、`envmond` デーモンを実行するユーザ定義スクリプトへのパスを指定します。クラスタでこのスクリプトを使う場合、スクリプトを実行した後、クラスタがクォーラムを失わないようにする必要があります。作成する `ENVMON_SHUTDOWN_SCRIPT` スクリプトでは、シャットダウンするとメンバがクォーラムを失うことになるかどうかを確実に調べる必要があります。クォーラムを失う場合には、残りのクラスタ・メンバをシャットダウンします。

たとえば、5 つのメンバで構成されるクラスタがあるとします。また、すべてのクラスタ・メンバが同じコンピュータ室に配置されていて、空調設備が故障したとします。気温がシステムの許容限度を超えた場合、クラスタ・メンバはそれぞれで気温が高すぎることを感知し、`ENVMON_SHUTDOWN_SCRIPT` スクリプトを呼び出します。3 番目のメンバがクラスタを離脱したときに、クラスタはクォーラムを失い ( $\text{クォーラム・ポート} = (\text{クラスタ期待ポート} + 2) / 2$  (小数点以下切捨て))、2 つの残りのメンバがすべての処理を停止します。その結果として、これらの残りのメンバが過熱した研究室の中で稼働し続けます。



クラスタ・メンバの喪失がクォーラムの喪失の原因になる場合、残りのクラスタ・メンバをシャットダウンします。

ENVMON\_SHUTDOWN\_SCRIPT スクリプトはすべてのケースで shutdown -c を使用してシャットダウンを実行できますが、この方法は推奨できません。

次のサンプル・スクリプトは、現在のメンバをシャットダウンすると、クォーラムを失うことになるかどうかを判定します。

```
#!/usr/bin/ksh -p
#
typeset currentVotes=0
typeset quorumVotes=0
typeset nodeVotes=0

clu_get_info -q
is_cluster=?

if [ "$is_cluster" = 0 ]
then

    # The following code checks whether it is safe to shut down
    # another member. It is considered safe if
    # the cluster would not lose quorum if a member shuts down.
    # If it's not safe, shut down the entire cluster.

    currentVotes=$(sysconfig -q cnx current_votes | \
sed -n 's/current_votes.* //p')

    quorumVotes=$(sysconfig -q cnx quorum_votes | \
sed -n 's/quorum_votes.* //p')

    nodeVotes=$(sysconfig -q cnx node_votes | \
sed -n 's/node_votes.* //p')

    # Determine if this node is a voting member

    if [ "$nodeVotes" -gt 0 ]
    then

        # It's a voting member, see if we'll lose quorum.

        if [[ $((currentVotes-1)) -ge ${quorumVotes} ]]
        then
            echo "shutdown -h now..."
        else
            echo "shutdown -c now..."
        fi
    else
        echo "This member has no vote...shutdown -h now..."
    fi

else
    # not in a cluster...nothing to do
    exit 0
fi
```

#### 11.1.4 ブート時の CFS エラーの処理

システムのブート時にクラスタ単位のルート (/) が初めてマウントされると、CFS で次の警告メッセージが生成される場合があります。

```
"WARNING:cfs_read_advfs_quorum_data: cnx_disk_read failed with error-number
```

通常は、`error-number` は EIO の値です。

このメッセージには、次のメッセージが伴います。

```
"WARNING: Magic number on ADVFS portion of CNX partition on quorum disk \
is not valid"
```

これらのメッセージは、ブート・メンバによる、クォーラム・ディスクの CNX パーティションのデータへのアクセスで問題が生じていることを示しています。メッセージには、`cluster_root` ドメインのデバイス情報も含まれています。メッセージが表示されるのは、ブート・メンバがクォーラム・ディスクにアクセスできない場合で、その理由はクラスタが意図的にこのように構成されているか、またはバスの障害のどちらかです。前者の理由の場合、メッセージを情報とみなすことができます。後者の理由の場合は、バスの障害の原因に対して処置を講ずる必要があります。

メッセージは、クォーラム・ディスク自体に問題があることを示す場合があります。クォーラム・ディスクのハードウェア・エラーもレポートされている場合は、そのディスクを交換してください。クォーラム・ディスクの交換については、4.5.1 項を参照してください。

エラー番号については、`errno(5)` を参照してください。EIO については、`errno(2)` を参照してください。

#### 11.1.5 メンバのブート・ディスクのバックアップと修復

メンバのブート・ディスクには、3 つのパーティションがあります。表 11-1 は、このパーティションの詳細を示しています。

---

##### 注意

---

メンバのブート・パーティションにファイルセットを追加することは禁止しませんが、推奨しません。メンバがクラスタを離脱する場合、メンバのブート・パーティションからマウントされたすべてのファイルセットは、強制的にアンマウントされ、再配置することはできません。

---

表 11-1: メンバ・ブート・ディスク・パーティション

パーティション	内容
a	AdvFS (Advanced File System) ブート・パーティション , メンバ・ルート・ファイル・システム
b	スワップ・パーティション (a と h のパーティションの間の全領域)
h	CNX バイナリ・パーティション AdvFS と LSM (Logical Storage Manager) では , それぞれが機能するために重要な情報が h パーティションに格納される。この情報には , ディスクがメンバかクォーラム・ディスクかを表す情報 , およびクラスタのルート・ファイル・システムが配置されているデバイスの名前が含まれている。

メンバのブート・ディスクが破損している場合や利用できなくなった場合は , クラスタにメンバをリストアするために h パーティションの情報が必要です。clu\_bdmgr コマンドを使用すれば , メンバ・ブート・ディスクを構成して , メンバ・ブート・ディスクに関するデータの保存やリストアを行うことができます。

clu\_bdmgr コマンドを使用して , 次の作業を行うことができます。

- 新しいメンバ・ブート・ディスクを構成する。
- メンバ・ブート・ディスクの h パーティション上の情報をバックアップする。
- ファイルからのデータ , または現在利用可能なメンバのブート・ディスクの h パーティションからのデータで h パーティションを修復する。

このコマンドの詳細は , clu\_bdmgr(8) を参照してください。

メンバがブートするときは必ず , clu\_bdmgr により , そのメンバのブート・ディスクの h パーティションのコピーが自動的に保存されます。データは , /cluster/members/memberID/boot\_partition/etc/clu\_bdmgr.conf に保存されます。

一般に , すべてのメンバ・ブート・ディスクの h パーティションには , 同じデータが格納されます。ただし , 次の 2 つの場合は除きます。

- h パーティションの内容が破損している場合。

- メンバのブート・ディスクがメンバのプライベート・バス上にあり、ブート・ディスクの `h` パーティションの内容に影響を及ぼす更新がクラスタで行われたときにそのメンバが停止している場合。停止中のメンバのディスクはプライベート・バス上にあるので、`h` パーティションを更新できません。

メンバのブート・ディスクは、共用 SCSI バスに接続することをお勧めします。`h` パーティションを確実に最新にするほかに、この構成では、メンバをブートできない場合にブート・ディスクの診断と問題に対して処置を講ずることができます。

メンバのブート・ディスクが破損している場合は、`clu_bdmgr` を使用すれば、そのディスクの修復や交換が可能です。クラスタが稼働していない場合でも、少なくとも 1 つのクラスタ・メンバ上のクラスタ化されたカーネルをブートできる間は、`clu_bdmgr` コマンドを使用できます。

クラスタに新しいディスクを追加する方法については、9.2.3 項を参照してください。

メンバのブート・ディスクを修復するには、最初にブート・パーティションをバックアップしなければなりません。その 1 つの方法は、各メンバのブート・パーティションのダンプ・イメージ用のディスク・スペースを、共用 `/var` ファイル・システムに割り当てることです。

`member3` のブート・パーティションのダンプ・イメージをメンバ固有のファイル `/var/cluster/members/member3/boot_part_vdump` に保存するには、次のコマンドを入力します。

```
# vdump -0Df /var/cluster/members/member3/boot_part_vdump \
/var/cluster/members/member3/boot_partition
```

#### 11.1.5.1 メンバのブート・ディスクの修復例

次の手順は、`vdump` で保存されたファイルを使用してブート・ディスクを交換する方法を示しています。この手順では、次のような仮定をします。

- `member3` のブート・ディスクが `dsk3` である。
- クラスタに新しいメンバのブート・ディスクを既に追加しており、この交換するディスクの名前が `dsk5` である。

ディスクを追加する処理では、`hwmgr -scan comp -cat scsi_bus` コマンドを使用して、すべてのメンバが新しいディスクを認識できるよ

うにします。ディスクをクラスタに追加する方法については、9.2.3 項を参照してください。

---

#### 注意

---

メンバのブート・ディスクは、常にクラスタ・メンバすべてによって共用されるバスに接続する必要があります。このように配置すれば、少なくとも1つのクラスタ・メンバをブートできる間は、どのメンバのブート・ディスクでも修復できます。

---

1. `clu_get_info` を使用して、`member3` が停止しているかどうかを判断します。

```
# clu_get_info -m 3
Cluster memberid = 3
Hostname = member3.zk3.dec.com
Cluster interconnect IP name = member3-mc0
Member state = DOWN
```

2. 新しいディスク (この例では `dsk5`) を `member3` の交換ブート・ディスクとして選択します。`member3` のブート・ディスクは `dsk3` なので、`dsk5` が `member3` の新しいディスクとして使用されるように、`member3` の `/etc/sysconfigtab` を編集する必要があります。

`dsk5` を `member3` のブート・ディスクとして構成するには、次のコマンドを入力します。

```
# /usr/sbin/clu_bdmgr -c dsk5 3
The new member's disk, dsk5, is not the same name as the original disk
configured for domain root3_domain. If you continue the following
changes will be required in member3's/etc/sysconfigtab file:
vm:
  swapdevice=/dev/disk/dsk5b
clubase:
  cluster_seqdisk_major=19
  cluster_seqdisk_minor=175
```

3. `member3` のルート・ドメイン (新しい `dsk5` 上) をマウントし、`member3` の `/etc/sysconfigtab` を編集して、ブート・パーティションをリストアします。

```
# mount root3_domain#root /mnt
```

4. ブート・パーティションをリストアします。

```
# vrestore -xf /var/cluster/members/member3/boot_part_vdump -D /mnt
```

5. member3 の /etc/sysconfigtab を編集します。

```
# cd /mnt/etc
# cp sysconfigtab sysconfigtab-bu
```

clu\_bdmgr コマンドからの出力に指示されているとおり、  
vm スタンザの swapdevice 属性ならびに clubase スタンザの  
cluster\_seqdisk\_major 属性と cluster\_seqdisk\_minor 属性  
の値を変更します。

```
swapdevice=/dev/disk/dsk5b
clubase:
cluster_seqdisk_major=19
cluster_seqdisk_minor=175
```

6. h パーティションの CNX 情報をリストアします。

```
# /usr/sbin/clu_bdmgr -h dsk5
```

h パーティション情報は、clu\_bdmgr コマンドを実行するクラスタ・メンバから dsk5 の h パーティションにコピーされます。

クラスタ全体が停止している場合は、クラスタ化されたカーネルからいずれかのメンバをブートする必要があります。単一メンバのクラスタが稼働した後は、h パーティションの CNX 情報を /mnt/etc/clu\_bdmgr.conf から member3 の新規ブート・ディスク dsk5 にリストアできます。次のコマンドを入力します。

```
# /usr/sbin/clu_bdmgr -h dsk5 /mnt/etc/clu_bdmgr.conf
```

7. member3 のルート・ドメインをアンマウントします。

```
# umount root3_domain#root /mnt
```

8. member3 をクラスタ内にブートします。

9. 必要であれば、member3 で consvar -s bootdef\_dev disk\_name コマンドを使用して、bootdef\_dev 変数を新しいディスクに設定します。

### 11.1.6 ブート時の cluster\_root の指定

ブート時に、cluster\_root (クラスタ・ルート・ファイル・システム) のマウントにクラスタが使用するデバイスを指定することができます。この機能を使用するのは、障害回復のために新しいクラスタ・ルートでブートする必要があるときだけです。

CFS (クラスタ・ファイル・システム) カーネル・サブシステムでは、3 つまでの `cluster_root` デバイスに対する主番号と副番号を指定するために、6 つの属性をサポートしています。障害回復のために使用している `cluster_root` ドメインは複数のボリュームで構成されている可能性があるため、次のように `cluster_root` デバイスを 3 つまで指定することができます。

- `cluster_root_dev1_maj`  
1 つの `cluster_root` デバイスの主デバイス番号
- `cluster_root_dev1_min`  
同じ `cluster_root` デバイスの副デバイス番号
- `cluster_root_dev2_maj`  
2 番目の `cluster_root` デバイスの主デバイス番号
- `cluster_root_dev2_min`  
2 番目の `cluster_root` デバイスの副デバイス番号
- `cluster_root_dev3_maj`  
3 番目の `cluster_root` デバイスの主デバイス番号
- `cluster_root_dev3_min`  
3 番目の `cluster_root` デバイスの副デバイス番号

これらの属性を使用するには、クラスタをシャットダウンし、あるメンバを対話的にブートして適切な `cluster_root_dev` の主番号と副番号を指定します。メンバをブートすると、メンバのブート・ディスクの CNX パーティション (h パーティション) は、`cluster_root` デバイスのロケーションで更新されます。クラスタにクォーラム・ディスクがある場合は、その CNX パーティションも更新されます。その他のノードがクラスタの中にブートされるときに、それらのメンバのブート・ディスク情報も更新されます。

たとえば、`dsk6b` と `dsk8g` を構成する 2 つのボリュームのファイル・システムの `cluster_root` を使用するとします。`dsk6b` の主番号は 19、副番号は 227 とします。`dsk8g` の主番号は 19、副番号は 221 とします。次のようにクラスタをブートします。

#### 1. あるメンバを対話的にブートします。

```
>>> boot -fl "ia"  
(boot dkb200.2.0.7.0 -flags ia)
```

```

block 0 of dkb200.2.0.7.0 is a valid boot block
reading 18 blocks from dkb200.2.0.7.0
bootstrap code read in
base = 200000, image_start = 0, image_bytes = 2400
initializing HWRPB at 2000
initializing page table at fff0000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

:
:

Enter kernel_name [option_1 ... option_n]
Press Return to boot default kernel
'vmunix':vmunix cfs:cluster_root_dev1_maj=19 \
cfs:cluster_root_dev1_min=227 cfs:cluster_root_dev2_maj=19 \
cfs:cluster_root_dev2_min=221 Return

```

## 2. その他のクラスタ・メンバをブートします。

これらの属性を使用してクラスタのルート・ファイル・システムを回復させる方法についての詳細は、11.1.7 項および 11.1.8 項を参照してください。

### 11.1.7 クラスタの既存のディスクへのクラスタ・ルート・ファイル・システムの回復

次のすべてにあてはまる場合に、ここで記述する回復手順を使用してください。

- クラスタ・ルート・ファイル・システムが破損している場合、または使用できない場合
- 最新のクラスタ・ルート・ファイル・システムのバックアップがある場合

バックアップは、クラスタ・ルート・ファイル・システムが障害が発生した時点で認識していたディスク・ストレージ環境を反映していなければなりません。たとえば、バックアップを作成した以降に、デバイスを削除したり (hwmgr del を使用)、リダイレクトしたり (hwmgr redirect を使用)、リフレッシュしたり (hwmgr refresh comp を使用) してはいけません。

クラスタ・ルート・ファイル・システムのバックアップが現在のディスク・ストレージ環境を反映していない場合、このプロシージャはパニックを引き起こします。このパニックが起これば、クラスタ・ファイル・システムを回復できなくなり、clu\_create コマンドを使用してクラスタを再作成しなければならなくなります。



