Veritas Storage Foundation™ Cluster File System 管理者ガイド

Solaris

5.0

N18952F

symantec™

Veritas Storage Foundation Cluster File System 管理者ガイド

SFCFS 5.0

サードパーティ（第三者）製ソフトウェアの権利に関する通知

本製品には、特定のサードパーティ製ソフトウェアが配布、組み込み、または同梱されている場合があります。また、本製品のインストールおよび使用にともない、サードパーティ製ソフトウェアの使用を推奨する場合があります。同サードパーティ製ソフトウェアのライセンスは、著作権の保有者により別途付与されます。サードパーティのソフトウェアの使用に必要なライセンスおよび著作権に関する情報については、本製品リリースノートのサードパーティに関する章を参照してください。

Solaris は Sun Microsystems, Inc. の商標です。

ライセンスと登録

Veritas Storage Foundation Cluster File System はライセンスが必要な製品です。ライセンスのインストールについては、『Veritas Storage Foundation Cluster File System インストールガイド』を参照してください。

テクニカルサポート

製品のサポートを受けるには、http://support.veritas.com ページへアクセスし「Phone Support」または「E-mail Support」をクリックします。このページから TechNote、Software Alerts、ソフトウェアのダウンロード、ハードウェア互換性リスト、VERITAS Email Notifications サービス などにアクセスすることもできます。「Knowledge Base Search」機能を使用し、製品ドキュメントのリリースなどの製品情報へアクセスすることができます。

目次

1

技術的概要

この章では、次の内容について説明します。

Storage Foundation Cluster File System アーキテクチャ

Veritas Storage Foundation Cluster File System（SFCFS）を利用すると、ファイルシステムを使っているすべてのアプリケーションが同じサーバー上で動作しているかのように、複数のクラスタ化サーバーが 1 つのファイルシステムを同時にマウントして使うことができます。Veritas Volume Manager クラスタ機能（CVM）を使うと、クラスタ全体から論理ボリュームと RAW デバイスアプリケーションにアクセスできるようになります。

Storage Foundation Cluster File System の設計

SFCFS では、クラスタ内のすべてのノードをメタデータサーバーとして同時に稼動することができる対称アーキテクチャを SFCFS 5.0 から導入しています。SFCFS には、従来のマスター / スレーブまたはプライマリ / セカンダリという概念が一部に残っています。各クラスタファイルシステムを最初にマウントしたサーバーがそのプライマリになり、クラスタ内の他のノードはすべてセカンダリになります。アプリケーションは自らが動作しているサーバーから直接、ファイル内のユーザーデータにアクセスします。各 SFCFS ノードは、独自のインテントログを所有します。ファイルシステム操作（ファイルの割り当てや削除など）は、クラスタ内の任意のノードから行うことができます。

Storage Foundation Cluster File System のフェールオーバー

SFCFS プライマリが稼動するサーバーに障害が発生した場合は、残りのクラスタノードから新しいプライマリが選択されます。新しいプライマリは古いプライマリのインテントログを読み取り、障害時に処理中であったメタデータ更新を完了します。

SFCFS セカンダリ上で実行していたサーバーに障害が発生すると、プライマリは障害が発生したセカンダリのインテントログを読み取り、障害時に処理中であったメタデータ更新を完了します。

Group Lock Manager

SFCFS では、Veritas Group Lock Manager（GLM）を使って、各クラスタに UNIX の単一ホストファイルシステムのセマンティクスを複製します。これは書き込み動作で最も重要です。UNIX ファイルシステムでは、書き込み動作が原子的であるように見えます。つまり、あるアプリケーションがファイルに一連のデータを書き込むと、データがファイルシステムによってキャッシュされているがディスクに書き込まれていない場合でも、それ以降にそのファイルの同じ領域を読み取るアプリケーションはすべて、書き込まれた新しいデータを取り込みます。アプリケーションが整合性のないデータや、前回の書き込み結果の一部のみを取り込むことは一切できません。

単一ホスト書き込みセマンティクスを複製するために、書き込み元のクラスタノードに関係なく、システムのキャッシュは整合性を保持し、ノードごとにキャッシュされたデータへのすべての更新を瞬時に反映する必要があります。GLM はファイルをロックし、クラスタ内の他のノードがそのファイルを同時に更新できないようにしたり、更新が完了するまでそのファイルを読み取ることができないようにします。

クラスタファイルシステムでの VxFS 機能

Veritas Storage Foundation Cluster File System は Veritas File System（VxFS）をベースとしています。VxFS ローカルファイルシステムのほとんどの主要機能をクラスタファイルシステムで利用できます。次のものが含まれます。

- 最大 1 TB のサイズのファイルをマップするエクステントベースの領域管理
- インテントログを使ってファイルシステムメタデータの最近の更新を追跡することによるシステムクラッシュからの高速リカバリ
- ファイルシステムが使用中でもその拡張や断片化解消が可能なオンライン管理

次に示すのは、SFCFS で利用できる機能とコマンドの一覧です。VxFS のすべてのオンラインマニュアルページには、Storage Foundation Cluster File System の問題についてのセクションがあります。このセクションでは、クラスタマウントされたファイルシステムでコマンドが機能するかどうかについての情報が記載されています。また、ローカルマウントされたファイルシステムの場合との動作の違いを示しています。

『Veritas Storage Foundation Cluster File System リリースノート』を参照してください。

サポートされている機能

機能	コメント
Quick I/O	SFCFS では、クラスタ化された Oracle Disk Manager（ODM）を使った Quick I/O for Databases 機能がサポートされます。Quick I/O は Veritas Database Editions 製品でのみライセンスが交付されます。
Storage Checkpoint	Storage Checkpoint はクラスタファイルシステムでサポートされますが、他の Veritas 製品との組み合わせでのみライセンスが交付されます。
スナップショット	スナップショットはクラスタファイルシステムでサポートされます。
クォータ	クォータはクラスタファイルシステムでサポートされます。
NFS マウント	クラスタファイルシステムを NFS にマウントできます。
入れ子マウント	クラスタマウントされたファイルシステム上のディレクトリを、ローカルファイルシステムまたは別のクラスタファイルシステムのマウントポイントとして使うことができます。
凍結と解凍	ファイルシステムの凍結と解凍を必要とする同期操作はクラスタ全体に対して行われます。
メモリマッピング	`mmap()` 関数によって確立される共有メモリマッピングは SFCFS でサポートされます。 `mmap`（2）のマニュアルページを参照してください。
ディスクレイアウトバージョン	SFCFS はディスクレイアウトバージョン 6 と 7 のみをサポートします。クラスタマウントされたファイルシステムはアップグレードでき、ローカルマウントされたファイルシステムはアップグレード、マウント解除、クラスタの一部としての再マウントが可能です。VxFS システムのディスクレイアウトバージョンを確認するには、`fstyp -v` *`special_device`* コマンドを使います。ディスクレイアウトバージョンを更新するには `vxupgrade` コマンドを使います。

機能	コメント
ロック	アドバイザリファイルとレコードロックは SFCFS でサポートされます。F_GETLK コマンドについては、競合ロックを保持しているプロセスがある場合、l_pid フィールドは競合ロックを保持しているプロセスのプロセス ID を返します。ノード ID からノード名への変換は、/etc/llthosts ファイルの検証、または fsclustadm コマンドを使って行うことができます。SFCFS では強制ロックはサポートされず、従来の fcntl ロックでサポートされていたデッドロック検出もサポートされません。 fcntl（2）のマニュアルページを参照してください。

サポートされない機能

サポートされない機能として説明されている機能でも、クラスタファイルシステムに対して操作を実行できる場合はありますが、実際の動作は予測できません。サポートされていない機能を SFCFS で使わないようにしてください。また、マウントしているファイルシステムを、ローカルマウントやクラスタマウントとしてこれらのオプションで置き換えないようにしてください。

SFCFS でサポートされない機能とコマンド	
qlog	Quick Log はサポートしていません。
スワップファイル	スワップファイルは、クラスタマウントされたファイルシステムではサポートされません。
mknod コマンド	クラスタマウントされたファイルシステムで、mknod コマンドを使ってデバイスを作成することはできません。
キャッシュアドバイザリ	キャッシュアドバイザリは個別のファイルシステム上で mount コマンドによって設定されますが、クラスタの他のノードには伝達されません。
Cached Quick I/O	ファイルシステムのキャッシュにデータをキャッシュする、この Quick I/O for Databases 機能はサポートされていません。
ファイルアクセス時間に依存するコマンド	クラスタファイルシステムでは atime ファイル属性が厳密に同期されないため、ファイルアクセス時間の表示がノード間で異なる場合があります。そのため、アクセス時間のチェックに依存するユーティリティは確実に機能しない場合があります。

Storage Foundation Cluster File System のメリットとアプリケーション

Storage Foundation Cluster File System を使う利点

SFCFS は、ハードウェアの制限によるシステム管理タスクを単純化またはなくします。

- SFCFS の単一ファイルシステムのイメージ管理モデルでは、すべてのファイルシステム管理操作とサイズ変更と再構成（断片化解消）を任意のノードから実行できるようにし、管理が単純化されています。
- クラスタ内のすべてのサーバーが SFCFS のクラスタ共有ファイルシステムにアクセスできるため、複数のサーバー間でデータの一貫性が自動的に保たれます。すべてのクラスタノードは同じデータにアクセスでき、すべてのデータは、単一のサーバーファイルシステムのセマンティクスを使っているすべてのサーバーからアクセス可能です。
- すべてのファイルにすべてのサーバーからアクセスできるため、アプリケーションをサーバーに割り当てることで負荷を分散したり、その他の操作上の必要条件を満たすことができます。同様に、データへのアクセスのしやすさによって制約されないことから、フェールオーバーをより柔軟に行うことができます。
- 個々の SFCFS ファイルシステムをクラスタ内の任意のノードにすることができるため、ファイルシステムをクラスタノード間で均等に分散させることにより、*n* 個のノードクラスタでのフェールオーバー時間のうち、ファイルシステムのリカバリに占める時間を *n* 分の 1 に短縮できます。
- エンタープライズ RAID サブシステムのすべての容量がすべてのサーバーによってマウントでき、ハードウェア再構成の代わりに管理操作を行って容量を割り当てることができるため、より効率的に利用できます。
- より広範なストライプ化によるボリュームの拡張により、アプリケーション I/O の負荷分散が改善されます。各サーバーの I/O 負荷がストレージリソース間で分散されるだけでなく、SFCFS の共有ファイルシステムを利用することで、すべてのサーバーの負荷も互いに分散されます。
- 新しいサーバーはボリュームとファイルシステムに関するクラスタ全体の設定をそのまま取り入れるため、新しいサーバーのそれぞれに対してストレージ設定を行う必要はなく、サーバーの追加によるクラスタの拡張が容易になっています。
- ファイルベースのデータベースの処理効率を RAW パーティションベースのデータベースと同等にするクラスタ化 Oracle Disk Manager（ODM）機能を、クラスタ内で動作中のアプリケーションで利用できます。

Storage Foundation Cluster File System を使うタイミング

SFCFS は、ホームディレクトリ、ブートサーバーファイル、Web ページなどを対象としたファイル共有が必要なアプリケーションや、クラスタ対応アプリケーションでの利用に適しています。また、大部分が読み取り専用の、データへのアクセスのみが必要な環境でいつでも使えるスタンバイデータが必要なときや、ファイル共有のために NFS を利用したくないときにも SFCFS を利用できます。

SFCFS はほとんどすべてのアプリケーションにとってメリットがあります。「クラスタ対応」でないアプリケーションでも、クラスタ内の任意の場所にあるデータにアクセスし、操作できるようになります。異なるサーバー上で動作する複数のクラスタアプリケーションが、クラスタファイルシステム内のデータにアクセスしている場合、別々の下位ボリューム上に 1 つのクラスタファイルシステムを置く負荷分散効果によって、システム全体の I/O 処理速度が向上します。この処理は自動的に行われ、調整やその他の管理操作は不要です。

多くのアプリケーションは複数の同時実行スレッドで構成され、各スレッドのデータアクセスを調整する手段がある場合、それらのスレッドを複数の異なるサーバー上で実行できます。SFCFS はこの調整を行います。そのようなアプリケーションをクラスタ対応にし、アプリケーションのインスタンスどうしが協調してクライアント負荷とデータアクセス負荷を分散できるようになり、その結果として単一サーバーの容量を超えた拡張が可能になります。そのようなアプリケーションでは SFCFS が共有データアクセスを可能にし、クラスタノード間でのアプリケーションレベルの負荷分散ができるようにします。

- 常に利用可能でなければならない単一ホストアプリケーションの場合、SFCFS は、サーバーエラー発生後にアプリケーションが再起動できる稼働中のファイルシステム環境を提供することで、アプリケーションのフェールオーバー時間を短縮できます。
- 分散データベース管理システムや Web サーバーなどの並列アプリケーションの場合、SFCFS は、共有データをすべてのアプリケーションインスタンスに同時に提供します。また SFCFS は、サーバーの追加によってこれらのアプリケーションを拡張できるようにするとともに、サーバーエラー発生時にネットワークアドレスを再割り当てするだけで負荷を再分散できるようにすることでアプリケーションの可用性を向上させます。
- ビデオ制作など、非常に大きなファイルをステーション間で受け渡しするワークフローアプリケーションの場合、SFCFS によりすべてのステーションからファイルにアクセスできるようになるため、時間がかかってエラーになりやすいデータコピーは不要になります。
- バックアップに関しては、SFCFS では別々のサーバーで実行し、クラスタ共有ファイルシステム内のデータにアクセスすることによる操作上の影響を軽減できます。

以下に示すのは、SFCFS と連携した場合にアプリケーションがどのように機能するかの例です。

- ファイルサーバーでの Storage Foundation Cluster File System の利用
  クラスタ設定で接続された（同じクライアントと同じストレージに接続された）2 台以上のサーバーが、別々のファイルシステムとして機能します。サーバーのうち 1 台にエラーが発生すると、他のサーバーがエラーを認識し、回復させ、プライマリシップを引き受け、エラーの発生したサーバーの IP アドレスを使ってクライアントへの応答を開始します。
- Web サーバーでの Storage Foundation Cluster File System の利用
  Web サーバーのアプリケーションは通常は読み取り専用であるため、Web サーバーは共有クラスタ化に特に適しています。さらに、クライアント負荷分散フロントエンドを利用すると、サーバーとサイトのコピーを追加することで Web サーバークラスタの容量を拡張できます。SFCFS ベースのクラスタにより、このタイプのアプリケーションの拡張と管理は大幅に単純化されます。

2

Storage Foundation Cluster File System アーキテクチャ

コンポーネント製品の役割

SFCFS には、Veritas Cluster Server（VCS）と Veritas Volume Manager（VxVM）が含まれます。Veritas Cluster Server（VCS）により、クラスタの作成に必須となる通信、設定およびメンバーシップサービスを実行できます。VCS は、クラスタファイルシステムを設定する際に最初にインストールおよび設定するコンポーネントです。

Veritas Cluster Server

GAB（Group Membership and Atomic Broadcast）および LLT（Low Latency Transport）は、イーサネットデータリンクに直接実装する VCS 固有のプロトコルです。この 2 つは、クラスタ内のノードどうしを接続する冗長データリンク上で実行されます。単一点障害を回避するために、VCS には冗長クラスタ通信リンクが必要です。