- 全クラスタ・メンバにアクセス可能な共用バス上のディスク (複数可) が、ファイル・システムをリストアするために使用でき、かつ、このディスクがルート・ファイル・システムの問題発生前にはクラスタ構成に組み込まれていた場合

この手順は次のことを想定しています。

- クラスタ・ルート (`cluster_root`) ファイル・システムをバックアップするのに、コマンド `vdump` を使用した。  
別のバックアップ・ツールを使用した場合は、適切なツールを使用してファイル・システムをリストアしてください。
- 少なくとも 1 つのメンバは次のようなデバイスにアクセスできる。
  - ブート可能な Tru64 UNIX ディスク  
ブート可能なベース・ディスクが使用できない場合は、クラスタ・メンバのローカル・ディスク上に Tru64 UNIX をインストールします。クラスタ上にインストールされた Tru64 UNIX と同じバージョンである必要があります。
  - 対象メンバ用のメンバ・ブート・ディスク (この例では `dsk2a`)
  - クラスタのルート・バックアップ用デバイス
- クラスタの全メンバが停止している。

クラスタ・ルートをリストアするには、次のように行います。

1. Tru64 UNIX ディスクでシステムをブートします。

この手順では、このシステムを `member 1` と想定します。

2. このシステムの、新しいクラスタ・ルートになるデバイスの名前が、クラスタのそのデバイスの名前と異なる場合は、コマンド `dsfmgr -m` を使用してデバイス名を変更し、クラスタのデバイスの名前が一致するようにします。

たとえば、クラスタの新しいクラスタ・ルートになるデバイスの名前が `dsk6b` で、システムのそのデバイスの名前が `dsk4b` の場合、デバイスの名前を次のコマンドで変更します。

```
# dsfmgr -m dsk4 dsk6
```

3. 必要に応じて、ディスクにパーティションを設定し、ディスクがクラスタ・ルートとして、パーティション・サイズとファイル・システム・タイプが適切になるようにします。
4. 新しいクラスタ・ルート用の新規ドメインを作成します。

```
# mkfdmn /dev/disk/dsk6d cluster_root
```
5. そのドメインのルート・ファイルセットを作成します。

```
# mkfset cluster_root root
```
6. この復旧手順では、cluster\_root に 3 つまでのボリュームを使用できます。復旧が完了したら、新しいボリュームをクラスタ・ルートに追加することができます。この例の場合、1 つのボリューム (dsk6b) のみを追加します。

```
# addvol /dev/disk/dsk6b cluster_root
```
7. 新しいクラスタ・ルートになるドメインをマウントします。

```
# mount cluster_root#root /mnt
```
8. バックアップ・メディアからクラスタ・ルートをリストアします (vdump 以外のバックアップ・ツールを使用した場合は、vrestore の代りに適切なリストア・ツールを使用します)。

```
# vrestore -xf /dev/tape/tape0 -D /mnt
```
9. 新しくリストアされたファイル・システムの /etc/fdmns/cluster\_root を変更し、新しいデバイスを参照するようにします。

```
# cd /mnt/etc/fdmns/cluster_root
# rm *
# ln -s /dev/disk/dsk6b
```
10. コマンド file を使用して、新しい cluster\_root デバイスの主番号と副番号を書き出しておきます (cluster\_root デバイスの主番号と副番号の使用についての詳細は、11.1.6 項を参照してください)。

たとえば、次のように行います。

```
# file /dev/disk/dsk6b
/dev/disk/dsk6b:          block special (19/221)
```
11. システムをシャットダウンし、新しいクラスタ・ルート的主番号と副番号を指定して対話的にリブートします。11.1.6 項では、ブート時のクラスタ・ルートの指定方法について説明しています。

## 注意

4.10.2 項で説明されているように、メンバをブートするには、おそらく、期待ボードを調整する必要があります。

```
>>> boot -fl "ia"
(boot dkb200.2.0.7.0 -flags ia)
block 0 of dkb200.2.0.7.0 is a valid boot block
reading 18 blocks from dkb200.2.0.7.0
bootstrap code read in
base = 200000, image_start = 0, image_bytes = 2400
initializing HWRPB at 2000
initializing page table at fff0000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

:

Enter kernel_name [option_1 ... option_n]
Press Return to boot default kernel
'vmunix':vmunix cfs:cluster_root_dev1_maj=19 \
cfs:cluster_root_dev1_min=221 Return
```

メンバをブートすると、メンバのブート・ディスクの CNX パーティション (h パーティション) は、cluster\_root デバイスの位置で更新されます。クラスタにクォーラム・ディスクがある場合、その CNX パーティションも更新されます。クラスタに他のノードをブートするときには、メンバのブート・ディスク情報も更新されます。

12. その他のクラスタ・メンバをブートします。

### 11.1.8 新しいディスクへのクラスタ・ルート・ファイル・システムの回復

クラスタの既存のディスク以外を使用して cluster\_root を回復する手順は複雑です。回復させる前に、新しいクラスタ・ルート・ディスクとして使用できる、クラスタに既にインストールされているディスクを探してみてください。見つかったら、11.1.7 項の手順を実行してください。

次のような場合に、ここで記述する回復手順を使用します。

- クラスタ・ルート・ファイル・システムが破損している場合、または使用できない場合

- 最新のクラスタ・ルート・ファイル・システムのバックアップがある場合  
バックアップは、クラスタ・ルート・ファイル・システムが障害が発生した時点で認識していたディスク・ストレージ環境を反映していなければなりません。たとえば、バックアップを作成した以降に、デバイスを削除したり (`hwmgr del` を使用)、リダイレクトしたり (`hwmgr redirect` を使用)、リフレッシュしたり (`hwmgr refresh comp` を使用) してはなりません。

クラスタ・ルート・ファイル・システムのバックアップが現在のディスク・ストレージ環境を反映していない場合、このプロシージャはパニックを引き起こします。このパニックが起こると、クラスタ・ファイル・システムを回復できなくなり、`clu_create` コマンドを使用してクラスタを再作成しなければならなくなります。

- ファイル・システムをリストアするために、全クラスタ・メンバにアクセス可能な共用バス上のディスクで、ルート・ファイル・システムの問題発生前にはクラスタ構成に組み込まれていたディスクが利用できない場合

この手順は次のことを想定しています。

- `cluster_usr` および `cluster_var` ファイル・システムが、`cluster_root` と同じディスクにない。  
この手順では、`cluster_root` ファイル・システムの回復についてのみ説明します。
- クラスタ・ルート (`cluster_root`) ファイル・システムをバックアップするのに、コマンド `vdump` を使用した。  
別のバックアップ・ツールを使用した場合は、適切なツールを使用してファイル・システムをリストアしてください。
- 少なくとも 1 つのメンバは次のようなデバイスにアクセスできる。
  - クラスタにインストールされていた Tru64 UNIX とバージョンが同じブート可能な Tru64 UNIX ディスク  
ブート可能なベース・オペレーティング・システム用ディスクが使用できない場合は、クラスタ・メンバのローカル・ディスク上に Tru64 UNIX をインストールします。クラスタ上にインストールされた Tru64 UNIX と同じバージョンであることを確認してください。
  - 対象メンバ用のメンバ・ブート・ディスク (この例では `dsk2a`)

- クラスタ・ルート・バックアップ用デバイス
- 新しいクラスタ・ルート用ディスク (複数可)
- クラスタの全メンバが停止している。

クラスタ・ルートをリストアするには、次のように行います。

1. Tru64 UNIX ディスクでシステムをブートします。

この手順では、このシステムを member 1 と想定します。

2. 必要に応じて、ディスクにパーティションを設定し、ディスクがクラスタ・ルートとして、パーティション・サイズとファイル・システム・タイプが適切になるようにします。

3. 新しいクラスタ・ルート用の新規ドメインを作成します。

```
# mkfdmn /dev/disk/dsk5b new_root
```

TruCluster Server 『クラスタ・インストール・ガイド』で説明されているように、cluster\_root ファイル・システムは b パーティションによく置かれます。この場合、例として /dev/disk/dsk5b が使用されます。

4. そのドメインのルート・ファイルセットを作成します。

```
# mkfset new_root root
```

5. この復旧手順では、new\_root に 3 つまでのボリュームを使用できます。復旧が完了したら、新しいボリュームをクラスタ・ルートに追加することができます。この例の場合、1 つのボリューム (dsk8e) を追加します。

```
# addvol /dev/disk/dsk8e new_root
```

6. 新しいクラスタ・ルートになるドメインをマウントします。

```
# mount new_root#root /mnt
```

7. バックアップ・メディアからクラスタ・ルートをリストアします (vdump 以外のバックアップ・ツールを使用した場合は、vrestore の代わりに適切なリストア・ツールを使用します)。

```
# vrestore -xf /dev/tape/tape0 -D /mnt
```

8. リストアされたデータベースを、Tru64 UNIX システムの /etc ディレクトリにコピーします。

```
# cd /mnt/etc
```

```
# cp dec_unid_db dec_hwc_cdb dfsc.dat /etc
```

9. リストアされたデータベースの現在のメンバのメンバ固有データを Tru64 UNIX システムの /etc ディレクトリにコピーします。

```
# cd /mnt/cluster/members/member1/etc
# cp dfs1.dat /etc
```

10. メンバのブート・ディスクのドメインが存在しない場合には、ドメインを作成します。

```
# cd /etc/fdmns
# ls
# mkdir root1_domain
# cd root1_domain
# ln -s /dev/disk/dsk2a
```

11. メンバのブート・パーティションをマウントします。

```
# cd /
# umount /mnt
# mount root1_domain#root /mnt
```

12. メンバのブート・パーティションから Tru64 UNIX システムの /etc ディレクトリにデータベースをコピーします。

```
# cd /mnt/etc
# cp dec_devsw_db dec_hw_db dec_hwc_ldb dec_scsi_db /etc
```

13. メンバのブート・ディスクをアンマウントします。

```
# cd /
# umount /mnt
```

14. データベースの .bak バックアップ・ファイルを更新します。

```
# cd /etc
# for f in dec_*db ; do cp $f $f.bak ; done
```

15. 同じ Tru64 UNIX ディスクを使ってシステムをシングルユーザ・モードにリブートし、/etc にコピーされたデータベースを使うようにします。

クラスタ・ルート・ファイル・システムのバックアップが現在のディスク・ストレージ環境を反映していない場合、このプロシージャはこの時点でパニックを引き起こします。パニックが起これば、クラスタ・ファイル・システムを回復できなくなり、clu\_create コマンドを使用してクラスタを再作成しなければならなくなります。

16. シングルユーザ・モードにブートした後、バス上にあるデバイスをスキャンします。

```
# hwmgr -scan scsi
```

17. 書き込み可能としてルートを再マウントします。

```
# /sbin/mountroot
```

18. デバイス・データベースを検証し，更新します。

```
# dsfmgr -v -F
```

19. hwmgr を使用して現在のデバイス命名規則を確認します。

```
# hwmgr -view devices
```

20. 必要に応じて，ローカル・ドメインを更新し，デバイス命名規則を反映させます (特に，usr\_domain，new\_root，root1\_domain)。

この更新作業は，適切な /etc/fdmns ディレクトリへ移り，既存リンクを削除してから新たに現在のデバイス名へのリンクを作成して行います (現在のデバイス名は前の手順で調べました)。たとえば，次のように行います。

```
# cd /etc/fdmns/root_domain
# rm *
# ln -s /dev/disk/dsk1a
# cd /etc/fdmns/usr_domain
# rm *
# ln -s /dev/disk/dsk1g
# cd /etc/fdmns/root1_domain
# rm *
# ln -s /dev/disk/dsk2a
# cd /etc/fdmns/new_root
# rm *
# ln -s /dev/disk/dsk5b
# ln -s /dev/disk/dsk8e
```

21. コマンド bcheckrc を実行し，ローカル・ファイル・システムをマウントします。特に /usr のマウントが必要です。

```
# bcheckrc
```

22. 更新されたクラスタ・データベース・ファイルをクラスタ・ルート上にコピーします。

```
# mount new_root#root /mnt
# cd /etc
# cp dec_unid_db* dec_hwc_cdb* dfsc.dat /mnt/etc
# cp dfsl.dat /mnt/cluster/members/member1/etc
```

23. 新しいクラスタ・ルートで，cluster\_root ドメインを使用します。

```
# rm /mnt/etc/fdmns/cluster_root/*
# cd /etc/fdmns/new_root
# tar cf - * | (cd /mnt/etc/fdmns/cluster_root && tar xf -)
```

24. 更新されたクラスタ・データベース・ファイルをメンバのブート・ディスクにコピーします。

```
# umount /mnt
# mount root1_domain#root /mnt
# cd /etc
# cp dec_devsw_db* dec_hw_db* dec_hwc_ldb* dec_scsi_db* /mnt/etc
```

25. コマンド `file` を使用して、新しい `cluster_root` デバイスの主番号と副番号を調べます (`cluster_root` デバイスの主番号と副番号の使用についての詳しい情報は、11.1.6 項を参照してください)。この番号は次の手順で使用するので書き出しておきます。

たとえば、次のように行います。

```
# file /dev/disk/dsk5b
/dev/disk/dsk5b:          block special (19/227)
# file /dev/disk/dsk8e
/dev/disk/dsk8e:          block special (19/221)
```

26. システムを停止させ、新しいクラスタ・ルートの主番号と副番号を指定して対話形式でリブートします。ブート時のクラスタ・ルートの指定方法については、11.1.6 項で説明しています。

---

#### 注意

---

4.10.2 項で説明しているように、メンバをブートするには、おそらく、期待ポートを調整する必要があります。

---

```
>>> boot -fl "ia"
(boot dkb200.2.0.7.0 -flags ia)
block 0 of dkb200.2.0.7.0 is a valid boot block
reading 18 blocks from dkb200.2.0.7.0
bootstrap code read in
base = 200000, image_start = 0, image_bytes = 2400
initializing HWRPB at 2000
initializing page table at fff0000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

:

Enter kernel_name [option_1 ... option_n]
Press Return to boot default kernel
'vmunix':vmunix cfs:cluster_root_dev1_maj=19 \
cfs:cluster_root_dev1_min=227 cfs:cluster_root_dev2_maj=19 \
cfs:cluster_root_dev1_min=221 Return
```



27. その他のクラスタ・メンバをブートします。

ブート時にデバイス・ファイルでエラーが発生した場合は、コマンド `dsfmgr -v -F` を実行します。

### 11.1.9 AdvFS の問題の処理

ここでは、AdvFS を利用した場合に生じるおそれのある問題について説明します。

#### 11.1.9.1 addvol または rmvol からの警告メッセージに対する応答

ある条件下では、`cluster_root` ドメインで `addvol` または `rmvol` を使用すると、次の警告メッセージが出力される場合があります。

```
"WARNING:cfs_write_advfs_root_data: cnx_disk_write failed  
for quorum disk with error-number."
```

通常は、`error-number` は EIO の値です。

このメッセージは、`addvol` または `rmvol` が実行されたメンバが、クォーラム・ディスクの CNX パーティションに書き込みができないことを示しています。CNX パーティションには、`cluster_root` ドメインのデバイス情報が含まれています。

この警告が生じるのは、ブート・メンバがクォーラム・ディスクにアクセスできない場合で、その理由はクラスタが意図的にこのように構成されているか、またはバスの障害のどちらかです。前者の理由の場合、メッセージを情報とみなすことができます。後者の理由の場合は、バスの障害の原因に対して処置を講ずる必要があります。

メッセージは、クォーラム・ディスク自体に問題があることを示す場合があります。クォーラム・ディスクのハードウェア・エラーもレポートされている場合は、そのディスクを交換する必要があります。クォーラム・ディスクの交換については、4.5.1 項を参照してください。

エラー番号については、`errno(5)` を参照してください。EIO については、`errno(2)` を参照してください。

#### 11.1.9.2 デバイスの接続障害による AdvFS ドメイン・パニックの解消

ドメインまたはファイルセットの置かれているストレージが 1 つでも使用できなくなると、AdvFS でドメイン・パニックが発生するおそれがあります。この問題がもっともよく発生するケースは、AdvFS ドメインで使用され

ているプライベート・ストレージに接続しているクラスタ・メンバがクラスタを離れた場合です。次によくあるケースは、ストレージ・デバイスにハードウェア・トラブルが発生して使用できなくなった場合です。どちらの場合も、クラスタ・メンバからそのストレージへアクセスするパスがすべてなくなって、そのストレージが使用できなくなるため、ドメイン・パニックが発生します。

通常は、ドメイン・パニックが発生して最初に現れる症状として、デバイスから入出力エラーが返ったり、システム・コンソールにパニック・メッセージが出力されたりします。ドメインでは、まだ動作しているクラスタ・メンバによってサービスが行われていると、`cfsmgr -e` のような CFS コマンドに対して OK のステータスが返ることがあり、問題がただちに反映されるとは限りません。

```
# ls -l /mnt/mytst
/mnt/mytst: I/O error

# cfsmgr -e
Domain or filesystem name = mytest_dmn#mytst
Mounted On = /mnt/mytst
Server Name = deli
Server Status : OK
```

デバイスの接続を回復してサービスに使用できる場合は、`cfsmgr` コマンドを使用して、影響を受けたドメイン内のファイルセットを、パニックが生じる前にそのドメインへサービスを提供していたメンバに (または別のメンバに) 再配置することで、継続してドメインを使用することができます。

```
# cfsmgr -a SERVER=provolone -d mytest_dmn

# cfsmgr -e
Domain or filesystem name = mytest_dmn#mytests
Mounted On = /mnt/mytst
Server Name = provolone
Server Status : OK
```

### 11.1.9.3 AdvFS ファイル・システムまたはドメインの強制アンマウント

デバイスの接続性を回復してサービスに戻せない場合、TruCluster Server バージョン 5.1B の `cfsmgr -u` と `cfsmgr -U` コマンドを使用します。

クラスタ・メンバがサービスしていない AdvFS ファイル・システムやドメインは、`cfsmgr -u` を使用して、強制的にアンマウントできます。ファイル・システムまたはドメインがサービス中の場合は、アンマウントできません。

cfsmgr -U コマンドを使用して、クラスタ・メンバによってサービスされている AdvFS のドメイン全体を強制的にアンマウントできます。

上記のコマンドのどちらを使用するのは、その時点で CFS (クラスタ・ファイル・システム) がドメインをどう認識しているかによって異なります。

- cfsmgr -e コマンドを実行してドメインまたはファイル・システムの状態を調べ、サービス中でないことが分かれば、cfsmgr -u コマンドを使用してそのドメインまたはファイルを強制的にアンマウントします。

```
# cfsmgr -e
Domain or filesystem name = mytest_dmn#mytests
Mounted On = /mnt/mytst
Server Name = deli
Server Status : Not Served

# cfsmgr -u /mnt/mytst
```

ファイル・システム上でネストされたマウントがアンマウントされる場合、強制アンマウントは実行されません。同様に、ネストされたマウントがファイルセット上にある場合、全ドメインが強制的にアンマウントされるとき、およびネストされたマウントが同じドメインにないときは、強制アンマウントは実行されません。

- cfsmgr -e コマンドを実行してドメインまたはファイル・システムがサービス中であることが分かった場合は、cfsmgr -U コマンドを使用して、全ドメインを強制アンマウントできます。

```
# cfsmgr -e
Domain or filesystem name = mytest_dmn#mytests
Mounted On = /mnt/mytst
Server Name = deli
Server Status : OK

# cfsmgr -U /mnt/mytst
```

-u フラグとは違って、-U はドメイン全体に対してだけ用いられます。サービスされているドメイン中のファイル・システムのうち、1 つだけをアンマウントすることはできません。すなわち、1 つのファイル・システムを指定すると、ドメイン全体がアンマウントされます。アンマウントされるドメインのファイル・システム上にネストされたマウントがあり、同じドメイン上にネストされたマウントがない場合は、強制アンマウントは実行されません。

cfsmgr コマンドについての詳細は、cfsmgr(8) を参照してください。

#### 11.1.9.4 ドメイン・パニックの回避

AdvFS GUI (グラフィカル・ユーザ・インタフェース) エージェント `advfsd` は、定期的にシステム・ディスクを走査します。メタデータの書き込みエラーが発生した場合、または単一の AdvFS ファイル・ドメインで破損が検出された場合、`advfsd` デーモンは、システム・パニックではなく、ファイル・ドメイン上でドメイン・パニックを発生させます。この措置により、障害の発生したドメインが分離されるので、システムが引き続き他のすべてのドメインに対するサービスを提供できます。

クラスタのメンバ上で実行中の `advfsd` デーモンから見ると、AdvFS ドメインが含まれているディスクにアクセスできなくなったためにドメイン・パニックが生じる場合があります。正常な場合には、これは予定された動作です。このようなパニックを診断するには、Tru64 UNIX 『AdvFS 管理ガイド』のトラブルシューティングに関する章の手順に従ってください。ただし、クラスタ・メンバが、別のメンバのプライベート・ディスクが利用できなくなった (メンバの停止など) ためにドメイン・パニックを検出した場合は、ドメイン・パニックによって無用な混乱が生じます。

このようなメイン・パニックを回避するには、メンバの `/usr/var/advfs/daemon/disks.ignore` ファイルを編集して、AdvFS ドメインが含まれている他のメンバのプライベート・ストレージのディスク名を指定してください。これにより、ローカル・メンバ上の `advfsd` デーモンが、指定されたデバイスを走査しなくなります。

プライベート・デバイスを識別するには、`sms` コマンドを使用して SysMan Station のグラフィック・インタフェースを起動し、次に [Views] メニューから [Hardware] を選択してください。

#### 11.1.10 停止したシステム上のブート・パーティションへのアクセス

正常なシャットダウンによって、またはパニックなどの予定外の方法で、メンバがクラスタから外れる場合は、そのメンバのブート・パーティションはアンマウントされます。ブート・パーティションが共用バス上にある場合は、そのパーティションをマウントすれば、他のどのメンバでもブート・パーティションにアクセスできます。

システム `provolone` の停止中に `provolone` の `/etc/sysconfigtab` を編集するとします。その場合は、次のように入力できます。

```
# mkdir /mnt
# mount root2_domain#root /mnt
```

provolone をリブートする前に、root2\_domain#root をアンマウントしなければなりません。たとえば、次のように指定します。

```
# umount root2_domain#root
```

### 11.1.11 ブート・ディスクがすでにマウントされているメンバのブート

期待されるクォーラム・ポートまたはクォーラム・ディスク・デバイスが変更されると、各メンバの /etc/sysconfigtab ファイルが更新されます。クラスタ・メンバがダウンしている場合は、クォーラムに影響するクラスタ・ユーティリティ (clu\_add\_member, clu\_quorum, clu\_delete\_member など) が、ダウンしているメンバのブート・ディスクをマウントして更新します。ダウンしたメンバをブート・ディスクのマウント中にブートしようとする、次のようなパニック・メッセージが出力されます。

```
cfs_mountroot: CFS server already exists for this nodes boot partition
```

クラスタ・ユーティリティは、正常にダウンしているメンバのブート・ディスクをアンマウントします。

一般に、他のメンバがあるメンバのブート・ディスクをマウントしているときにそのメンバをブートしようすると、パニックが発生します。たとえば、ダウンしているメンバのブート・ディスクをマウントして修復する場合、修復できたメンバをブートする前にそのブート・ディスクをアンマウントしなければ、パニックが発生します。

### 11.1.12 クラッシュ・ダンプの生成

クラスタに重大な問題が生じた場合は、すべてのクラスタ・メンバからのクラッシュ・ダンプが必要になる可能性があります。動作中のメンバからクラッシュ・ダンプを取得するには、dumpsys コマンドを使用して、システム・メモリのスナップショットをダンプ・ファイルに保存してください。

クラッシュ・ダンプを生成するには、動作中の各クラスタ・メンバにログインして dumpsys を実行してください。省略時には、dumpsys を実行すると、メンバ固有のディレクトリ /var/adm/crash にダンプが書き込まれます。

詳細は、dumpsys(8) を参照してください。

SysMan Menu Configure Dump および Create Dump Snapshot を使用しても、クラッシュ・ダンプを構成できます。Configure Dump 機能は、`savecore` コマンドに関連する一般的なシステム構成変数に設定します。Create Dump Snapshot 機能は、`dumpsys` コマンドを使用します。このコマンドは、通常のクラッシュ・ダンプを生成するためにそのシステムを停止することができないときに、メモリのスナップショットを手動でファイルにダンプするために使用します。

#### 11.1.12.1 メンバがハングしたときのクラッシュ・ダンプの生成

クラスタ・メンバがハングした場合、クラッシュ・ダンプを生成するために `dumpsys` コマンドを使用することができません。この場合、クラッシュ・ダンプを生成するには次の手順を使用します。

1. `dumpsys` コマンドを使用して、動作中のメンバのメモリ・スナップショットをダンプ・ファイルにコピーします。省略時には、`dumpsys` コマンドは `/var/adm/crash` にダンプを書き込みます。このパスは、CDSL であり、実際のパスは `/cluster/members/{memb}/adm/crash` となります。
2. 5.5 節に説明されているように、`clu_quorum` コマンドを使用して、ハングしたメンバを停止したときにクォーラムを失っていないことを確認します。
3. ハングしたメンバをクラッシュさせます。これを行うには、メンバを手動で停止し、コンソールのプロンプトで `crash` を実行します。
4. メンバをブートします。ブートで `savecore (savecore(8))` を実行し、`/var/adm/crash` にダンプします。

#### 11.1.13 ネットワーク問題の修正

ここでは、クラスタ内の潜在的なネットワークの問題とその解決策について説明します。

##### 症状

- クラスタに対して `ping` を実行できない。
- クラスタへの、あるいはクラスタからの `rlogin` を実行できない。
- クラスタから `telnet` を実行できない。

## 確認事項

- すべてのクラスタ・メンバで `gated` が実行されていることを確認します。

さらに、`/etc/rc.config` に次の行が含まれていることを確認します。

```
GATED="yes"
export GATED
```

- `/etc/rc.config` に次の行が含まれていることを確認します。

```
ROUTER="yes"
export ROUTER
```

- `/etc/hosts` に省略時のクラスタ別名とクラスタ・メンバの正しいエントリが設定されていることを確認します。

`/etc/hosts` には、少なくとも次のエントリが設定されている必要があります。

- クラスタ別名の IP アドレスと名前

### 注意

クラスタ別名のアドレスでは、ブロードキャスト・アドレスやマルチキャスト・アドレスを使用できず、また、クラスタ・インターコネクトの使用するサブネット内のアドレスも使用できません。さらに、クラスタ・メンバは IPv6 アドレスを公開して使用できますが、クラスタ別名サブシステムはそのアドレスをサポートしていません。したがって、IPv6 アドレスをクラスタ別名に割り当てることはできません。

RFC 1918 で定義されているプライベート・アドレス・スペースの IP アドレスであれば、クラスタ別名に割り当てることができますが、別名サブシステムがその別名アドレスへのルートを公開できるようにするために、次のようなコマンドを実行する必要があります。

```
# rcmgr -c set CLUAMGR_ROUTE_ARGS resvok
# cluamgr -r resvok
```

各クラスタに対して `cluamgr` コマンドを繰り返します。  
`resvok` フラグについての詳細は、`cluamgr(8)` を参照してください。

---

- 各クラスタ・メンバの IP アドレスと名前
- 各メンバのクラスタ・インターコネクト・インタフェースに関連付けられた IP アドレスとインタフェース名

たとえば、次のようになります。

```
127.0.0.1      localhost
16.140.102.238 trees.tyron.com    trees
16.140.102.176 birch.tyron.com   birch
16.140.102.237 oak.tyron.com    oak
16.140.102.3   hickory.tyron.com hickory
10.0.0.1       birch-mc0
10.0.0.2       oak-mc0
10.0.0.3       hickory-mc0
```

- すべてのクラスタ・メンバ上で `aliasd` が実行していることを確認します。
- すべてのクラスタ・メンバが省略時の別名のメンバになっている (`joined` と `enabled`) ことを確認します。これを確認するには、次のコマンドを入力します。

```
# cluamgr -s default_alias
```

あるメンバを省略時の別名のメンバにするには、そのメンバ上で `cluamgr` コマンドを実行します。たとえば、次のように指定します。

```
# cluamgr -a alias=default_alias,join
```

次に、各メンバ上で次のコマンドを実行します。

```
# cluamgr -r start
```

- 1つのメンバが、省略時の別名に対してルーティングしていることを確認します。これを確認するには、各メンバ上で次のコマンドを実行します。

```
# arp default_alias
```

実行結果には、`permanent published` というフレーズが含まれるはずですが、1つのメンバにおいて、省略時のクラスタ別名に向けた静的発行経路が設定されている必要があります。



- クラスタ別名の IP アドレスが別のシステムで使用されていないことを確認します。

クラスタ別名デーモン `aliasd` を誤って別のシステムで使用中の別名 IP アドレスで構成した場合は、クラスタで接続障害が発生するおそれがあります。たとえば、一部のマシンはクラスタ別名に到達できますが、その他のマシンは到達できない場合があります。別名に到達できないマシンは、まったく別のマシンに接続する可能性があります。

クラスタ外のシステム上の `arp` キャッシュを検査すれば、影響を受ける別名 IP アドレスが 2 つ以上の異なるハードウェア・アドレスにマップされていることが明白になる可能性があります。

クラスタが重大度 `err` のメッセージを記録するように構成されている場合は、システム・コンソールとカーネル・ログ・ファイルで次のメッセージを調べます。

```
local IP address nnn.nnn.nnn.nnn in use by hardware
address xx-xx-xx-xx-xx
```

`/etc/rc.config` と `/etc/hosts` 内のエントリが正しく、他の問題がすべて修正されたことを確認した後は、`gateway` と `inet` のデーモンを停止してから再起動してください。この操作を行うには、各クラスタ・メンバー上で次のコマンドを入力します。

```
# /sbin/init.d/gateway stop
# /sbin/init.d/gateway start
```

### 11.1.14 クラスタでの `routed` の実行

TruCluster Server バージョン 5.1B より前のリリースでは、ルーティング・デーモンとして `gated` をクラスタで実行する必要がありました。`routed`、`ogated`、または静的ルーティングのセットアップを使用できず、また、ルーティング・デーモンも使用できませんでした。バージョン 5.1B からは、`gated`、`routed`、または静的ルーティングが使用可能になりました。`ogated` は使用できません。省略時の構成では、`gated` となります。3.14 節でルーティング・オプションについて説明しています。

## 11.2 クラスタ管理のヒント

ここでは、クラスタの構成と管理のヒントおよび提案について説明します。

### 11.2.1 /tmp の移動

省略時には、メンバ固有の /tmp 領域は同じファイル・システム内にありますが、その領域は個々のファイル・システムに移動できます。場合によっては、共用 SCSI バスのトラフィックを軽減するために、各メンバの /tmp 領域をメンバのローカル・ディスクに移動してもかまいません。

クラスタ・メンバに対してプライベート・バス上の固有の /tmp ディレクトリを設定したい場合は、そのクラスタ・メンバのローカル・バス上のディスクに AdvFS ドメインを作成して、そのドメインの /etc/fstab に /tmp のマウント・ポイントを持つエントリを追加できます。