GAB は、クラスタとクラスタアプリケーションにメンバーシップとメッセージサービスを提供します。また、GAB のメンバーシップにより、クラスタの起動と停止が正しい順序で実行されます。GAB を設定するには、`/etc/gabtab` ファイルを使います。設定を完了するには `gabconfig` コマンドを使います。たとえば、`-n` コマンドオプションで、クラスタのノード数を指定します。GAB は、VCS インストールスクリプトの実行時に自動的に設定されますが、クラスタにノードを追加する場合は、GAB を再設定する必要があります。

`gabconfig`（1M）のマニュアルページを参照してください。

LLT により、カーネル間の通信が可能になり、ネットワーク通信が監視されます。/etc/llthosts および /etc/lltab の各 LLT ファイルを設定して、クラスタ内のシステム ID の構成、複数のクラスタ ID の構成、ハートビートの頻度などのネットワークパラメータのチューニングを実行します。LLT により、状態変化などのイベントを即座に反映し、高速な応答が可能になります。

GAB と同様に、LLT は VCS のインストールスクリプトの実行時に自動的に設定されます。/etc/lltab ファイルには、インストール時の入力情報に基づく構成情報が含まれます。クラスタにノードを追加する際に、LLT の再設定が必要になる場合があります。

lltab（4）のマニュアルページを参照してください。

『Veritas Cluster Server ユーザーズガイド』を参照してください。

SFCFS の各コンポーネントは、メンバーシップポートに登録されます。ポートメンバーシップは、各コンポーネントに対して、クラスタを構成するノードを識別します。次に、ポートメンバーシップの例を示します。

port a ハートビートメンバーシップ

port b I/O フェンシングメンバーシップ

port f クラスタファイルシステムメンバーシップ

port h GAB と高可用性デーモン（HAD）との間の Veritas Cluster Server 通信のメンバーシップ

port u CVM によって一時的に使われるメンバーシップ

port v Cluster Volume Manager メンバーシップ

port w 複数のノード上の Cluster Volume Manager デーモンがこのポートを使って相互に通信するが、クラスタメンバーシップ情報は GAB（ポート v）を通じて受信するメンバーシップ

Veritas Volume Manager のクラスタ機能

Veritas Volume Manager クラスタ機能（CVM）を使うと、クラスタ全体のアプリケーションから論理ボリュームにアクセスできるようになります。CVM により、複数のホストから制御下にある論理ボリュームに並列アクセスできるようになります。VxVM クラスタは、一連のデバイスを共有するノードで構成されます。これらのノードは、1 つのネットワークを介して接続されます。1 つのノードに障害が発生しても、他のノードはデバイスにアクセスできます。VxVM クラスタ機能により、すべてのノードで、変更などのデバイス設定が同じ論理ビューとして表示されます。VCS によってクラスタの設定がセットアップされたら、CVM 共有ストレージを設定します。

Storage Foundation Cluster File System について

SFCFS プライマリが稼動するサーバーに障害が発生した場合は、残りのクラスタノードから新しいプライマリが選択されます。新しいプライマリはファイルシステムのインテントログを読み取り、障害時に処理中であったメタデータ更新を完了します。この処理中は、他のノードからのアプリケーション I/O がブロックされて、遅延が発生することがあります。ファイルシステムの整合性が回復すると、アプリケーションの処理が再開されます。

SFCFS を使うセカンダリモードのノードでは、ファイルシステムのメタデータを直接には更新しないため、セカンダリノードの障害が発生してもどのメタデータも修復する必要はありません。そのため、セカンダリノードの障害からの SFCFS のリカバリは、プライマリノードの障害からのリカバリより高速です。

21 ページの「クラスタ内の負荷分散」を参照してください。

Storage Foundation Cluster File System と Group Lock Manager

SFCFS では、Veritas Group Lock Manager（GLM）を使って、各クラスタに UNIX の単一ホストファイルシステムのセマンティクスを複製します。UNIX ファイルシステムでは、書き込み動作が原子的であるように見えます。つまり、あるアプリケーションがファイルに一連のデータを書き込むと、データがファイルシステムによってキャッシュされているがディスクに書き込まれていない場合でも、それ以降にそのファイルの同じ領域を読み取るアプリケーションは、書き込まれた新しいデータを取得することができます。アプリケーションが整合性のないデータや、前回の書き込み結果の一部のみを取得することはありません。

単一ホスト書き込みセマンティクスを複製するために、書き込み元のノードに関係なく、システムのキャッシュは整合性を保持し、ノードごとにキャッシュされたデータへの更新を瞬時に反映する必要があります。

非対称マウント

mount -o cluster オプション付きでマウントされた VxFS ファイルシステムは、ローカルマウント（非共有マウント）ではなく、クラスタマウント（共有マウント）です。共有モードでマウントされたファイルシステムはクラスタ環境で VxVM 共有ボリュームに配置される必要があります。ローカルマウントを共有モードで再マウントしたり、共有マウントをローカルモードで再マウントすることはできません。クラスタ内のファイルシステムは、異なる読み取りおよび書き込みオプションでマウントできます。このようなファイルシステムは非対称マウントと呼ばれます。

非対称マウントは、共有ファイルシステムを異なる読み取りおよび書き込み機能でマウントされます。クラスタ内の 1 つのノードは読み取りおよび書き込みモードでマウントし、一方で、他のノードは読み取り専用モードでマウントすることができます。

クラスタ読み取りおよび書き込み（crw）オプションは、最初にファイルシステムをマウントするときに非対称マウントを行うことを指定する場合に使います。また、再マウント（mount -o remount）を実行して、オプションを変更する際に、指定することもできます。次の表の左列は、プライマリがマウントされるモードを示します。チェックマークは、セカンダリマウントが使えるモードを示します。

mount_vxfs（1M）のマニュアルページを参照してください。

		セカンダリ		
		ro	rw	ro、crw
プライマリ	ro	✔		
	rw		✔	✔
	ro、crw		✔	✔

-o cluster,ro オプションのみを使ってプライマリをマウントすると、セカンダリが異なるモード（読み取りおよび書き込み）でマウントされることを回避できます。rw オプションは、クラスタ全体で読み取りおよび書き込み機能を使う際に指定します。

並列 I/O

一部の分散アプリケーションは、クラスタ内の 1 つ以上のノードから同じファイルに同時に読み取りや書き込みを行うことがあります。たとえば、1 つのスレッドがファイルに追加を行い、同時にいくつかのスレッドがそのファイルの様々な領域から読み取りを行うような分散アプリケーションがあります。一部の HPC（高性能コンピュータ）アプリケーションにも、利点として、このように 1 つのファイルに同時に I/O を行う機能があります。並列 I/O の機能を使うためにアプリケーションを変更する必要はありません。

従来は、小さな領域への I/O を実行するためにファイル全体をロックしていました。並列 I/O をサポートするために、SFCFS は、I/O の要求に対応するファイル内の領域をロックします。ロックされる領域の単位はページです。2 つの I/O 要求があり、その少なくとも一方が書き込み要求で、その I/O 範囲がもう一方の I/O 範囲と重なる場合、それらの I/O 要求は競合します。

並列 I/O の機能により、要求が競合しない限り、複数のスレッドから 1 つのファイルへの同時 I/O が可能になります。I/O 要求を同時に発行する複数のスレッドは、同じノードで実行されていても、クラスタ内の異なるノードで実行されていてもかまいません。

メモリ割り当てが必要な I/O 要求を他の I/O 要求と同時に実行することはできません。書き込み側のファイルが拡張しており、読み取り側がこれより遅れている場合は、拡張部分に書き込むごとに必ずしもブロックの割り当てが行われないことに注意してください。

ファイルのサイズを事前に決定できる場合は、事前にファイルを割り当てて、I/O 時のブロックの割り当てを回避することができます。これにより、ファイルに並列 I/O を実行するアプリケーションの同時処理能力が向上します。また、並列 I/O により、クラスタ全体のキャッシュの整合性を損なわずに、ページキャッシュの無駄なフラッシュや無効な領域ロックの使用も回避します。

複数のノードから 1 つのファイルを更新するアプリケーションに対しては、マウントのオプション `-nomtime` によって同時処理能力がさらに向上します。クラスタ全体でファイルの変更と変更時刻が同期されないため、I/O とロックが増えることによるオーバーヘッドはありません。ノードのこれらのファイルに表示されるタイムスタンプは、過去 60 秒間に更新されていない可能性があります。

Storage Foundation Cluster File System の名前空間

マウントポイントの名前は、同じクラスタファイルシステムをマウントするすべてのノードで同じにする必要があります。この設定は、VCS マウントエージェント（オンライン、オフラインおよび監視）が正しく動作するために必要です。

Storage Foundation Cluster File System のバックアップ方法

名前空間にアクセスするための API やコマンドが同じであるため、VxFS Standard Edition で使われているバックアップ方法と同じ方法を SFCFS でも使えます。ファイルシステムの CheckPoint は、ファイルシステムのオンディスク PITC（point-in-time copy）です。クラスタファイルシステムの静止イメージを取得する際は、I/O パターンによって、チェックポイントを作成したファイルシステムの方が処理効率上有効なため、クラスタファイルシステムの静止イメージを取得する際は、ファイルシステムのスナップショットより CheckPoint を使うことをお勧めします（下記を参照）。

ファイルシステムのスナップショットも、ファイルシステムのオンディスク静止イメージを取得する方法です。チェックポイント機能とは対照的に、この静止イメージは非永続的です。スナップショットに読み取り専用モードでマウントされたファイルシステムとしてアクセスすると、ファイルシステムのオンラインバックアップを効率よく実行できます。スナップショットは、スナップファイルシステムでデータブロックが上書きされるとデータブロックを逐次コピーするコピーオンライトセマンティクスを実装しています。クラスタファイルシステムのスナップショットは、クラスタノードから発生する I/O のコピーオンライト機構を拡張します。

バックアップ用にスナップショットファイルシステムをマウントすると、コピーオンライトを実行し、スナップショットからデータブロックを読み取るためにリソースが使われるため、システム上の負荷が増大します。このような場合、クラスタのスナップショットを使ってオフホストバックアップを行うことができます。オフホストバックアップは、プライマリサーバーでのバックアップアプリケーションによる負荷を軽減します。スナップショット全体のオーバーヘッドに比べて、リモートスナップショットからのオーバーヘッドは小さくて済みます。したがって、比較的負荷の少ないノードからスナップショットをマウントして、バックアップアプリケーションを実行すると、クラスタ全体の処理効率に効果があります。

クラスタスナップショットには、次の特性があります。

- マウントしたクラスタファイルシステムのスナップショットは、クラスタ内の任意のノードにマウントできます。マウントしたファイルシステムは、プライマリ、セカンダリ、またはセカンダリ専用にできます。ファイルシステムの安定したイメージを任意のノードからの書き込み用に提供できます。
- クラスタファイルシステムの複数のスナップショットを同じクラスタノードまたは異なるクラスタノードにマウントできます。
- スナップショットは、スナップショットをマウントしているノードでのみアクセスできます。スナップショットデバイスを 2 つのノードに同時にマウントすることはできません。

- スナップショットをマウントするためのデバイスとしてローカルディスクまたは共有ボリュームを使えます。共有ボリュームは、スナップショットマウントによって排他的に使われ、スナップショットがそのデバイスでアクティブである間は、他のノードから使えません。
- スナップショットをマウントしているノードでは、スナップショットがマウントされている間はスナップショットを取られたオリジナルのファイルシステムのマウントを解除できません。
- SFCFS スナップショットは、スナップショットのマウントが解除された場合、またはスナップショットをマウントしているノードに障害が発生した場合、消失します。ただし、他のノードがクラスタから離れたり、クラスタに参加しても、スナップショットはその影響を受けません。
- 読み取り専用でマウントされたファイルシステムのスナップショットは作成できません。クラスタファイルシステムのスナップショットは、クラスタファイルシステムが crw オプションでマウントされている場合のみ、スナップショットを取得できます。

ファイルレベルの静止イメージの他に、Mirror Split および再結合を使って、共有ボリュームでボリュームレベルの静止イメージを使うこともできます。FastResync、Space Optimized スナップショットなどの機能も使えます。

『Veritas Volume Manager システム管理者ガイド』を参照してください。

クラスタファイルシステムの時間の同期

SFCFS では、Network Time Protocol（NTP）デーモンなどの外部コンポーネントを使って、すべてのノードのシステムクロックを同期化する必要があります。ノードを同期化しないと、ファイル作成時のタイムスタンプ（ctime）と同じファイル修正時のタイムスタンプ（mtime）が実際の操作順序と一致しない場合があります。

クラスタ内の負荷分散

たとえば、8 つのファイルシステムと 4 つのノードがある場合、ノードごとに 2 つのファイルシステムをプライマリとして指定すると効果的です。ファイルシステムをマウントする最初のノードが、そのファイルシステムのプライマリノードになります。

また、fsclustadm を使って、SFCFS プライマリを指定することもできます。プライマリを変更するには、マウントポイントに対して fsclustadm setprimary を使います。マウントを解除または再ブートすると、プライマリに加えた変更が無効になります。クラスタ内の 1 つ以上のノードでファイルシステムがマウントされている間、この変更は有効です。プライマリの選択ポリシーを SFCFS のマウントリソースに関連する VCS 属性によって定義することもできます。

ファイルシステムの設定

チューニングパラメータは、tunefstab ファイルまたは vxtunefs コマンドを使ってマウントした際に更新されます。ファイルシステム tunefs のパラメータは、すべてのノードで同じになるように各クラスタノードに伝達されます。ファイルシステムがノードにマウントされる場合は、プライマリノードの tunefs パラメータが使われます。このノードが、そのファイルシステムをマウントする最初のノードである場合は、ノードの tunefstab ファイルが使われます。このファイルは各ノードで同じにしておくことをお勧めします。

スプリットブレインと Jeopardy 処理

クラスタノード間でクラスタのメンバーシップビューが異なり、データの破損の可能性が増加した場合は、スプリットブレイン状態になります。プライベートリンクのすべてのクラスタ相互接続に同時に障害が発生した場合や、あるノードがハートビートメッセージに応答できない場合にも、メンバーシップの変更が発生します。データが破損する可能性は、I/O フェンシングによって抑えられます。I/O フェンシングには、SCSI-3 PGR をサポートするディスクが必要です。

Jeopardy 状態

I/O フェンシングを使わない場合、SFCFS のインストールには 2 つのハートビートリンクが必要です。単一のハートビート接続に対してノードが停止した場合、SFCFS は、システムの障害か最終的なネットワーク接続の障害かを区別できなくなります。この状態は「jeopardy」と呼ばれます。

SFCFS は、スプリットブレインによるデータ損失を jeopardy によって回避します。状況によっては、データ損失の可能性が残ることに注意してください。次に例を示します。

- すべてのリンクが同時に切断される。
- 1 つのノードが停止し、ハートビートメッセージに応答できない。

このような状況でデータの破損の可能性をなくすには、I/O フェンシングが必要です。I/O フェンシングを使う場合、jeopardy 状態は、SFCFS スタックによる特別な処理を必要としません。

Jeopardy 処理

SCSI-3 PGR をサポートしないインストールの場合、潜在的なスプリットブレイン状態は jeopardy 処理によって保護されます。jeopardy 状態の通知に従うことができないクラスタノードがあった場合、それらのノードでマウントされているクラスタファイルシステムは無効になります。jeopardy 状態の通知後に、あるノードに障害が発生した場合、すべてのクラスタノードは、共有ディスクグループのメンバーシップも解放します。