たとえば、次の /etc/fstab のエントリは、メンバ ID がそれぞれ 58 と 59 の 2 つのクラスタ・メンバ tcr58 と tcr59 の /tmp ディレクトリに対応しています。

```
tcr58_tmp#tmp    /cluster/members/member58/tmp    advfs rw 0 0
tcr59_tmp#tmp    /cluster/members/member59/tmp    advfs rw 0 0
```

tcr58\_tmp ドメインは、tcr58 メンバのみが接続できるバス上にあります。tcr59\_tmp ドメインは、tcr59 メンバのみが接続できるディスク上にあります。

各メンバは、ブート時に /etc/fstab 内のすべてのファイル・システムをマウントしようとしませんが、マウントできるのは、まだマウントされていない、デバイスへのパスが存在するドメインのみです。この例では、tcr58 は tcr58\_tmp#tmp のみをマウントできます。また、tcr59 は tcr59\_tmp#tmp のみをマウントできます。

/etc/fstab に次のように設定できます。

```
tcr58_tmp#tmp    /tmp    advfs rw 0 0
tcr59_tmp#tmp    /tmp    advfs rw 0 0
```

/tmp は CDSL (コンテキスト依存シンボリック・リンク) なので、/cluster/members/membern/tmp に解決されます。ただし、/etc/fstab に完全パス名を設定すれば、より明らかになるので混乱が生じる確率は低くなります。

### 11.2.2 MC\_CABLE コンソール・コマンドの実行

任意のメンバ上で MC\_CABLE Memory Channel 診断コマンドを実行する前に、すべてのメンバをコンソール・プロンプトに応じてシャットダウンしなければなりません。これは正常な操作です。

停止しているクラスタ・メンバのコンソールから、他のメンバの起動時に MC\_CABLE コマンドを実行すると、クラスタがクラッシュします。

### 11.2.3 Korn シェルにおけるメンバ固有のディレクトリへの実パスの非表示

Korn シェル (ksh) では、ディレクトリまでのパスが記憶され、pwd コマンドを入力すると、そのパス名が表示されます。これは、パス内のシンボリック・リンクのために別のディレクトリに移動している場合でも同じです。TruCluster Server では、CDSL を利用してメンバ固有のディレクトリがクラスタ単位のネームスペースに保持されるので、作業ディレクトリが CDSL になっている場合は、Korn シェルでは実パスは返されません。

パス名の表示時にシェルでシンボリック・リンクを解釈させたい場合は、Korn シェル以外のシェルを使用してください。たとえば、次のようになります。

```
# ksh
# ls -l /var/adm/syslog
lrwxrwxrwx  1 root system  36 Nov 11 16:17 /var/adm/syslog
->../cluster/members/{memb}/adm/syslog
# cd /var/adm/syslog
# pwd
/var/adm/syslog
# sh
# pwd
/var/cluster/members/member1/adm/syslog
```



---

## クラスタ・イベント

クラスタ・イベントは、個々のメンバではなくクラスタとしてポストされる EVM (イベント・マネージャ) イベントです。

全クラスタ・イベントのリストを表示させるには、次のようなコマンドを使用します。

```
# evmwatch -i | evmshow -t "@name @cluster_event" | \
grep True$ | awk '{print $1}'
```

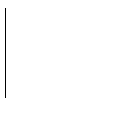
EVM 優先順位とイベントの説明を表示させるには、次のようなコマンドを使用します。

```
# evmwatch -i -f '[name event_name]' | \
evmshow -t "@name @priority" -x
```

たとえば次のように表示されます。

```
# evmwatch -i -f '[name sys.unix.clu.cfs.fs.served]' | \
evmshow -t "@name @priority" -x sys.unix.clu.cfs.fs.served 200
    This event is posted by the cluster filesystem (CFS) to
    indicate that a filesystem has been mounted in the cluster,
    or that a filesystem for which this node is the server has
    been relocated or failed over.
```

EVM 優先順位については、[EvmEvent\(5\)](#) を参照してください。イベント管理についての詳細は、[EVM\(5\)](#) を参照してください。



# B

## 構成変数

表 B-1 は、メンバ固有の `rc.config` ファイルに登録できるクラスタ構成変数の一部を示しています。

`rc.config` または `rc.config.common` を変更した後に、その変更を有効にするには、各メンバを個々にリブートしてください。

`rc.config` についての詳しい説明は、5.1 節を参照してください。

表 B-1: クラスタ構成変数

変数	説明
CLU_BOOT_FILESYSTEM	このメンバのブート・ディスクのドメインとファイルセットを指定する。
CLU_NEW_MEMBER	今回がこのメンバの最初のブートかどうかを指定する。値が 1 の場合は、初回のブートを示す。値が 0 (ゼロ) の場合は、メンバが以前にブートしていることを示す。
CLU_VERSION	クラスタにインストールされている TruCluster Server ソフトウェアのバージョンを指定する。
CLUSTER_NET	システムの 1 次ネットワーク・インタフェースの名前を指定する。
IMC_AUTO_INIT	この変数に 1 が設定されている場合は、Memory Channel API ライブラリがブート時に自動的に初期化される。この初期化には、Memory Channel API ライブラリ用に約 4.5 MB を確保する必要がある。IMC_AUTO_INIT の省略時の値は 0。
IMC_MAX_ALLOC	クラスタ全体で Memory Channel API ライブラリ用に割り当て可能な Memory Channel アドレス空間の総容量の最大値を確定する。この変数の値がクラスタ・メンバ間で異なる場合は、任意のメンバに対して指定された最大値で、クラスタに設定される値が確定される。省略時のアドレス空間の容量は 10 MB。
IMC_MAX_RECV	Memory Channel API ライブラリで Memory Channel アドレス空間を読み込む場合にマップできる物理メモリの最大容量を確定する。この制限はノード固有で、メンバによって異なる場合がある。省略時のアドレス空間の容量は 10 MB。

表 B-1: クラスタ構成変数 (続き)

変数	説明
TCR_INSTALL	TCR に等しい場合は、正常にインストールされたことを示す。BAD に等しい場合は、正常にインストールされていないことを示す。
TCR_PACKAGE	TCR に等しい場合は、正常にインストールされたことを示す。



# C

## clu\_delete\_member ログ

clu\_delete\_member を実行するたびに , /cluster/admin/clu\_delete\_member.log にログ・メッセージが書き込まれます。次に示すのは , サンプルの clu\_delete\_member ログ・ファイルです。

This is the TruCluster Delete Member Program

You will need the following information in order to delete a member from the cluster:

- Member ID (1-63)

The program will prompt for this information, offering a default value when one is available. To accept the default value, press Return. If you need help responding to a prompt, either type the word 'help' or type a question mark (?) at the prompt.

The program does not begin to delete the member until you answer all the prompts, and confirm that the answers are correct.

Deleting a member involves the following steps:

- Removing the files from the member boot disk. (If accessible)
- Removing member specific areas from the /, /usr, and /var file systems.
- Removing the deleted members entries in shared configuration files.

Do you want to continue deleting a member from this cluster? [yes]: Return

A member ID is used to identify each member in a cluster. Each member must have a unique member ID, which is an integer in the range 1-63, inclusive.

Enter a cluster member ID []: 2  
Checking cluster member ID: 2

You entered '2' as the member ID.  
Is this correct? [yes]: Return

You entered the following information:

- Member's ID: 2

If any of this information is incorrect, answer 'n' to the following prompt. You can then enter the correct information.

Do you want to continue to delete a cluster member?: [no] y Return

Deleting member disk boot partition files  
Member disk boot partition files deleted

Initial cluster deletion successful, member '2' can no longer join the cluster. Deletion continuing with cleanup  
cluster\_expected\_votes: reconfigured

```
Removing deleted member entries from shared configuration files
  Removing cluster interconnect interface 'polishham-mc0' from /.rhosts
  :
Deleting Member Specific Directories
  Deleting: /cluster/members/member2/
  Deleting: /usr/cluster/members/member2/
  Deleting: /var/cluster/members/member2/

clu_delete_member: The deletion of cluster member '2' completed successfully.
```

# D

## LAN クラスタの管理

この付録では、以下の項目について説明します。

- クラスタ・インターコネクต์に使用する NetRAIN 仮想インタフェースの構成 (D.1 節)
- LAN インターコネクットの性能を最適化するためのチューニング (D.2 節)
- ネットワーク・アダプタ構成情報の取得 (D.3 節)
- LAN インターコネクต์上の動作の監視 (D.4 節)
- Memory Channel インターコネクต์から LAN インターコネクต์への移行 (D.5 節)
- LAN インターコネクต์から Memory Channel インターコネクต์への移行 (D.6 節)
- 高速イーサネット LAN からギガビット・イーサネット LAN への移行 (D.7 節)
- LAN インターコネクต์で起こる問題のトラブルシューティング (D.8 節)

### D.1 クラスタ LAN インターコネクต์に使用する NetRAIN 仮想インタフェースの構成

クラスタをインストールするときに NetRAIN (redundant array of independent network adapters) 仮想インタフェースからクラスタ・インターコネクットを構成しなくても、後から構成することができます。ただし、クラスタ・インターコネクットで NetRAIN 仮想インタフェースを構成するための要件と規則は、Tru64 UNIX 『ネットワーク管理ガイド：接続編』で説明されているものと異なります。

通常の NetRAIN 仮想デバイスとは異なり、クラスタ・インターコネクットに使用する NetRAIN デバイスは、`/etc/rc.config` ではなく `/etc/sysconfigtab` にある `ics_ll_tcp` カーネル・サブシステムの中に設定します。そうすれば、ブートの非常に早い段階でインターコネクットを確

立できるので、クラスタ構成要素をメンバとして早く組み込んで入出力を始めたいという要求に応えることができます。

---

#### 警告

---

メンバのクラスタ・インターコネクト用 NetRAIN デバイスの属性は、`/etc/sysconfigtab` ファイル以外で変更しないでください。つまり、`ifconfig` コマンドまたは SysMan Station を使って変更したり、`/etc/rc.config` ファイルの中の定義を追加または変更してネットワークを再起動したりしないでください。そうすると、クラスタで NetRAIN デバイスを制御できなくなり、メンバ・システムがクラスタから削除されてしまうことがあります。

---

クラスタをインストールした後でクラスタ・インターコネクト用に NetRAIN インタフェースを構成する場合は、各メンバで以下の手順を実行してください。

1. クラスタ・インターコネクトでは、LAN インターコネクトに障害が起きてもシステム全体がダウンしないようにする必要があります。そのため、NetRAIN セットとして構成するメンバの冗長イーサネット・アダプタとは別に、イーサネット・スイッチが必要です (LAN インターコネクトを NSPOF (No Single Point of Failure) 構成にする場合は 2 つ必要)。ネットワーク・ハードウェアを追加してインストールする必要がある場合は、そのメンバ・システムを停止し、電源を切ってください。その状態で、そのメンバにネットワーク・カードをインストールし、そのカードを、『クラスタ・ハードウェア構成ガイド』で推奨しているように、異なるスイッチへそれぞれケーブル接続します。その後、スイッチの電源を入れ、メンバをリブートします。追加ハードウェアをインストールする必要がない場合は、この手順を省略します。
2. `ifconfig -a` コマンドを使って、NetRAIN セットに使うイーサネット・アダプタの名前を調べます。
3. 既存の NetRAIN セットをクラスタ・インターコネクトに構成する場合は、最初に、次のようにして現在の構成を無効にする必要があります。
  - a. `rcmgr delete` コマンドを使って、メンバの `/etc/rc.config` ファイルからデバイスに関係する以下の変数を削除します。  
`NRDEV_X NRCONFIG_X NETDEV_X IFCONFIG_X`

- b. `rcmgr set` コマンドを使って、`NR_DEVICES` 変数と `NUM_NETCONFIG` 変数の値を減らします。

4. `/etc/sysconfigtab` ファイルを編集して、新しいアダプタを追加します。たとえば、次のように変更します。

```
ics_ll_tcp:

ics_tcp_adapter0 = alt0

を

ics_ll_tcp:

ics_tcp_adapter0 = nr0
ics_tcp_nr0[0] = alt0
ics_tcp_nr0[1] = alt1
```

のように変更。

5. メンバをリブートして、メンバが NetRAIN 仮想インタフェースを物理 クラスタ・インターコネクトとして使えるようにします。
6. たとえば、次のように `ifconfig` コマンドを使って、`CLUIF` フラグで定義されている NetRAIN デバイスを表示させます。

```
# ifconfig nr0
nr0: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
    NetRAIN Attached Interfaces: ( alt0 alt1 ) Active Interface: ( alt0 )
    inet 10.1.0.2 netmask ffffffff broadcast 10.1.0.255 ipmtu 1500
```

7. この手順を、残りのメンバについて繰り返します。

## D.2 LAN インターコネクトのチューニング

この節では、LAN インターコネクトのチューニングについて、そのガイドラインを示します。

### 警告

クラスタ・インターコネクトで使っている NetRAIN 仮想インタフェースのチューニングは、他の NetRAIN デバイスに対して使用する方法で行わないでください。使用してはならない方法には、`ifconfig`、`niffconfig`、`niffd` の各コマンド・オプションや `netrain`、または `ics_ll_tcp` カーネル・サブシステム属性も含まれます。この制約に従わなければ、クラスタの運用が中断する可能性があります。クラスタ・インターコネクトで使われている

NetRAIN デバイスは、クラスタ・ソフトウェアによって、クラスタの運用が最適になるようにすでにチューニングされています。

### D.2.1 ipmtu 値の設定によるクラスタ・インターコネクトの性能改善

アプリケーションによっては、各メンバのクラスタ・インターコネクト仮想インタフェース (ics0) で使用している IP の MTU (最大伝送単位) を物理インタフェース (member $n$ -tcp0) で使用している値と同じ値に設定することにより、性能を改善できるものがあります。推奨値は、使用しているクラスタ・インターコネクトの種類によって異なります。

- 高速イーサネットまたはギガビット・イーサネットの場合は、ipmtu の値を 1500 に設定します。

#### 注意

LAN インターコネクトは、ギガビット・イーサネットによる大きな MTU サイズ (ジャンボ・フレーム) の性能特性を得るようには構成できません。

- Memory Channel の場合は、ipmtu の値を 7000 に設定します。

仮想および物理クラスタ・インターコネクト・デバイスに現在設定されている ipmtu の値を調べる場合は、次のコマンドを使います。

```
# ifconfig -a
ee0: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
    inet 10.1.0.100 netmask ffffffff00 broadcast 10.1.0.255 ipmtu 1500

ics0: flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
    inet 10.0.0.1 netmask ffffffff00 broadcast 10.0.0.255 ipmtu 7000
```

このクラスタ・メンバは、物理クラスタ・インターコネクト・デバイスに ee0 イーサネット・デバイスを使っているので、仮想クラスタ・インターコネクト・デバイス (ics0) の ipmtu を 7000 から 1500 に変更します。

ics0 仮想デバイスの ipmtu 値を変更する場合は、以下の手順を実行します。

1. ipmtu の値が設定されている各メンバの /etc/inet.local ファイルに、次の行を追加します。

```
ifconfig ics0 ipmtu value
```

2. `rcinet restart` コマンドを使って、各メンバでネットワークを再起動させます。

### D.2.2 イーサネット・スイッチ・アドレス・エイジングに 15 秒を設定する

イーサネット・スイッチは、MAC アドレス (および仮想 LAN (VLAN) 識別子) をポートに関連付けるテーブルを保持しており、それによって、スイッチはパケットを効率的に転送できるようになります。これらの転送データベース (ユニキャスト・アドレス・テーブルとも呼ばれる) では、動的に取得した転送情報が古くなって無効になる時間間隔を設定するメカニズムを提供します。このメカニズムは、エイジング・タイムとも呼ばれます。

LAN インターコネクトに関連するすべてのイーサネット・スイッチに対して、エイジング・タイムは 15 秒に設定します。

転送情報の更新に失敗すると、スイッチは、(たとえば、NetRAIN のフェイルオーバーによって) MAC アドレスが別のポートへ移動した後、転送テーブル内にリストされているポートへの MAC アドレスにパケットを誤って転送し続けることになります。これは、クラスタの通信を混乱させ、ノードのいくつかがクラスタから削除されたのと同じ結果になります。このため、クォーラムを喪失して、1 つ以上のノードがハングします。また、いくつかのパニック・メッセージが発生します。例を次に示します。

```
CNX MGR: this node removed from cluster
CNX QDISK: Yielding to foreign owner
```

## D.3 ネットワーク・アダプタ構成情報の取得

ネットワーク・アダプタ用データリンク・ドライバから名前、速度、動作モードといった情報を取得して表示させる場合は、SysMan Station または `hwmgr -get attr -cat network` コマンドを使います。次に示すのはその例で、`tu2` が 10 Mb/秒の半二重モードで動作しているクライアント・ネットワーク・アダプタであることと、`ee0` と `ee1` が LAN インターコネクトとして構成されている NetRAIN 仮想インタフェースであって、100 Mb/秒の全二重モードで動作していることが分かります。

```
# hwmgr -get attr -cat network | grep -E 'name|speed|duplex'
# hwmgr -get attr -cat network | grep -E 'name|speed|duplex'
name = ee0
media_speed = 100
full_duplex = 1
```

```

user_name = (null) (settable)
name = ee1
media_speed = 100
full_duplex = 1
user_name = (null) (settable)
name = tu0
media_speed = 10
full_duplex = 1
user_name = (null) (settable)
name = tu1
media_speed = 10
full_duplex = 0
user_name = (null) (settable)
name = tu2
media_speed = 10
full_duplex = 0
user_name = (null) (settable)
name = tu3
media_speed = 10
full_duplex = 0
user_name = (null) (settable)
name = alt0
media_speed = 1000
full_duplex = 1
user_name = (null) (settable)

```

## D.4 LAN インターコネクト上の動作の監視

LAN インターコネクト上のトラフィックの監視には `netstat` コマンドを使います。次にその例を示します。

```

# netstat -acdnots -I nr0
nr0 Ethernet counters at Mon Apr 30 14:15:15 2001

    65535 seconds since last zeroed
3408205675 bytes received
4050893586 bytes sent
  7013551 data blocks received
  6926304 data blocks sent
  7578066 multicast bytes received
  115546 multicast blocks received
  3182180 multicast bytes sent
   51014 multicast blocks sent
     0 blocks sent, initially deferred
     0 blocks sent, single collision
     0 blocks sent, multiple collisions
     0 send failures
     0 collision detect check failure
     0 receive failures
     0 unrecognized frame destination
     0 data overruns
     0 system buffer unavailable
     0 user buffer unavailable
nr0: access filter is disabled

```

### D-6 LAN クラスタの管理



NetRAIN 仮想インタフェースのステータスがアクティブまたは非アクティブのどちらであるかを監視する場合は、次の例に示すように、`ifconfig -a` コマンドと `niffconfig -v` コマンドを使います。

```
# ifconfig -a
ee0: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
    NetRAIN Virtual Interface: nr0
    NetRAIN Attached Interfaces: ( ee1 ee0 ) Active Interface: ( ee1 )

ee1: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
    NetRAIN Virtual Interface: nr0
    NetRAIN Attached Interfaces: ( ee1 ee0 ) Active Interface: ( ee1 )

ics0: flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
    inet 10.0.0.200 netmask ffffffff broadcast 10.0.0.255 ipmtu 15u00

lo0: flags=100c89<UP,LOOPBACK,NOARP,MULTICAST,SIMPLEX,NOCHECKSUM>
    inet 127.0.0.1 netmask ff000000 ipmtu 4096

nr0: flags=1000c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX,CLUIF>
    NetRAIN Attached Interfaces: ( ee1 ee0 ) Active Interface: ( ee1 )
    inet 10.1.0.2 netmask ffffffff broadcast 10.1.0.255 ipmtu 1500

sl0: flags=10<POINTOPOINT>

tu0: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX>
    inet 16.140.112.176 netmask ffffffff broadcast 16.140.112.255 ipmtu 1500

tun0: flags=80<NOARP>

# niffconfig -v
Interface: ee1, description: NetRAIN internal, status: UP, event: ALERT, state: GREEN
    t1: 3, dt: 2, t2: 10, time to dead: 3, current_interval: 3, next time: 1
Interface: nr0, description: NetRAIN internal, status: UP, event: ALERT, state: GREEN
    t1: 3, dt: 2, t2: 10, time to dead: 3, current_interval: 3, next time: 1
Interface: ee0, description: NetRAIN internal, status: UP, event: ALERT, state: GREEN
    t1: 3, dt: 2, t2: 10, time to dead: 3, current_interval: 3, next time: 2
Interface: tu0, description: , status: UP, event: ALERT, state: GREEN
    t1: 20, dt: 5, t2: 60, time to dead: 30, current_interval: 20, next time: 20
```

## D.5 Memory Channel から LAN への移行

この節では、クラスタ・インターコネクトとして Memory Channel インターコネクトを使っているクラスタを LAN インターコネクトへ移行する方法を説明します。

Memory Channel インターコネクトを LAN インターコネクトで置き換えるときは、クラスタをしばらくダウンさせる必要があるので、サービスが一時中断します。

---

### 注意

『クラスタ・インストレーション・ガイド』の説明に従ってローリング・アップグレードする際に、Memory Channel を LAN インターコネクトで置き換える作業も一緒に行う場合は、各メンバで

ロールするときに LAN ハードウェアをインストールするように計画してください。そうすれば、以下に示す手順のうち、1~4 を省略することができます。

---

Memory Channel インターコネクトを使っている既存のクラスタを LAN インターコネクトを使うように移行する場合は、準備段階として、それぞれのクラスタ・メンバで以下の手順を実行してください。

1. クラスタ・メンバを停止し、電源を切ります。
2. ネットワーク・アダプタをインストールし、必要なスイッチまたはハブを構成します。
3. クラスタ・メンバの電源を入れます。
4. Memory Channel を使って、`genvmunix` カーネルでメンバをブートして、クラスタへ組み込みます。
5. `doconfig` を使用して `vmunix` カーネルを再構築します。
6. ルート (/) ディレクトリに新しいカーネルをコピーします。

新しくインストールしたイーサネット・ハードウェアは、この時点で、すべてのクラスタ・メンバで共用するプライベートな従来型サブネットとして構成できます。LAN インターコネクトをセットアップする前に、ハードウェアが適切に構成されていて正しく動作することを確認してください。`rcmgr` コマンドを使用したり `/etc/rc.config` ファイルを静的に編集したりして、このネットワークを永続的に設定してはなりません。この試験用ネットワークは LAN インターコネクト上でクラスタをリブートしたときに残っていないようにする必要があります。したがって、各メンバで設定を行うときは、`ifconfig` コマンドを使ってください。

LAN インターコネクトを構成する場合は、以下の手順を実行してください。

---

#### 注意

---

クラスタから Memory Channel ハードウェア (特にハブ) を削除しようとする場合、次の手順の最初からの 7 ステップを実行します。すべてのメンバを停止した後で、各メンバの電源を切ります。そうすると、Memory Channel のハードウェアを削除できま

す。手順 8 で説明されているとおり，すべてのメンバの電源を入れて，1 つずつブートします。

実行中のクラスタで Memory Channel ハブをシャット・オフすると，LAN インターコネクトを使用しているのがたった 1 つであっても，クラスタ全体でパニックを引き起こします。ハブからメンバの Memory Channel ケーブルを切断すると，そのメンバにパニックを引き起こします。

1. 各メンバで，メンバ固有の `/etc/sysconfigtab` ファイルと `/etc/rc.config` ファイルについてバックアップ・コピーを作成します。
2. 各メンバで，メンバ固有の `/etc/rc.config` ファイルを検査します。特に，`NETDEV_x` 構成変数と `NRDEV_x` 構成変数に注意してください。LAN インターコネクトで使うネットワーク・アダプタは，ブート手順の非常に早い段階で構成されるので，`/etc/rc.config` ではなく `/etc/sysconfigtab` (次の手順を参照) で定義してください。これは NetRAIN デバイスにも当てはまります。LAN インターコネクトとして，新しいデバイスを構成するのか以前のデバイスを再構成するのかを決めます。以前のデバイスを再構成する場合は，`NRDEV_x`，`NRCONFIG_x`，`NETDEV_x`，`IFCONFIG_x`，`NR_DEVICES`，および `NUM_NETCONFIG` の各変数を適切に編集して，`/etc/sysconfigtab` ファイルの `ics_ll_tcp` スタンザと `/etc/rc.config` ファイルに同じネットワーク・デバイス名が現れないようにしてください。
3. 各メンバで，`clubase` カーネル属性の `cluster_interconnect` に `tcp` を設定し，以下の `ics_ll_tcp` カーネル属性を，メンバのネットワーク構成に合わせて設定します。たとえば，次のようにします。

```
clubase:
cluster_interconnect = tcp
#
ics_ll_tcp:
ics_tcp_adapter0 = nr0
ics_tcp_nr0[0] = ee0
ics_tcp_nr0[1] = ee1
ics_tcp_inetaddr0 = 10.1.0.1
ics_tcp_netmask0 = 255.255.255.0
```

TruCluster Server バージョン 5.1 からロールしたクラスタでも、clubase カーネル・サブシステムの cluster\_node\_inter\_name 属性を編集します。たとえば、次のように編集します。

```
clubase:
cluster_node_inter_name = pepicelli-ics0
```

4. クラスタ単位の /etc/hosts ファイルを編集して、クラスタ・インターコネクトの下位レベル TCP インタフェースにおける IP 名と IP アドレスを設定します。たとえば、次のように設定します。

```
127.0.0.1          localhost
16.140.112.238     pepicelli.zk3.dec.com      pepicelli
16.120.112.209     deli.zk3.dec.com          deli
10.0.0.1           pepicelli-ics0
10.1.0.1           member1-icstcp0
10.0.0.2           pepperoni-ics0
10.1.0.2           member2-icstcp0
16.140.112.176     pepperoni.zk3.dec.com      pepperoni
```

5. TruCluster Server バージョン 5.1 からロールしたクラスタでは、クラスタ単位の /etc/hosts.equiv ファイルと /.rhosts ファイルを編集して、mc0 エントリを ics0 エントリに変更します。たとえば、次のように変更します。

```
deli.zk3.dec.com
pepicelli-mc0
pepperoni-mc0
```

を

```
deli.zk3.dec.com
pepicelli-ics0
member1-icstcp0
pepperoni-ics0
member2-icstcp0
```

に変更。

6. TruCluster Server バージョン 5.1 からロールしたクラスタでは、rcmgr set コマンドを使って、各メンバの /etc/rc.config ファイルにある CLUSTER\_NET 変数を変更します。たとえば、次のようにします。

```
# rcmgr get CLUSTER_NET
pepicelli-mc0
# rcmgr set CLUSTER_NET pepicelli-ics0
```

7. すべてのクラスタ・メンバを停止させます。
8. すべてのクラスタ・メンバを一度に 1 つずつブートします。

## D.6 LAN から Memory Channel への移行

この節では、クラスタ・インターコネクต์に LAN インターコネクต์を使っているクラスタを Memory Channel へ移行する方法を説明します。

Memory Channel を構成する場合は、以下の手順を実行してください。

1. すべてのメンバの電源を切ります。
2. 『クラスタ・ハードウェア構成ガイド』に説明されているように、Memory Channel アダプタ、ケーブル、およびハブをインストールして構成します。
3. LAN インターコネクต์を使用して、`genvmunix` カーネルで各メンバをリブートし、クラスタへ組み込みます。
4. `doconfig` を使用して、各メンバの `vmunix` カーネルを再構築します。
5. メンバのルート (/) ディレクトリに各メンバの新しいカーネルをコピーします。
6. 各メンバで、メンバ固有の `/etc/sysconfigtab` ファイルについて、そのバックアップ・コピーを作成します。
7. 各メンバで、`clubase` カーネル属性の `cluster_interconnect` を `mct` に設定します。
8. すべてのクラスタ・メンバを停止させます。
9. すべてのクラスタ・メンバを一度に 1 つずつリブートします。

## D.7 高速イーサネット LAN からギガビット・イーサネット LAN への移行

この節では、高速イーサネット LAN インターコネクต์を使うクラスタを、ギガビット・イーサネット LAN インターコネクต์を使うようにするための移行方法について説明します。

高速イーサネット LAN インターコネクต์をギガビット・イーサネット LAN インターコネクต์で置き換えるときは、クラスタをしばらくダウンさせる必要があるので、サービスが一時中断します。

高速イーサネット LAN インターコネクトを使用する既存のクラスタを、ギガビット・イーサネット LAN インターコネクトへ移行する準備作業は、それぞれのクラスタで以下の手順で行います。

1. クラスタ・メンバを停止し、電源を切ります。
2. ギガビット・イーサネット・ネットワーク・アダプタをインストールし、必要なスイッチ、またはハブを構成します。
3. クラスタ・メンバの電源を入れます。
4. 高速イーサネット LAN インターコネクトを使用し、`genvmunix` カーネルでメンバをリブートして、クラスタへ組み込みます。
5. `doconfig` コマンドを使用して、メンバの `vmunix` カーネルを再構築します。
6. メンバのルート (`/`) ディレクトリに新しいカーネルをコピーします。

新しくインストールしたイーサネット・ハードウェアは、この時点で、すべてのクラスタ・メンバで共用するプライベートな従来型サブネットとして構成できます。LAN インターコネクトとしてセットアップする前に、ハードウェアが適切に構成されていて正しく動作することを検証できます。`rcmgr` コマンドを使用したり、`/etc/rc.config` ファイルを手動で編集したりして、永続的なネットワーク設定にしてはなりません。この試験用ネットワークは LAN インターコネクトを使用してクラスタをリブートしたときに存在していないようにする必要があります。したがって、各メンバで設定を行うときは、`ifconfig` コマンドを使用します。

ギガビット・イーサネット LAN インターコネクトを構成するには、次の手順で行います。

1. それぞれのメンバで、メンバ固有の `/etc/sysconfigtab` ファイルのバックアップ・コピーを作ります。
2. それぞれのメンバで、メンバ固有の `/etc/rc.config` ファイルを調べます。特に、`NETDEV_x` 構成変数と `NRDEV_x` 構成変数に注意します。LAN インターコネクトで使うネットワーク・アダプタは、ブート手順の非常に早い段階で構成されるので、`/etc/sysconfigtab` (次の手順を参照) で定義されます。`/etc/rc.config` で定義してはなりません。これは NetRAIN デバイスにも当てはまります。LAN インターコネクト用に新しいデバイスを構成するのか以前のデバイスを再構成するのかを決め

ます。以前のデバイスを再構成する場合は、`NRDEV_X`、`NRCONFIG_X`、`NETDEV_X`、`IFCONFIG_X`、`NR_DEVICES`、および `NUM_NETCONFIG` の各変数を適切に編集して、`/etc/sysconfigtab` ファイルの `ics_ll_tcp` スタンザと `/etc/rc.config` ファイルに同じネットワーク・デバイス名が現れないようにします。

3. それぞれのメンバで、`ics_ll_tcp` カーネル属性をメンバのネットワーク構成に合わせて設定します。たとえば、次のようにします。

```
clubase:
cluster_interconnect = tcp
#
ics_ll_tcp:
ics_tcp_adapter0 = nr0
ics_tcp_nr0[0] = alt0
ics_tcp_nr0[1] = alt1
ics_tcp_inetaddr0 = 10.1.0.100
ics_tcp_netmask0 = 255.255.255.0
```