Jeopardy からのリカバリ

無効になったファイルシステムは、マウントの強制解除によってリストアできます。また、再ブートしなくてもリソースをオンラインにでき、これによって共有ディスクグループのリソースもオンラインになります。jeopardy 条件が修正されていない場合、それらのクラスタは、その後のノードの障害でクラスタから切断されやすくなることに注意してください。

『Veritas Cluster Server ユーザーズガイド』を参照してください。

フェンシング

I/O 対応のフェンシングを使うと、jeopardy 処理では回避できないデータの破損を防止することができます。

45 ページの「フェンシングの管理」を参照してください。

単一ネットワークリンクと信頼性

環境によっては、ネットワーク障害に備えるための冗長性がなくなっても、クラスタ内のノードの接続に単一のプライベートリンクまたはパブリックネットワークを使う方がよい場合があります。ハードウェアのトポロジーが単純で、費用も抑えられる利点はありますが、格段に可用性が低下します。

このような環境においては、SFCFS により、単一のプライベートリンクを選択できます。つまり、パブリックネットワークをプライベートリンクとして使えます（I/O フェンシングが存在する場合）。このとき、ノードは、jeopardy 状態（45 ページの「I/O フェンシング」を参照）で起動します。スプリットブレインは、I/O フェンシングを使って処理されます。単一ネットワークは、インストール時に選択できます。

優先度の低いリンク

LLT は、通常のハートビートチャネルのバックアップとして、優先度の低いリンクを使うように設定できます。優先度の低いリンクは、カスタマのパブリックネットワークや管理用のネットワークで設定されることが普通です。通常、これは、クラスタのプライベートな相互接続とはまったく異なるネットワークインフラストラクチャになり、1 つの問題によってすべてのリンクが停止する可能性が減少します。優先度の低いリンクは、これだけが最後に残らない限り、クラスタメンバーシップのトラフィックには使われません。通常の稼動状態で優先度の低いリンクは、クラスタメンバーシップおよびリンク状態のメンテナンスのハートビートトラフィックのみを伝送します。ハートビートの頻度は、ネットワークのオーバーヘッドを減少させるために 50 % になります。優先度の低いリンクだけがネットワークリンクとして残った場合、LLT は、すべてのクラスタの状態に関するデータのやり取りも切り替えます。設定済みのプライベートリンクのいずれかが修復されると、LLT は、クラスタの状態に関するデータのやり取りを「優先度の高いリンク」に戻します。

クライアントが接続している間も、LLT リンクを追加または削除できます。GAB や高可用性デーモン HAD を終了する必要はありません。

リンクを追加するには

```
# lltconfig -d device -t tag
```

リンクを削除するには

```
# lltconfig -u tag
```

変更はすぐに適用されますが、再ブートすると元に戻ります。再ブート後も変更を有効にするには、/etc/lltab を更新する必要があります。

メモ : LLT クライアントは、リンクが 1 つしかなく、GAB が jeopardy を宣言している場合にのみ相違を認識します。

I/O エラー処理ポリシー

I/O エラーは、ファイバーチャネルリンク、ホストバスアダプタ、ディスクの障害などの様々な原因で発生します。SFCFS は、I/O エラーが発生したノードでファイルシステムを無効にします。ファイルシステムは、他のノードからは引き続き使えます。

ハードウェアの障害が修正（ファイバーチャネルリンクの再確立など）されたら、ファイルシステムのマウントを強制解除し、マウントリソースを無効のノードからオンラインにしてファイルシステムを再起動できます。

Veritas Volume Manager のクラスタ機能について

CVM を利用すると、クラスタ内の最大 32 個のノードが VxVM 制御下のディスク（VM ディスク）の集合に同時にアクセスし、管理できるようになります。各ノード上でディスク構成と任意の変更の同じ論理表示を利用できます。クラスタ機能が有効なとき、すべてのクラスタノードが VxVM オブジェクトを共有できます。ミラー化、高速ミラー再同期、DRL（Dirty Region Logging）など、基本ボリュームマネージャによって提供される機能もクラスタ環境でサポートされます。

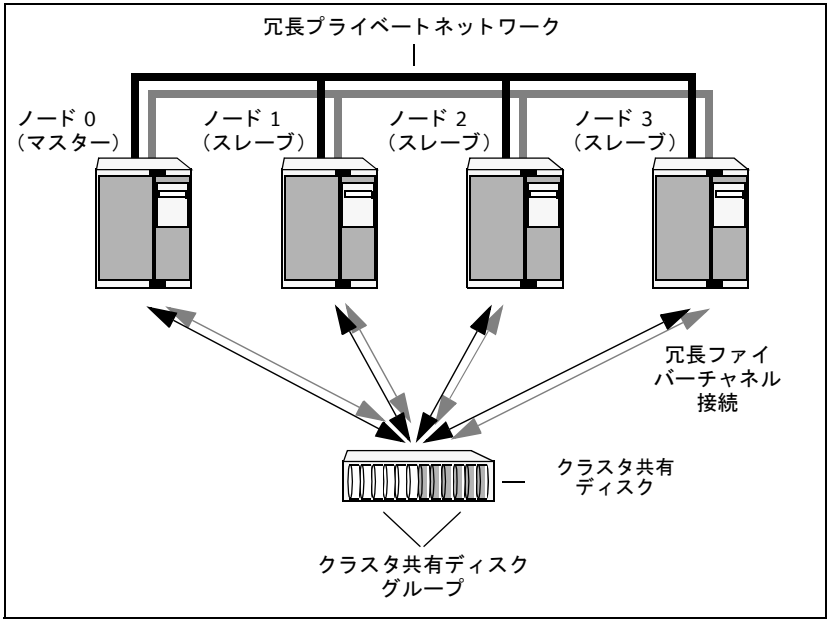
メモ : 共有ディスクで RAID 5 ボリュームはサポートされていません。

クラスタ機能を実装するために、VxVM はホストのオペレーティングシステムまたは VCS によって提供されるクラスタモニタデーモンと連携します。クラスタモニタは、クラスタメンバーシップの変更を VxVM に通知します。VxVM の各ノードは独自に起動し、クラスタモニタと OS のコピーおよび CVM をそれぞれ独自に備えています。あるノードをクラスタに結合すると、そのノードは共有ディスクにアクセスできるようになります。クラスタから切り離されたノードは、それ以降共有ディスクにアクセスできなくなります。クラスタモニタをノード上で起動すると、そのノードはクラスタに参加します。

26 ページの「4 ノードクラスタの例」の図は、類似した、または同じハードウェア特性（CPU、RAM、ホストアダプタ）を持ち、同一のソフトウェア（オペレーティングシステムを含む）で設定された 4 つのノードで構成される単純なクラスタ配置の例です。ノードはプライベートネットワークで互いに接続され、ファイバーチャネル経由で共有外部ストレージ（ディスクアレイまたは JBOD（just a bunch of disks））とも個別に接続されます。各ノードはこれらのディスク（1 つ以上のクラスタ共有ディスクグループで構成される）への 2 つの独立したパスを持ちます。

プライベートネットワークにより、ノードはシステムリソースと互いの状態についての情報を共有できます。プライベートネットワークを使って、すべてのノードが現在アクティブなノード、クラスタに参加しているノード、クラスタから切り離されているノード、障害が発生しているノードを認識できます。プライベートネットワークは、チャネルの 1 つで障害が発生した場合に備えた冗長性を確保するために、少なくとも 2 つの通信チャネルを必要とします。1 つのチャネルのみが使われていた場合、その障害はノードの障害と区別が付きません。これは「ネットワーク分割」として知られる状況です。

図 2-1 4 ノードクラスタの例



クラスタモニタにとって、すべてのノードが等価です。共有ディスクグループ内に設定された VxVM オブジェクトは、潜在的に、クラスタに参加しているすべてのノードからアクセスされる可能性があります。ただし、VxVM のクラスタ機能では、ノードの 1 つがマスターノードとして機能し、クラスタ内の他のノードすべてがスレーブノードとして機能するメンバーシップが要求されます。任意のノードがマスターノードとして VxVM の特定の動作の調整役を担うことができます。

メモ : VxVM オブジェクトを設定または再設定するコマンドは、マスターノード上で実行する必要があります。マスターノードから開始する必要がある作業には、共有ディスクグループの設定、ボリュームの作成と再設定などがあります。

VxVM は、クラスタに最初に参加したノードがマスターノードとして機能する仕様です。マスターノードがクラスタから切断されると、スレーブノードの 1 つが新しいマスターに選択されます。前述の例では、ノード 0 がマスターノード、ノード 1、2 および 3 がスレーブノードです。

専有および共有ディスクグループ

ディスクグループには 2 種類あります。

- 専用ディスクグループは、1 つのノードにのみ所属します。専用ディスクグループのインポートは、1 つのシステムでのみ行われます。専用ディスクグループ内のディスクには、物理的には 1 つ以上のシステムからアクセスできますが、インポートは 1 つのシステムに制限されます。ルートディスクグループは、常に、専用ディスクグループです。
- 共有ディスクグループはすべてのノードで共有されます。共有（またはクラスタ共有）ディスクグループは、すべてのクラスタノードにインポートされます。共有ディスクグループ内のディスクは、クラスタに参加可能なすべてのシステムから、物理的にアクセスできる必要があります。

クラスタでは、大多数のディスクグループが共有されます。共有ディスクグループ内のディスクは、クラスタ内のすべてのノードからアクセスでき、複数のクラスタノード上のアプリケーションが同じディスクに同時にアクセスすることもできます。共有ディスクグループ内のボリュームは、ライセンスおよびディスクグループアクティブ化モードによる制約の範囲内で、クラスタ内の複数のノードから同時にアクセスできます。

vxdg コマンドを使って、ディスクグループをクラスタ共有に設定することができます。ディスクグループを 1 つのノードに対してクラスタ共有としてインポートすると、各ディスクヘッダーにはクラスタ ID が設定されます。各ノードを順次クラスタに参加していくと、ディスクグループをクラスタ共有であると認識しインポートします。共有ディスクのインポートまたはデポートを随時実行できます。この操作は、すべてのノードに対して分散方式で実行されます。

各物理ディスクは、一意のディスク ID でマークされます。マスターノード上で VxVM のクラスタ機能を起動すると、マスターノードはすべての共有ディスクグループを（noautoimport 属性セットが設定されている場合を除き）インポートします。スレーブノードがクラスタへの参加を試みると、マスターノードはインポートされているディスク ID の一覧をスレーブノードに送信し、スレーブノードはそのすべてにアクセスできるかをチェックします。スレーブノードからアクセスできないディスクがある場合には、クラスタへの参加は中断されます。一覧内のディスクすべてにアクセスできる場合には、スレーブノードは、マスターノードと同じ共有ディスクグループをインポートして、クラスタに参加します。ノードがクラスタから切断されると、そのノードにインポートされていた共有ディスクグループはすべてデポートされますが、それらのディスクグループは残りのノードからはデポートされません。

共有ディスクグループの再設定は、すべてのノードで協調して実行されます。ディスクグループの設定の変更は、すべてのノード上で同時に実行され、各ノード上の変更は同一になります。これらの変更には原子的な性質があります。これは、この変更がすべてのノード上で同時に発生するか、あるいはまったく発生しないかのどちらかであることを意味します。

クラスタのすべてのメンバーがクラスタ共有ディスクグループへの読み書きアクセスを実行できるかどうかは、「共有ディスクグループのアクティブ化モード」で説明する（クラスタ共有ディスクグループの）アクティブ化モードの設定によって異なります。クラスタ内の少なくとも 1 つのノードがアクティブ状態であれば、クラスタ共有ディスクグループ内に格納されているデータは使用可能です。あるクラスタノードの障害が、残りのアクティブノードのアクセスに影響を与えることはありません。どのノードからアクセスしても、クラスタ共有ディスクグループの設定は同一に見えます。

メモ : 各ノード上で稼動しているアプリケーションは、同時に VM ディスク上のデータにアクセスできます。VxVM は、複数のノードによる共有ボリュームへの同時書き込みに対する保護を行いません。アプリケーション側で（たとえば Veritas Storage Foundation Cluster File System や Distributed Lock Manager（DLM）を使って）整合性を管理することが前提となっています。

共有ディスクグループのアクティブ化モード

共有ディスクグループのアクティブ化は、ノード上のアプリケーションからの I/O に、このディスクグループへのアクセスを許可するため、そのノード上で実行する必要があります。アプリケーションがボリュームに対して読み取りや書き込みができるかどうかは、共有ディスクグループのアクティブ化モードで指示されます。共有ディスクグループの有効なアクティブ化モードには、`exclusivewrite`（排他書き込み）、`readonly`（読み取り専用）、`sharedread`（共有読み取り）、`sharedwrite`（共有書き込み）、および `off`（非アクティブ化）があります。このアクティブ化モードについて詳しくは、「共有ディスクグループのアクティブ化モード」の表で説明しています。

メモ : ディスクグループのアクティブ化は VxVM 3.0 の新機能でした。以前のリリースとの互換性を保つため、共有ディスクグループのデフォルトのアクティブ化モードは `shared-write`（共有書き込み）になっています。

高可用性（HA）アプリケーションやオフホストバックアップなどのクラスタの特殊な用途では、ディスクグループのアクティブ化を使って、クラスタ内の個別のノードからのボリュームアクセスを明示的に制御できます。

表 2-1　　共有ディスクグループのアクティブ化モード

アクティブ化モード	説明
`exclusivewrite`（ew）	ノードはディスクグループに対する排他書き込みのアクセス権を持ちます。書き込みアクセスを実行するためにディスクグループをアクティブ化できる他のノードはありません。

表 2-1　　共有ディスクグループのアクティブ化モード

アクティブ化モード	説明
readonly (ro)	ノードはディスクグループに対する読み取りのアクセス権を持ち、クラスタ内の他のすべてのノードの書き込みアクセスを拒否します。ノードはディスクグループに対する書き込みのアクセス権は持ちません。書き込みモードのいずれかを他のノード上で実行するためにディスクグループのアクティブ化を試みると失敗します。
sharedread (sr)	ノードはディスクグループに対する読み取りのアクセス権を持ちます。ノードはディスクグループに対する書き込みのアクセス権は持ちませんが、他のノードは書き込みアクセスを取得できます。
sharedwrite (sw)	ノードはディスクグループに対する書き込みのアクセス権を持ちます。
off	ノードはディスクグループに対する読み取りと書き込みのいずれのアクセス権も持ちません。ディスクグループに対するクエリー操作は許可されます。

次の表は、共有ディスクグループのアクティブ化モードの対応関係の概略を示します。

表 2-2　　アクティブ化モードの対応関係

クラスタでのディスクグループのアクティブ化モード	別のノードでディスクグループをアクティブにする際のモード			
	exclusive-write	**readonly**	**sharedread**	**sharedwrite**
exclusivewrite	失敗する	失敗する	成功する	失敗する
readonly	失敗する	成功する	成功する	失敗する
sharedread	成功する	成功する	成功する	成功する
sharedwrite	失敗する	失敗する	成功する	成功する

アクティブ化モードをユーザーの制御下に置くには、次の行を含むデフォルトファイル /etc/default/vxdg を作成します。

```
enable_activation=true
default_activation_mode=activation-mode
```

アクティブ化モードは、exclusivewrite（排他書き込み）、readonly（読み取り専用）、sharedread（共有読み取り）、sharedwrite（共有書き込み）、off（非アクティブ化）のいずれかです。

共有ディスクグループが作成またはインポートされると、指定したモードにアクティブにされます。ノードがクラスタに参加すると、そのノードからアクセス可能なすべての共有ディスクグループが、指定したモードでアクティブにされます。