4. すべてのクラスタ・メンバを停止します。
5. すべてのクラスタ・メンバを一度に 1 つずつブートします。

## D.8 トラブルシューティング

この節では、LAN インターコネクトを誤って構成したときに起こる次のような問題について説明し、その解決方法を示します。

- メンバをブートすると、ブロードキャスト・エラーが多発して、既存メンバがパニックに陥る (D.8.1 項)。
- `ifconfig nrx` スイッチ・コマンドの実行が失敗して、「No such device nr0」というメッセージが出る (D.8.2 項)。
- クラスタで動作するアプリケーションが、既知のポートへバインドできない (D.8.3 項)。

### D.8.1 新しいメンバをブートすると、ブロードキャスト・エラーが多発して既存メンバがパニックに陥る

スパニング・ツリー・プロトコル (STP) は、クラスタ・メンバのアダプタに接続されているすべてのイーサネット・スイッチ・ポートで無効にしなければなりません。この設定は、スイッチ・ポートが単一のアダプタに接続しているか NetRAIN 仮想インタフェースに含まれる複数のアダプタに接続しているかに関係なく行う必要があります。そうしない場合、クラスタ・メンバ

にブロードキャスト・メッセージが溢れ、クラスタは、実質的にサービスを提供できなくなります。最初のメンバのブートから最後のメンバのブートまで、次のようなメッセージが大量に表示されることがあります。

```
arp: local IP address 10.1.0.100 in use by hardware address 00-00-00-00-00-00
```

これらのメッセージに続いて、以下のメッセージが表示されます。

```
CNX MGR: cnx_pinger: broadcast problem: err 35
```

新しいメンバをブートしてこのクラスタに追加すると、既存のメンバがハングしたりパニックに陥ったりする可能性があります。特に、クォーラム・ディスクが構成されていると、その可能性は高くなります。この場合は、ブート時に次のメッセージが表示されます。

```
CNX MGR: cannot form: quorum disk is in use. Unable to establish  
contact with members using disk.
```

ただし、既存メンバがハングしたりパニックに陥ったりしても、30 秒ほど経過すれば、メンバがクォーラム・ディスクの検出に成功して、クラスタを形成することもあります。

## D.8.2 NetRAIN 仮想インタフェースのデバイスを手動でフェイルオーバーできない

NetRAIN は、非アクティブ側のインタフェースがパケットを受信しているかどうかを判定することで、そのインタフェースの正常性を監視します。その場合、必要に応じて、探査パケットをアクティブ・インタフェースから送信します。非アクティブ側のインタフェースが切断されると、NetRAIN は、そのインタフェースに DEAD のマークを付けます。アクティブ側のアダプタのケーブルを引き抜くと、NetRAIN は DEAD 状態のスタンバイ・アダプタをアクティブにしようとします。このアダプタに問題が発生していなければ、フェイルオーバーは正しく行われます。

ただし、手動による NetRAIN の切り替え操作 (たとえば、`ifconfig nr0 switch`) は、異なった動作をします。この場合、正常なスタンバイ・アダプタがなければ、NetRAIN は DEAD 状態のアダプタへフェイルオーバーしようとはしません。`ifconfig nr0 switch` コマンドを実行すると、次のようなメッセージが返ってきます。

```
ifconfig ioctl (SIOCIFSWITCH) No such device nr0
```

デュアル・スイッチ構成で、一方のスイッチに電源を再投入した後すぐに、もう一方のスイッチから手動でアクティブ・アダプタをフェイルオーバーしようとする、この現象が見られます。電源を投入して初期化が行われた後



(2～3 分かかる) に手動で NetRAIN のフェイルオーバーを行えば、正しい動作が行われます。フェイルオーバーが正しく行われない場合は、スイッチのケーブル接続やアダプタを調べるとともに、`ifconfig` コマンドや `niffconfig` コマンドを使ってインタフェースの状態を調べてください。

### D.8.3 アプリケーションをポートにマップできない

LAN インターコネクトを使うクラスタの通信サブシステムでは、特に指定しなければ、クラスタのブロードキャスト・トラフィック用ランデブー・ポートとしてポート 900 が使われ、ポート 901～910 および 912～917 が非ブロードキャスト・チャンネル用に予約されます。アプリケーションがこの範囲のポートを固定的に参照するように作られていると、そのポートにバインドできません。

この状況は、LAN インターコネクトで使うポートを変更することによって解決します。`ics_ll_tcp` サブシステムの `ics_tcp_rendezvous_port` 属性と `ics_tcp_ports` 属性を `sys_attrs_ics_ll_tcp(5)` で説明しているように編集した後、クラスタ全体をリブートします。ランデブー・ポートはすべてのクラスタ・メンバで同一にしなければなりません。非ブロードキャスト・ポートはメンバ間で異なってもかまいませんが、すべてのメンバで同一にしておく方が管理が容易になります。



## 索引

### 数字および記号

/ (クラスタのルート・ファイル・システム)..... 1-4

## A

**acct** コマンド ..... 5-37

**ac** コマンド ..... 5-37

**addvol** コマンド ..... 1-5t

CFS 警告メッセージ ..... 11-19

CFS サーバがフェイルオーバーしたときの再入力 ..... 9-54

LSM と ~ ..... 1-5t

/.rhosts 内のクラスタ別名 .... 9-55

クラスタと ~ ..... 9-53

### **advanced file system**

( AdvFS を参照 )

### **AdvFS**

CFS 警告メッセージ ..... 11-19

管理コマンド ..... 9-53

クラスタでの管理 ..... 9-51

クラスタのルートの拡張 ..... 1-5t

クラスタ・ルート・ドメイン内の  
ファイルの制限 ..... 9-52

ストレージの接続性の要件 ... 9-57

メンバのブート・ドメインでのファ  
イルセットの制限 ..... 9-52

メンバのルート・ドメインヘファ  
イルセットを追加する場合の制  
限 ..... 9-53

**AdvFS** ドメイン・パニック .. 11-19

回避 ..... 11-22

~ の結果として強制アンマウン  
ト ..... 11-21

**aliasd** デーモン ..... 3-4, 3-6

geted 構成ファイルの変更 .... 3-6

RIP のサポート ..... 3-4

### **ARP** ブロードキャスト・エラー

クラスタ・メンバのブート時 D-14

### **AutoFS**

クラスタでの使用 ..... 7-17

ファイル・システムの強制アンマウ  
ント ..... 7-18

**autofs** コマンド ..... 7-15

**autofs** デーモン ..... 7-15

**autofs**mount コマンド . 7-15, 7-17

**automount** コマンド ... 7-15, 7-17

## B

**bcheckrc** コマンド ..... 5-11

### **Berkeley Internet Name Domain**

( BIND クライアント を参照 )

### **BIND**

クラスタをクライアントおよびサー  
バとして構成 ..... 7-6

**bootptab** ファイル  
省略時のクラスタ別名と～ ... 7-35  
**bttape** コマンド ..... 1-19

## C

**CAA**..... 8-1  
  **caad** デーモン..... 8-30  
  CAA デーモンの再起動 ..... 8-31  
  CAA デーモンのモニタ ..... 8-31  
  EVM とともに使用..... 8-32  
  アプリケーション・リソースの起  
    動 ..... 8-13  
  アプリケーション・リソースの再配  
    置 ..... 8-11  
  アプリケーション・リソースの停  
    止 ..... 8-15  
  状態のチェック ..... 8-3  
  トラブルシューティング ..... 11-1  
  別名と～ ..... 3-27  
  リソースの登録 ..... 8-19  
  リソース名..... 8-2  
**caa\_register** コマンド..... 8-19  
**caa\_relocate** コマンド ..... 8-11  
**caa\_stat** コマンド ..... 8-3  
**caad** デーモン ..... 8-30  
  再起動 ..... 8-31  
  モニタ ..... 8-31  
**CAA** アプリケーションの起動.. 8-13  
**CAA** アプリケーションの停止.. 8-13  
**CAA** 制御下のアプリケーションの再  
  配置 ..... 8-11  
**CAA** へのアプリケーションの登  
  録..... 8-19  
**CD-ROM** ファイル・システム

( CDFS を参照 )

### CDFS

クラスタでの CDFS の制限 .. 9-63  
クラスタでの管理..... 9-63  
サポートされた読み取りアクセ  
  ス ..... 9-45

**CD-ROM** 装置 ..... 9-11

### CDSL

Korn シェルと～ ..... 11-29  
**mkcdsl** コマンド ..... 9-2  
エクスポート ..... 9-4  
確認 ..... 9-3  
コピー ..... 9-2  
使用 ..... 9-2  
バックアップ ..... 9-64  
保守 ..... 9-3

**CDSL** のエクスポート ..... 9-4

**CDSL** のコピー ..... 9-2

### CFS

**cfsmgr** コマンド ..... 9-12  
CFS サーバの再配置 ..... 9-27  
EMFILE エラー ..... 9-37  
**raw** 入出力..... 9-32  
管理 ..... 9-11  
最適化 ..... 9-31  
サーバ ..... 9-12  
制限 ..... 9-45  
性能 ..... 9-12  
直接入出力..... 9-32  
統計情報..... 9-12  
統計情報の収集 ..... 9-12  
ファイル・システムのパーティショ  
  ニング..... 9-42  
フェイルオーバー ..... 9-12  
フェイルオーバーの制限 ..... 9-13

- ブート時のエラー ..... 11-4
- メモリ使用量の調整..... 9-37
- メンバ・ルート・ドメインの再配  
置 ..... 9-23
- cfs\_async\_biod\_threads** 属性. 9-31
- cfsd.conf** 構成ファイル  
変更 ..... 9-18
- cfsd** デーモン
  - SIGHUP シグナルの影響 .... 9-18
  - 起動と停止..... 9-17
  - クラスタでの使用..... 9-12, 9-15
  - 構成ファイルの変更..... 9-18
  - 生成された EVM イベント ... 9-18
  - 負荷分散 ..... 9-14
- cfsmgr** コマンド ..... 9-12
- cfsstat** コマンド ..... 9-12
  - 直接入出力の統計情報 ..... 9-34
- CFS** キャッシュ同期
  - df と showfsets コマンド ..... 1-5t,  
1-7t
- CFS** サーバの再配置 ..... 9-27
- CFS** の最適化
  - 後書きスレッド ..... 9-31
  - 先読みスレッド ..... 9-31
  - 直接入出力..... 9-32
  - 負荷分散 ..... 9-14
  - 方法 ..... 9-41
- CFS** の調整
  - 後書きスレッド ..... 9-31
  - 先読みスレッド ..... 9-31
  - 性能 ..... 9-31
  - 直接入出力..... 9-32
  - 統計情報の収集 ..... 9-12
  - 負荷分散 ..... 9-14
- 方法 ..... 9-41
- chargefee** コマンド..... 5-37
- ckpacct** コマンド ..... 5-37
- CLSM**  
( LSM を参照 )
- clu\_alias.config** ファイル ..... 3-5
  - SysMan Menu による変更 .... 3-2
- clu\_alias** スクリプト ..... 3-5
- clu\_delete\_member** コマンド. 5-12
  - サンプル・ログ・ファイル.... C-1
- clu\_quorum** コマンド
  - クォーラム・ディスクの定義 4-15
  - ~ によるクォーラム情報の表  
示 ..... 4-19
- clu\_wall** デーモン ..... 1-9
- clua\_services** ファイル..... 3-5
  - 内のエントリの変更..... 3-13
- cluamgr** コマンド ..... 3-1
  - nogated オプション ..... 3-8n
  - クラスタ別名へのルーティングの再  
起動 ..... 3-10
  - 説明 ..... 3-10
  - 別名からのメンバの削除 ..... 3-13
- Cluster Application Availability**  
( CAA を参照 )
- cluster\_adjust\_expected\_votes**
  - カーネル属性 ..... 4-25
- cluster\_alias**
  - 参加 ..... 3-10
  - 指定 ..... 3-10
- cluster\_event** イベント属性... 1-15
- cluster\_expected\_votes** 属性 .. 4-4
- cluster\_node\_votes** 属性..... 4-5

**cluster\_qdisk\_votes** 属性 ..... 4-5  
**CNX**  
     CFS 警告メッセージ ..... 11-4  
**CNX MGR**  
     パニック ..... 4-23  
**cnx\_pinger** ブロードキャスト・エ  
     ラー  
     クラスタ・メンバのブート時 D-14  
**crontab** ファイル ..... 5-37  
**Cw** メール・マクロ ..... 7-32

## D

**Dataless Management Services**  
     ( DMS を参照 )  
**DECnet** プロトコル ..... 7-30  
**DEFAULTALIAS** キーワード ... 3-3  
**df** コマンド ..... 1-5t  
     古い情報 ..... 1-5t  
**DHCP**  
     構成 ..... 7-1  
**diskusg** コマンド ..... 5-37  
**DMS**  
     クラスタではサポートされな  
     い ..... 1-15t  
**dodisk** コマンド ..... 5-37  
**drd\_target\_reset\_wait** 属性 ... 9-8  
**drdmgr** コマンド ..... 9-27, 9-46  
**dsfmgr** コマンド ..... 9-5  
**DVD-ROM** 装置 ..... 9-11  
**Dynamic Host Configuration**  
     **Protocol**  
     ( DHCP を参照 )

## E

**EMFILE** エラー  
     CFS メモリ使用量の調整 ..... 9-37  
**/etc/bootptab** ファイル  
     省略時のクラスタ別名と ~ ... 7-35  
**/etc/clu\_alias.config** ファイル . 3-5  
     SysMan Menu による変更 .... 3-2  
**/etc/clua\_services** ファイル .... 3-5  
     内のエントリの変更 ..... 3-13  
**/etc/exports.aliases** ファイル .. 3-6  
**/etc/fstab** ファイル  
     スワップ・エントリ ..... 9-65  
**/etc/gated.conf.member** ファイ  
     ル ..... 3-6  
**/etc/gated.conf** ファイル ..... 3-6  
**/etc/hosts** ファイル ..... 5-6  
**/etc/printcap** ファイル ..... 1-12  
**/etc/rc.config.common** ファイ  
     ル ..... 1-17t  
**/etc/rc.config.site** ファイル .. 1-17t  
**/etc/rc.config** ファイル 1-17t, 5-3t  
**/etc/sysconfigtab** ファイル  
     ics\_tcp\_ports 属性 ..... D-15  
     ics\_tcp\_rendezvous\_port 属  
     性 ..... D-15  
**Event Management**  
     ( EVM を参照 )  
**EVM** ..... 1-15  
     CAA と ~ ..... 8-32  
**exports.aliases** ファイル ..... 3-6

## F

**freezefs** コマンド  
     クラスタ内での使用 ..... 9-30

**fwtmp** コマンド ..... 5-37

## G

**gated.conf.member** ファイル.. 3-6

**gated.conf** ファイル..... 3-6

**gated** コマンド ..... 1-11

**gated** デーモン ..... 3-4

## H

**halt** コマンド ..... 5-8

**hosts** ファイル..... 5-6

### HP Insight Manager

SysMan との統合 ..... 2-3

起動 ..... 2-24

クラスタでの使用..... 2-22

構成レポート ..... 2-24

初期化された Web エージェン  
ト ..... 2-19

テスト ..... 2-19

表示のみの考慮 ..... 2-5

**hwmgr** コマンド..... 1-16t, 9-5

ネットワーク・アダプタ情報の調  
査 ..... D-5

## I

### ifaccess.conf

更新 ..... 6-1

### inetd デーモン

構成 ..... 7-29

**init -s** コマンド ..... 5-11

**init** コマンド..... 1-16

PID ..... 1-17t

停止とリブート ..... 5-8

### Insight Manager

( HP Insight Manager を参照 )

**iostat** コマンド ..... 1-5

**ipport\_userreserved** 属性 .... 3-19

### IP アドレス

クラスタの～の変更... 5-30, 5-32

クラスタ別名 ..... 3-2

メンバの～の変更..... 5-32

**IP** ルータ ..... 6-4

## J

### Java

SysMan Menu PC アプリケーショ  
ン ..... 2-20

## K

**kill** コマンド ..... 1-17t

### Korn シェル

パス名と～ ..... 11-29

## L

### LAN

ifaccess.conf の更新 ..... 6-1

### LAN インターコネクト

Memory Channel からの移行 . D-7

Memory Channel への移行.. D-11

監視 ..... D-7

高速イーサネットからギガビット・  
イーサネットへの移行 .... D-11

チューニング ..... D-3  
**lastcomm** コマンド ..... 5-37  
**lastlogin** コマンド ..... 5-37  
**last** コマンド ..... 5-37  
**lmf reset** コマンド ..... 5-11  
**lockd** デーモン ..... 7-10  
**Logical Storage Manager**  
 ( LSM を参照 )  
**lprsetup** コマンド ..... 1-12  
**lpr** コマンド ..... 7-4  
**LSM**  
 addvol コマンドと ~ ..... 1-5t  
 RAID 5 ..... 10-2  
 クォーラム・ディスク ..... 10-2  
 クラスタでの LSM の制約 ... 10-2  
 クラスタでの違い ..... 1-19t

## M

**mailconfig** コマンド ..... 7-31  
**mailsetup** コマンド ..... 7-31  
**MC\_CABLE** コマンド  
 制約 ..... 11-29  
**Memory Channel**  
 LAN インターコネクトからの移行 ..... D-11  
 LAN インターコネクトへの移行 ..... D-7  
**Message Transport System**  
 ( MTS を参照 )  
**MFS** ..... 9-45  
 書き込みアクセスのサポート 1-8t  
 クラスタ・サポート ..... 9-41  
 読み取りアクセスのサポート 1-8t

**mkcdsl** コマンド ..... 9-2  
**monacct** コマンド ..... 5-37  
**mount** コマンド  
 NFS クライアントによるクラスタ  
 別名の使用 ..... 7-13  
 server\_only オプション ..... 9-42  
 クラスタ・メンバ上での NFS ファ  
 イル・システムのマウント 7-14  
 ファイル・システムのパーティショ  
 ニング ..... 9-42  
 ループバック・マウントの非サポー  
 ト ..... 7-15  
**mount** コマンドの **server\_only** オプ  
 ション ..... 9-42  
**MTS** ..... 7-30  
**MTU** ..... D-4

## N

**net\_wizard** コマンド ..... 6-6  
**netconfig** コマンド ..... 6-6  
**NetRAIN** ..... 6-4  
 LAN インターコネクトとしての  
 チューニング ..... D-3  
 既存クラスタでの LAN インターコ  
 ネクトとしての構成 ..... D-1  
 手動によるフェイルオーバー.. D-14  
**Network Information Service**  
 ( NIS を参照 )  
**Network Interface Failure Finder**  
 ( NIFF を参照 )  
**Network Time Protocol**  
 ( NTP を参照 )  
**NFS**



クライアント・ファイル・システム  
のフェイルオーバーはサポートされ  
ない ..... 9-45  
クラスタ内での使用 ..... 7-15  
クラスタ別名との関わり ..... 3-26  
構成 ..... 7-10  
~に関する制限 ..... 7-15  
ファイル・システムのエクスポー  
ト ..... 1-10  
ファイル・システムの強制アンマウ  
ント ..... 7-17  
ファイル・システムのマウン  
ト ..... 7-17  
ループバック・マウント ..... 7-15  
**NFS** クライアント  
正しい別名の使用 ..... 7-13  
**NFS** ファイル・システム  
CDSL によるマウント ..... 7-14  
**NIFF** ..... 1-11, 6-4  
**NIS** ..... 7-3  
Yellow Pages ..... 7-3  
エンハンスド・セキュリティ機能と  
NIS の構成 ..... 7-4  
構成 ..... 7-3  
**No such device: nr0** メッセージ  
NetRAIN からの ..... D-14  
**NSPOF** ..... D-2  
**NTP**  
外部サーバ ..... 7-8  
構成 ..... 7-7  
**nulladm** コマンド ..... 5-37

## O

**ogated** デーモン  
クラスタでサポートされない . 3-4  
**OSPF** ..... 6-4  
**OSPF** ルーティング・プロトコル  
(OSPF を参照)

## P

**pac** コマンド ..... 5-37  
**PID** ..... 1-17  
**ping** コマンド  
~によるトラブルシューティン  
グ ..... 11-24  
**prctmp** コマンド ..... 5-37  
**prdaily** コマンド ..... 5-37  
**Prestoserve**  
クラスタで非対応 ..... 1-7t, 1-20t  
**printpw** コマンド ..... 5-37  
**protocols.map** ファイル ..... 7-31  
**prtacct** コマンド ..... 5-37  
**ps** コマンド ..... 1-17

## R

**RAID 5**  
LSM ..... 10-2  
**raw** 入出力 ..... 9-32  
**rc.config.common** ファイル . 1-17t  
**rc.config.site** ファイル ..... 1-17t  
**rc.config** ファイル ..... 1-17t, 5-3t  
**rcmgr** コマンド ..... 1-17t, 5-3t  
**rcp** コマンド ..... 5-6

**reboot** コマンド ..... 5-8  
**Redundant Array of Independent Disks**  
 ( RAID 5 を参照 )  
**Remote Installation Services**  
 ( RIS を参照 )  
**remove** コマンド ..... 5-37  
**/rhosts** ファイル ..... 5-6  
**RIP**  
 aliasd のサポート ..... 3-4  
**RIS**..... 7-35  
 省略時のクラスタ別名の設定 7-35  
**rlogin** コマンド..... 5-6  
 トラブルシューティング ... 11-24  
**rmvol** コマンド  
 CFS 警告メッセージ ..... 11-19  
 CFS サーバがフェイルオーバーしたときの再入力 ..... 9-54  
**/rhosts** 内のクラスタ別名の必要性 ..... 9-55  
 クラスタと ~ ..... 9-53  
**routed** デーモン  
 クラスタでサポート..... 3-4  
**rsh** コマンド..... 5-6  
**runacct** コマンド ..... 5-37

## S

**sa** コマンド ..... 5-37  
**/sbin/init.d/clu\_alias** スクリプト..... 3-5  
**Secure Shell**  
 クラスタ別名選択とメンバ間の接続要求の分散 ..... 3-17  
**sendmail** ファイル ..... 7-31  
**showfssets** コマンド..... 1-7t

古い情報..... 1-7t  
**shutacc** コマンド ..... 5-37  
**shutdown -s** コマンド  
 使用の制限..... 5-11  
**shutdown** コマンド ..... 5-7  
**SMTP**..... 7-30  
**startup** 課金サービス・コマンド..... 5-37  
**statd** デーモン ..... 7-10  
**STP**  
 NetRAIN ポートが有効になっているときの問題 ..... D-14  
**svrcfstok\_max\_percent** カーネル属性..... 9-37  
**sysman**  
 クラスタ非対応の -clone..... 1-20  
**SysMan**  
 利用可能なインタフェース・オプション..... 2-6  
**SysMan Java** アプレット  
 起動 ..... 2-20  
 クラスタでの使用..... 2-18  
 ブラウザの要件 ..... 2-19  
**SysMan Menu**..... 2-7  
 PC アプリケーションの起動 . 2-21  
 起動 ..... 2-11  
 クラスタでの使用..... 2-10  
 クラスタ別名の構成..... 3-1  
 フォーカスの考慮..... 2-10  
**SysMan Station**..... 2-8  
 起動 ..... 2-17  
 クラスタでの使用..... 2-13  
 タイプに基づく有効なアクション ..... 2-16  
 ハードウェア・ビューの例... 2-15

モニタ・ウィンドウの例 ..... 2-13  
**sysman\_clone** コマンド  
クラスタ非対応 ..... 1-20  
**SysMan** コマンド行 ..... 2-9  
クラスタでの使用 ..... 2-21

## T

**telnet** コマンド  
トラブルシューティング ... 11-24  
**/tmp** の移動 ..... 11-28  
**/tmp** の再配置 ..... 11-28  
**/tmp** ファイル ..... 11-28  
**turnacct** コマンド ..... 5-37

## U

**UFS** ..... 9-45  
UFS 読み取り/書き込みファイル・  
システムのサポート ..... 1-20t  
書き込みアクセスのサポート 1-8t  
クラスタ・サポート ..... 9-41  
ディスクット ..... 9-11  
読み取りアクセスのサポート 1-8t  
**UNIX** 間コピー・プロトコル  
( UUCP を参照 )  
**UNIX** ファイル・システム  
( UFS を参照 )  
**/usr/adm/sendmail/aliases** ファイ  
ル ..... 7-31  
**/usr/adm/sendmail/sendmail.cf**  
ファイル ..... 7-31  
**/usr/adm/sendmail/sendmail.st**  
ファイル ..... 7-31

**/usr/sbin/acct/holidays** ファイ  
ル ..... 5-37  
**/usr/spool/cron** ディレクトリ . 5-37  
**/usr/spool/mail** ディレクトリ.. 7-31  
**/usr/spool** ディレクトリ ..... 7-4  
**/usr** ファイル・システム  
クラスタ単位の **/usr** のバックアッ  
プ ..... 9-64  
**UUCP** ..... 7-30

## V

**/var**  
バックアップ ..... 9-65  
**/var/adm/sendmail/protocols.map**  
ファイル ..... 7-31  
**/var/spool/mqueue** ディレクト  
リ ..... 7-31  
**/var** ファイル・システム  
クラスタ単位の **/var** のバックアッ  
プ ..... 9-64  
**verify** コマンド ..... 1-8, 9-67  
**vm\_page\_free\_min** と  
**vm\_page\_free\_reserved** 属  
性  
制限 ..... 5-4  
**vMAC** ..... 3-19  
**/vmunix**  
CDSL と ~ ..... 9-4  
**voldisk list** コマンド  
クラスタ ..... 10-2  
**volmigrate** コマンド  
クラスタ・ルート ..... 10-2  
**volstat** コマンド

クラスタ ..... 10-2

## W

**wall** コマンド ..... 1-9

**who** コマンド

クラスタ非対応の ..... 1-18

**wtmpfix** コマンド ..... 5-37

**w** コマンド

クラスタ非対応の ..... 1-18

## X

**X Window** アプリケーション

リモートへの表示 ..... 7-37

**X.25** ..... 7-30

**xhost** コマンド

省略時のクラスタ別名をホスト名として使用する場合 ..... 7-37

## Y

**Yellow Pages**

( NIS を参照 )

## あ

アダプタ

ネットワーク ..... D-5

後書きスレッド ..... 9-31

アプリケーション

インストール ..... 5-36

クラスタからの削除 ..... 5-36

削除 ..... 5-36

アプリケーション・ライセンス ..... 5-35

アプリケーション・リソースの分

散 ..... 8-16

アプリケーション・リソース名 .. 8-2

アーカイブ ..... 9-64

## い

移行

LAN インターコネクトから Memory Channel へ ..... D-11

Memory Channel から LAN イン

ターコネクトへ ..... D-7

高速イーサネット LAN からギガ

ビット・イーサネット LAN への

移行 ..... D-11

イベント

クラスタ ..... A-1

イベント属性

cluster\_event ..... 1-15

印刷

構成 ..... 7-4

インストール, レイヤード・アプリ

ケーションの ..... 5-36

インターコネクト・アドレス

メンバの ~ の変更 ..... 5-32

インターネット・サーバ・デーモン

( inetd デーモン を参照 )

## え

エラー・メッセージ

メンバのブート時の ~ ..... 11-4

エンハンスト・セキュリティ機

能 ..... 1-13

~ を持つ NIS の構成 ..... 7-4

## か

課金サービス	
管理 .....	5-37
クラスタ非対応のコマンド...	1-18
課金サービスの <b>holidays</b> ファイル.....	5-37
課金サービスの停止 .....	5-37
仮想サブネット	
いつ使用するか .....	3-8
複数のクラスタに対して同じものを 使用する .....	3-22n
別名と~ .....	3-11
監査.....	1-13
管理ツール	
クイック・スタート.....	2-2
利用可能なインタフェース....	2-4
カーネル構築	
CDSL の考慮事項 .....	9-4
~と /vmunix CDSL.....	9-4
カーネル属性	
管理 .....	5-4

## き

期待ポート	
クラスタ.....	4-3, 4-6
~の計算.....	4-6
メンバ固有.....	4-3, 4-6
キャッシュ同期	
df と showfsets コマンド .....	1-5t, 1-7t
キャッシュの整合性とブロック・デバイス .....	9-44

## 強制アンマウント

AdvFS .....	11-21
AutoFS .....	7-17
共通サブネット	
いつ使用するか .....	3-8
共通の構成ファイル	
/etc/rc.config.common ファイル ル .....	1-17t
共用ファイル	
メモリ・マップ .....	9-40

## く

クォータ	
クラスタ内でのサポート .....	9-55
クラスタ内でのハード限界値の管 理 .....	9-56
クラスタ内でのハード限界値の設 定 .....	9-57
クォーラム	
シングルユーザ・モードと~	4-4, 5-11
喪失 .....	4-8
~の計算.....	4-6
クォーラム・アルゴリズム .....	4-6
クォーラム情報	
表示 .....	4-19
クォーラム喪失 .....	4-8
メンバのシャットダウン中の~の防 止 .....	5-8
クォーラム・ディスク	
LSM と~ .....	4-17, 10-2
交換 .....	4-18
構成 .....	4-16

使用 .....	4-13	SysMan Menu の起動 .....	2-11
信頼できる .....	4-18	SysMan Station .....	2-8, 2-13
ポート .....	4-5	SysMan Station の起動 .....	2-17
ポートの数 .....	4-17	SysMan アプリケーション ....	2-6
クォーラム・ポート .....	4-7	SysMan コマンド行 .....	2-9
~ の計算 .....	4-6	SysMan コマンド行インタフェー	
クライアント		ス .....	2-21
NFS .....	7-13	オプション .....	2-2
NFS クライアントと CDSL の使		クイック・スタート .....	2-2
用 .....	7-14	クラスタでの SysMan Menu の使	
クラスタ		用 .....	2-10
CAA .....	8-1	構成ツールとインタフェース .	2-5
CDSL .....	9-2	構成レポート .....	2-24
CFS .....	9-11	利用可能なツールとインタフェー	
管理 .....	1-2	ス .....	2-4
クラスタ <b>LSM (Logical Storage</b>		クラスタ単位のファイル・システム	
<b>Manager)</b>		バックアップ .....	9-64
( LSM を参照 )		クラスタでの <b>AdvFS</b> の管理 ...	9-51
クラスタ・イベント .....	A-1	クラスタでの <b>CDFS</b> の管理 ....	9-63
クラスタからのスタンドアロン・シス		クラスタの管理 .....	1-2
テムの復元 .....	5-15	クラスタのクォーラム	
クラスタ管理ツール .....	2-2, 2-9	計算 .....	4-2
( HP Insight Manager; SysMan		クラスタの構成	
Menu; SysMan Station も参		DHCP .....	7-1
照 )		inetd デーモン .....	7-29
HP Insight Manager ....	2-3, 2-22	NFS .....	7-10
HP Insight Manager の起動 ..	2-24	NIS .....	7-3
PC アプリケーション .....	2-20	NTP .....	7-7
PC アプリケーションの起動 .	2-21	印刷 .....	7-4
SysMan Java アプレット ....	2-18	クラスタ別名 .....	3-6
SysMan Java アプレットの起		ツール .....	2-5
動 .....	2-20	メール .....	7-30
SysMan Menu .....	2-7	クラスタの分断 .....	4-3
SysMan Menu に対するフォーカス		クラスタのメンバ .....	4-2
の考慮 .....	2-10	定義 .....	4-2