メモ: ディスクグループのアクティブ化モードは、クラスタ内の各ノードからのボリューム I/O を制御します。指定されたノードのディスクグループが、クラスタ内の別のノード上で競合するモードでアクティブにされている場合、そのディスクグループをアクティブにできません。デフォルトファイルを使ってアクティブ化を有効にする場合、そのファイルをクラスタ内のすべてのノードで同じにすることをお勧めします。そうしないと、アクティブ化の結果が予測不能になります。

vxconfigd デーモンがすでに稼動しているときにデフォルトファイルを編集する場合は、デフォルトファイルへの変更を有効にするために、vxconfigd プロセスを再起動する必要があります。

デフォルトのアクティブ化モードが off 以外のディスクグループに対して、クラスタ内の別のノードが競合するモードでアクティブにすると、クラスタへの参加、またはディスクグループの作成やインポートに続いて行われるアクティブ化の適用に失敗することがあります。

共有ディスクグループのアクティブ化モードを表示するには、vxdg list *diskgroup* コマンドを使います。

また、vxdg コマンドを使って、共有ディスクグループのアクティブ化モードを変更することもできます。

『Veritas Volume Manager 管理者ガイド』を参照してください。

共有ディスクグループの接続性ポリシー

クラスタ内の各ノードは、ディスクの状態に関して常に合意する必要があります。特に、1 つのノードが指定したディスクに書き込めない場合、書き込み操作の結果が呼び出し元に返される前に、すべてのノードがそのディスクへのアクセスを停止する必要があります。このため、あるノードがディスクにコンタクトできない場合は、別のノードにコンタクトしてディスクの状態をチェックします。ディスクに障害が発生している場合、そのディスクにはどのノードからもアクセスできないためノード間で合意が成立し、ディスクを切断できるようになります。ディスクではなく、いくつかのノードからのアクセスパスに障害が発生している場合、ディスクの状態に関するノード間の合意が成立しません。この種類の不具合を解決するために、次のポリシーのいずれかを適用できます。

- グローバル接続性ポリシーでは、クラスタ内のいずれかのノードがディスク障害を報告した場合は、クラスタ全体で切断を実行します。これはデフォルトのポリシーです。
- ローカル接続性ポリシーでは、ディスクの障害の範囲をその障害を認識した特定のノードに限定します。ただし、ミラーの 1 つが利用できても、ノードが停止するため、このポリシーは高可用性ではありません。ディスクの可用性を確認するために、クラスタ内のすべてのノードとの通信を試行し、すべてのノードがそのディスクに関して問題を報告した場合には、クラスタ全体で切断を実行します。

共有ディスクグループの制限

VxVM のクラスタ機能は、RAID 5 ボリュームまたはクラスタ共有ディスクグループに対するタスクの監視をサポートしていません。ただし、これらの機能は、クラスタの特定のノードに接続されている専用ディスクグループに対しては使えます。オンラインでの再レイアウトは、サポートされていますが、RAID 5 ボリュームは除きます。

ルートディスクグループはクラスタ共有にすることはできません。ルートディスクグループは専用ディスクグループである必要があります。

VxVM のクラスタ機能のみを使う場合、RAW デバイスだけがアクセス可能です。Veritas Storage Foundation Cluster File System などの適切なソフトウェアがインストールされ設定されている場合を除き、共有ボリューム内のファイルシステムへの共有アクセスはサポートされていません。

サポートされていないオブジェクトが共有ディスクグループに含まれている場合は、そのオブジェクトをデポートしてからクラスタノードのうちの 1 つで専用ディスクグループとして再インポートします。共有ディスクグループに対してサポートされているレイアウトにボリュームを再構成し、いったんデポートした後、共有グループとして再インポートします。

3

Storage Foundation Cluster File System の管理

Veritas Storage Foundation Cluster File System は、複数のホストが同じファイルを同時にマウントし、そのファイルに対して同時にファイル操作を実行できるようにする共有ファイルシステムです。クラスタ設定で SFCFS を操作するには、Veritas Storage Foundation Cluster File System に含まれる一連の統合 Veritas 製品が必要です。

SFCFS でクラスタを設定するには Veritas Cluster Server（VCS）が必要です。VCS は SFCFS に不可欠な 2 つの主要コンポーネントを提供します。LLT パッケージはノード間通信ができるようにし、ネットワーク通信を監視します。GAB パッケージはクラスタの状態、設定、メンバーシップサービスを提供し、システム間のハートビートリンクを監視してシステムがアクティブであることを保証します。SFCFS HA をインストールする場合、VCS によって提供されるその他複数のパッケージがあり、アプリケーションのフェールオーバーサポートが提供されます。

『Veritas Storage Foundation Cluster File System インストールガイド』を参照してください。

SFCFS では、クラスタファイルシステムをマウントするために必要な共有ボリュームを作成するために、Veritas Volume Manager（VxVM）のクラスタ機能（CVM）も必要です。

メモ : クラスタファイルシステムをインストールして管理するには、VxVM の実用知識が必要です。アプリケーションフェールオーバー機能をインストールして管理するには、VCS の実用知識が必要です。これらの製品について詳しくは Veritas Volume Manager と Veritas Cluster Server のマニュアルを参照してください。Volume Manager のユーザーズガイドは、Storage Foundation パッケージのインストール後に /opt/VRTSvmdoc ディレクトリに置かれます。VCS のユーザーズガイドは、Storage Foundation Cluster File System HA パッケージのインストール後に /opt/VRTSvcsdc ディレクトリに置かれます。

この章では、次の内容について説明します。

- VCS の概要
- CVM の概要
- SFCFS の概要
- SFCFS の管理
- SFCFS でのスナップショット

Veritas Cluster Server の概要

Veritas Cluster Server により、クラスタの作成に必須となる通信、設定およびメンバーシップサービスを実行できます。VCS は、クラスタファイルシステムを設定する際に最初にインストールおよび設定するコンポーネントです。

GAB と LLT は、イーサネットデータリンクまたはファイバーチャネルファブリックに直接実装する VCS 固有のプロトコルです。GAB と LLT はともに、クラスタ内のすべてのサーバーを接続する冗長データリンク上で実行されます。VCS では、単一の通信リンクの障害が原因でクラスタが機能停止するのを防ぐための冗長クラスタ通信リンクが必要です。

Veritas Cluster Server のメッセージング - GAB

GAB はメンバーシップサービスとメッセージングサービスを提供します。これらのサービスはともに、クラスタ全体と、クラスタを実行するアプリケーショングループを対象としたものです。GAB のメンバーシップサービスにより、クラスタの起動と停止が正しい順序で実行されます。

GAB を設定するには、/etc/gabtab ファイルを使います。設定を完了するには gabconfig コマンドを使います。たとえば、-n コマンドオプションで、クラスタのノード数を指定します。GAB は VCS インストールスクリプトの実行時に自動的に設定されますが、クラスタにノードを追加するときは GAB を再設定する必要があります。

gabconfig（1M）のマニュアルページを参照してください。

Veritas Cluster Server の通信 - LLT

LLT により、カーネル間の通信が可能になり、ネットワーク通信が監視されます。/etc/llthosts および /etc/lltab の各 LLT ファイルを設定して、クラスタ内のシステム ID の設定、複数のクラスタ ID の設定、ハートビートの頻度などのネットワークパラメータのチューニングを実行できます。LLT により、状態変化などのイベントを即座に反映し、高速な応答が可能になります。

GAB と同様に、LLT は VCS のインストールスクリプトの実行時に自動的に設定されます。/etc/lltab ファイルには、インストール時に入力した情報に基づく構成情報が含まれます。クラスタにノードを追加するときに、LLT の再設定が必要になる場合があります。

lltab（4）のマニュアルページを参照してください。

Veritas Volume Manager のクラスタ機能の概要

Veritas Volume Manager のクラスタ機能（CVM）により、VxVM の制御下にある一連の論理デバイスに複数のホストから同時にアクセスし、管理することができます。VxVM クラスタはデバイスの集合を共有するホストの集合であり、各ホストはクラスタ内のノードです。これらのノードは、1 つのネットワークを介して接続されます。1 つのノードに障害が発生しても、他のノードは引き続きデバイスにアクセスできます。VxVM クラスタ機能により、すべてのノードで、変更などのデバイス設定が同じ論理ビューとして表示されます。