クラスタのルート .....	1-4
拡張 .....	1-5t
ミラー化 .....	10-2
クラスタ非対応の <b>fuser</b> .....	1-18
クラスタ非対応の <b>uptime</b> .....	1-18
クラスタ非対応の <b>vmstat</b> .....	1-18
クラスタ・ファイル・システム (CFS を参照)	
クラスタ・ファイル・システムの管 理 .....	9-11
クラスタ別名	
aliasd デーモン .....	3-4, 3-6
CAA と .....	3-27
NFS 要求の処理 .....	3-26
RIS と .....	7-35
vMAC サポート .....	3-19
からのメンバの削除 .....	3-13
削除 .....	3-14
省略時 .....	3-2
省略時の変更 .....	5-30
属性の変更 .....	3-12
特性 .....	3-2
の設計 .....	3-6
へのルーティングの再起動 ...	3-10
ポート・スペースの拡張 .....	3-19
モニタ .....	3-13
ルーティング・オプション ...	3-23
クラスタ別名の属性	
選択重み .....	3-5
選択優先順位 .....	3-5
ルータ優先順位 .....	3-5
クラスタ名または <b>IP</b> アドレスの変 更 .....	5-30, 5-32

クラスタ・メンバ	
管理 .....	5-1
削除 .....	5-12
シャットダウン, 停止, リブ ート .....	5-8
リブート .....	5-12
クラスタ・ルート	
verify コマンドと ~ .....	9-68
ブート時の cluster_root デバイス指 定 .....	11-8
クラスタ・ルート・ドメイン	
クラスタでの定義 .....	1-4
~ で使用されるクローン・ファイル セット .....	9-52
~ 内のファイルセットの制限	9-52
クラッシュ・ダンプ .....	11-23
生成 .....	11-23
メンバのハング時の生成 ...	11-24
クローン作成	
クラスタ非対応 .....	1-20
クローン・ファイルセット	
クラスタ・ルート・ドメイン内で使 用する場合の考慮事項 .....	9-52
クラスタ・ルート・ドメイン内の ファイルの制限 .....	9-52

## け

警告	
clu_delete_member コマンド .	5-13
sysconfigdb -u コマンド .....	5-31
クラスタ・メンバの削除 .....	5-13
現在のポート .....	4-4, 4-8

## こ

### 更新

ifaccess.conf とネットワーク・インタフェース ..... 6-1

### 構成のクローン作成

クラスタ非対応 ..... 1-20

### 構成ファイル

rc.config ファイル ..... 5-2

実行時構成変数の設定 ..... 1-17t

### 構成レポート

HP Insight Manager による生成

..... 2-24

### コマンド

Tru64 UNIX との比較 ..... 1-4

クラスタ ..... 1-2

クラスタ非対応 ..... 1-18

クラスタ用のコマンドとユーティリティ ..... 1-2

### コンソール・コマンド

MC\_CABLE コマンドの制約 11-29

### コンテキスト依存シンボリック・リンク

( CDSL を参照 )

## さ

### 最適化

CFS ..... 9-31

### サイト単位の構成ファイル

/etc/rc.config.site ファイル .. 1-17t

先読みスレッド ..... 9-31

### サブネット

仮想 ..... 3-8, 3-22

共通 ..... 3-8

サポートされていない機能 ..... 1-19

### サービス

別名とネットワーク・ポート . 3-4

## し

### 時間のずれ

NTP 管理 ..... 7-9

時刻同期 ..... 7-7

### システム管理

クラスタ ..... 1-2

### 実行時構成変数

設定 ..... 1-17t

実際に見えるポート ..... 4-4

シャットダウン ..... 5-8

CAA アプリケーションと ~ .. 8-30

クラスタのシャットダウンの取り消し ..... 5-7

シャットダウンできないメン

バ ..... 11-1

シングルユーザ・モード ..... 4-4

シングルユーザ・モードに移

行 ..... 5-11

ジャンボ・フレーム ..... D-4

### 重要な投票メンバ

クォーラム喪失とシャットダウ

ン ..... 5-8

省略時のクラスタ別名 ..... 3-2

NFS クライアントによる使用 7-13

out\_alias サービス属性に対する関

係 ..... 7-37

変更 ..... 5-30

省略時のクラスタ別名の変更 ... 5-30

シングルユーザ・モード . 4-4, 5-11

シャットダウンして ~ に移行 5-11

シンプル・メール転送プロトコル



( SMTP を参照 )

## す

- スタンダロン・システム
  - クラスタからの復元..... 5-15
- ストレージ
  - 他社製デバイスの管理 ..... 9-8
- ストレージの接続性
  - AdvFS の要件..... 9-57
- スワップ
  - 構成 ..... 9-65
  - メンバ・ブート・ディスク... 11-5

## せ

- 性能
  - CFS の最適化..... 9-31
  - CFS 負荷分散..... 9-12
  - 改善 ..... D-3
  - フル・ファイル・システムに近い状態の回避 ..... 9-40
- セキュリティ..... 1-13
  - エンハンスト・セキュリティ機能を持つ NIS の構成 ..... 7-4
- 接続性
  - AdvFS とストレージの接続性 9-57
- 接続マネージャ ..... 4-1, 4-22
  - ( CNX も参照 )
  - トラブルシューティング ..... 4-24
  - パニックのモニタ..... 4-22
- 選択重み属性..... 3-5
  - 2 つの接続を使用するアプリケーション..... 3-17

選択優先順位属性 ..... 3-5

## そ

- 属性
  - クラスタ固有のプリンタ特性 . 7-5
- ソフトウェア・ライセンス ..... 5-35

## た

- ダイレクト・アクセス入出力デバイス..... 9-46
- 単一サーバ・デバイス..... 9-46
- ダンプ..... 11-23
- ターゲット状態 ..... 8-3

## ち

- 調整
  - ポート・スペース..... 3-19
- 直接入出力 ..... 9-32
- 統計情報の収集 ..... 9-34

## て

- ディスク
  - 位置の確定..... 9-6
- ディスクの識別 ..... 9-6
- ディスクセット
  - UFS 読み取り専用ファイル・システムと..... 9-11
- デバイス
  - ダイレクト・アクセス入出力 9-46
  - 他社製ストレージの管理 ..... 9-8
  - 単一サーバ..... 9-46

デバイス位置の確定..... 9-6  
デバイス・スペシャル・ファイル管理  
ユーティリティ ..... 9-5  
デバイス・スペシャル・ファイルの管  
理..... 9-5  
デバイス要求ディスパッチャ  
drdmgr コマンド..... 9-46  
データベース・アプリケーション  
raw 入出力..... 9-32  
テープ装置 ..... 9-9  
デーモン  
aliasd ..... 3-5  
autofsd ..... 7-15  
caad ..... 8-30  
gated ..... 3-4  
inetd..... 7-29  
lockd..... 7-10  
routed ..... 3-4  
statdd..... 7-10  
プリンタ ..... 7-4

## と

同期, **CFS** キャッシュ・データ  
df と showfssets コマンド ..... 1-5t,  
1-7t  
同時直接入出力 ..... 9-32  
~の統計情報収集..... 9-34  
特性  
クラスタ固有のプリンタ特性 . 7-5  
ドメイン  
クラスタのルートの拡張 ..... 1-5t  
トラブルシューティング  
クラスタ ..... 11-1

## な

名前  
クラスタの~の変更..... 5-30  
メンバ名の変更 ..... 5-32

## に

入出力データ・キャッシング ... 9-32  
入出力バリア ..... 9-8  
drd\_target\_reset\_wait 属性 ... 9-8

## ね

ネットワーク  
トラブルシューティング ... 11-24  
フェイルオーバ ..... 6-4  
ネットワーク・アダプタ  
動作情報の調査 ..... D-5  
ネットワーク・インタフェース  
変更時の ifaccess.conf の更新 . 6-1  
ネットワーク情報サービス  
(NIS を参照)  
ネットワークの時刻同期 ..... 7-7  
ネットワーク・ファイル・システム  
(NFS を参照)  
ネットワーク・ポート  
クラスタ内の~数の拡張 ..... 3-19  
別名と~ ..... 3-4  
ネットワーク・ルーティング  
クラスタ別名構成..... 3-23

## の

ノード・ポート ..... 4-4

## は

### パニック

CNX MGR からの～ ..... 4-23

～後のブート・パーティションへの  
アクセス ..... 11-22

接続マネージャ ..... 4-23

接続マネージャからの～ ..... 4-23

ブート・パーティションがすでにマ  
ウントされているとき ... 11-23

### ハングアップ

クォラム喪失による～ ..... 5-8

ハードウェア構成 ..... 9-6

## ふ

### ファイル・システム

CDFS の管理 ..... 9-63

UFS の制限付き読み取り/書き込み  
サポート ..... 1-20t

クラスタでの AdvFS の管理.. 9-51

クラスタでの管理 ..... 9-11

クラスタでの作成 ..... 9-58

クラスタ内でのフリーズ ..... 9-30

バックアップ ..... 9-64

パーティショニング..... 9-42

フル状態の回避 ..... 9-40

リストア..... 9-64

### ファイル・システムのエクスポート

..... 1-10

ファイル・システムの作成 ..... 9-58

### ファイル・システムのバックアップ

..... 9-64

/var ディレクトリの保護 ..... 9-65

ファイル・システムのパーティショニ  
ング ..... 9-42

### ファイル・システムのフリーズ

クラスタ内での freezefs の使  
用 ..... 9-30

ファイル・システムのリストア. 9-64

フェイルオーバー ..... D-14

AdvFS の要件..... 9-57

CFS..... 9-12

CFS フェイルオーバーの制限 .. 9-13

ネットワーク ..... 6-4

ブート・パーティション ... 11-22

フォーカス ..... 2-21

SysMan Menu での考慮..... 2-10

SysMan Menu での指定..... 2-10

定義 ..... 2-10

負荷分散 ..... 9-14

CFS..... 9-12

クラスタ別名とアプリケーション  
..... 3-16

別名と～ ..... 3-5

プリンタ・デーモン ..... 7-4

プリント ..... 1-12

クラスタ用の :on 特性..... 7-5

スプール・ファイル..... 7-4

### プロセス識別子

(PID を参照)

### プロセス制御

kill コマンド ..... 1-17t

### ブロック・デバイス

キャッシュの整合性... 9-31, 9-44

### ブロードキャスト・メッセージ

wall の起動 ..... 1-9

### 分断

drd\_target\_reset\_wait 属性 ... 9-8  
 分断, クラスタ ..... 4-3  
 ブート  
   ブート時の cluster\_root デバイス指  
   定 ..... 11-8  
   メンバのブート時の CFS エラー・  
   メッセージ ..... 11-4  
   ライセンスなし ..... 11-1  
 ブート可能テープ ..... 1-19  
 ブート・ディスク  
   メンバ・ブート・ディスクの修  
   復 ..... 11-5  
 ブート・パーティション  
   修復 ..... 11-23  
   すでにマウントされていればパニッ  
   ク ..... 11-23  
   マウント ..... 11-22  
   ミラー化 ..... 10-2  
 ブート・パーティションの修復 11-23

## へ

別名 ..... 3-1  
 ( クラスタ別名 も参照 )

## ほ

ポート  
   期待 ..... 4-3  
   クォーラム ..... 4-7  
   クォーラム・ディスク ..... 4-5  
   クラスタ内の ~ 数の拡張 ..... 3-19  
   現在の ~ ..... 4-8  
   ネットワーク・ポートとクラスタ別  
   名 ..... 3-4

ノード ..... 4-4  
 非ブロードキャスト ..... D-15  
 ブロードキャスト ..... D-15  
 マップできないアプリーケーショ  
   ン ..... D-15  
 見える ..... 4-4

## め

メッセージ・トランスポート・シス  
 テム  
 ( MTS を参照 )  
 メモリ・ファイル・システム  
 ( MFS を参照 )  
 メモリ・マップ・ファイル ..... 9-40  
 メンバ ..... 4-2  
   IP アドレスの変更 ..... 5-32  
   管理 ..... 5-1  
   シャットダウン ..... 5-8  
   シャットダウン, 停止, リブ  
   ート ..... 5-8  
   定義 ..... 4-2  
   ブート・ディスクの修復 ..... 11-5  
   ブート・パーティションのマウン  
   ト ..... 11-22  
   ルート・ドメイン ..... 1-4  
   ルート・ドメインの再配置... 9-23  
 メンバシップ  
   モニタ ..... 4-22  
 メンバのブート・ドメイン  
   ファイルセットの制限 ..... 9-52  
 メンバのブート・パーティションのマ  
   ウント ..... 11-22  
 メンバのルート・ドメイン  
   ~ 内のファイルセットの制限 9-53

メンバ名, **IP** アドレス, インターコ  
ネクト・アドレスの変更 ..... 5-32  
メンバ名の変更 ..... 5-32  
メンバ・ルート・ドメインの再配  
置..... 9-23  
メール  
Cw マクロ ..... 7-32  
mailconfig および mailsetup コマン  
ド ..... 7-31  
構成 ..... 7-30  
ファイル ..... 7-31  
メールの統計情報 ..... 7-32  
メール・プロトコル  
クラスタ・サポート..... 7-30

## も

モニタ, メンバシップの ..... 4-22

## ら

ライセンス  
~なしのブート ..... 11-1  
ライセンス登録 ..... 5-2  
アプリケーション・ライセン  
ス ..... 5-35  
ランデブー・ポート ..... D-15

## り

リソース  
状態のチェック ..... 8-3  
リソース名 ..... 8-2

## る

ルータ優先順位属性 ..... 3-5  
ルーティング  
cluamgr と ~ ..... 3-6  
gated ..... 1-11  
IP ..... 6-4  
別名と仮想サブネット ..... 3-11  
ルーティング情報プロトコル  
(RIP を参照)  
ルートのミラー化 ..... 10-2  
ルート・ファイル・システム .... 1-4  
クラスタ単位のバックアップ 9-64  
メンバの再配置 ..... 9-23  
ループバック・マウント  
クラスタでの非サポート ..... 7-15

## れ

レイヤード・アプリケーション  
インストールと削除..... 5-36



# マニュアルに対するご意見

TruCluster Server  
クラスタ管理ガイド  
AA-RM85D-TE

弊社のマニュアルに関して、ご意見、ご要望、または内容の不明確な部分など、お気づきの点がございましたら、下記にご記入の上、弊社社員にお渡しくださるようお願い申し上げます。

マニュアルの採点：

	大変良い	良い	普通	良くない
正確さ(説明どおりに動作するか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
情報量(十分か)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
分かり易さ	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
マニュアルの構成	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
図(役立つか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
例(役立つか)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
索引(項目の検索性)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ページ・レイアウト(情報の検索性)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

内容の不明確な部分がありましたら、以下にご記入ください：

ペ ー ジ


その他お気づきの点がございましたら、以下にご記入ください：


ご使用のソフトウェアのバージョン： \_\_\_\_\_

貴社名/部課名 \_\_\_\_\_

御名前 \_\_\_\_\_

記入日 \_\_\_\_\_

(注) 当用紙を受け取った弊社社員は、すみやかに下記にお送りください。

ビジネスクリティカルシステム統括本部 **BCS** 技術本部 **Alpha** ソフトウェア技術部