VCS によってクラスタの設定がセットアップされたら、CVM 共有ストレージを設定します。

61 ページの「Veritas Volume Manager のクラスタ機能の管理」を参照してください。

『Veritas Volume Manager 管理者ガイド』を参照してください。

Storage Foundation Cluster File System の概要

ファイルシステムクラスタは 1 つのプライマリと 31 個までのセカンダリで構成されます。プライマリとセカンダリの用語は、特定のノード（またはハードウェアプラットフォーム）ではなく 1 つのファイルシステムに適用されます。そのため同一のクラスタノードを、ある共有ファイルシステムではプライマリにし、同時に別の共有ファイルシステムではセカンダリにすることができます。そのようにファイルシステムのプライマリシップを分散させて、クラスタ上の負荷を分散させる管理方針が推奨されます。

40 ページの「クラスタ内の負荷分散」を参照してください。

CVM の場合、単一のクラスタノードがクラスタ内のすべての共有ディスクグループと共有ボリュームのマスターになります。

クラスタマウントと共有マウント

`mount -o cluster` オプション付きでマウントされた VxFS ファイルシステムは、ローカルマウント（非共有マウント）ではなく、クラスタマウント（共有マウント）です。共有モードでマウントされたファイルシステムはクラスタ環境で VxVM 共有ボリュームに配置される必要があります。ローカルマウントを共有モードで再マウントしたり、共有マウントをローカルモードで再マウントすることはできません。クラスタ内のファイルシステムは、異なる読み取りおよび書き込みオプションでマウントできます。このようなファイルシステムは非対称マウントと呼ばれます。

Storage Foundation Cluster File System プライマリと Storage Foundation Cluster File System セカンダリ

プライマリノードとセカンダリノードは両方とも、クラスタファイルシステムの自身のメタデータインテントログを処理します。クラスタファイルシステムを最初にマウントしたノードはプライマリノードと呼ばれます。その他のノードはセカンダリノードと呼ばれます。プライマリノードで障害が発生すると、内部選択処理によって、どのセカンダリをプライマリファイルシステムにするかが決定されます。

プライマリシップを確認するには、次のコマンドを使います。

```
# fsclustadm -v showprimary mount_point
```

ノードにプライマリシップを与えるには、次のコマンドを使います。

```
# fsclustadm -v setprimary mount_point
```

非対称マウント

非対称マウントでは、共有ファイルシステムを異なる読み取りおよび書き込み機能でマウントできます。そのため、クラスタ内の 1 つのノードは読み書き両用モードでマウントし、一方で、他のノードは読み取り専用モードでマウントすることができます。

クラスタ読み取りおよび書き込み（crw）オプションは、最初にファイルシステムをマウントするときに非対称マウントを行うことを指定する場合に使います。また、再マウント（mount -o remount）を実行して、オプションを変更する際に、指定することもできます。次の表の左列は、プライマリがマウントされるモードを示します。チェックマークは、セカンダリマウントが使えるモードを示します。

		セカンダリ		
プライマリ		ro	rw	ro、crw
	ro	✔		
	rw		✔	✔
	ro、crw		✔	✔

-o cluster,ro オプションを指定してプライマリをマウントした場合にのみ、セカンダリが異なるモード（読み書き両用モード）でマウントされることを回避できます。rw オプションは、クラスタ全体で読み取りおよび書き込み機能を使う際に指定します。

mount_vxfs（1M）のマニュアルページを参照してください。

Storage Foundation Cluster File System と Veritas Volume Manager のクラスタ機能エージェント

エージェントは、事前定義済みのリソースタイプを管理する VCS プロセスです。SFCFS と CVM は、VCS とやり取りするためにエージェントを必要とします。エージェントには、リソースをオンラインまたはオフラインにする、リソースを監視する、状態の変化を VCS に報告するなどの機能があります。VCS 付属エージェントは VCS の一部であり、VCS のインストール時にインストールされます。SFCFS エージェントと CVM エージェントは VCS のアドオンリソースであり、それぞれ Veritas File System と Veritas Volume Manager に固有です。

Storage Foundation Cluster File System の管理

ここでは、クラスタファイルシステム管理の主な側面と、単一ホストの VxFS 管理との相違点について説明します。

Storage Foundation Cluster File System のコマンド

SFCFS コマンドは次のとおりです。

- `cfscluster` - クラスタ設定コマンド
- `cfsmntadm` - クラスタがマウントされたファイルシステム上でポリシーを追加、削除、変更、設定する
- `cfsdgadm` - 共有ディスクグループをクラスタ設定に追加する、または構成から削除する
- `cfsmount`/`cfsumount` - 共有ボリューム上でクラスタファイルシステムをマウントまたはマウント解除する

Storage Foundation Cluster File System のコマンド

`mount` コマンドと `fsclustadm` コマンドは、クラスタファイルシステムを設定する上でも重要です。

mount

`mount` コマンドで `-o cluster` オプションを使うと、共有ファイルシステムにアクセスできます。

`mount_vxfs`（1M）のマニュアルページを参照してください。

fsclustadm

`fsclustadm` コマンドは、CFS の各種属性を報告します。`fsclustadm` を使ってできる作業には、クラスタ内のプライマリノードの表示と設定、ノード ID とホス

ト名の相互変換、指定されたファイルシステムマウントポイントのクラスタマウントを現在持つ全ノードの一覧表示、マウントがローカルマウントかクラスタマウントかの判別などがあります。fsclustadm コマンドは、ファイルシステムがマウントされているクラスタ内の任意のノードから実行でき、指定されたマウントポイントに対するプライマリの場所を制御できます。

fsclustadm（1M）のマニュアルページを参照してください。

fsadm

fsadm コマンドはプライマリノードまたはセカンダリノードから実行することができます。

fsadm（1M）のマニュアルページを参照してください。

クラスタ環境でのコマンドの安全な実行

共有環境では、データの破損を避けるため、RAW デバイスへ書き込み可能な UNIX コマンドを慎重に使う必要があります。共有 VxVM ボリュームの場合、SFCFS には、fsck や mkfs などの VxFS コマンドにより、クラスタ内の別のノードからマウントされたファイルシステムを誤って破損することを防止するために、クラスタ内のボリュームを専有して他のノードから保護する機能が用意されています。ただし、dd などのコマンドは他のノードから保護せず実行されるため、他のノードからマウントされたファイルシステムを破損する可能性があります。ファイルシステムでこれらの VxFS 以外のコマンドを実行する前に、ファイルシステムがクラスタにマウントされていないことを確認してください。mount コマンドを実行すると、ファイルシステムのマウントが共有か、ローカルかを確認できます。

クラスタファイルシステムの時間の同期

SFCFS では、Network Time Protocol（NTP）デーモンなどの外部コンポーネントを使って、すべてのノードのシステムクロックを同期化する必要があります。ノードを同期化しないと、ファイル作成時のタイムスタンプ（ctime）と同じファイル修正時のタイムスタンプ（mtime）が実際の操作順序と一致しない場合があります。

Storage Foundation Cluster File System の拡張

SFCFS に対するプライマリと同様に、CVM に対するマスターノードが存在します。ファイルシステムを拡張するときは、CVM マスターからボリュームを拡張し、続いて任意の SFCFS ノードからファイルシステムを拡張します。CVM マスターと SFCFS ノードは 2 つの異なるノードであってもかまいません。

クラスタ内のプライマリファイルシステムを確認するには、次のコマンドを入力します。

```
# fsclustadm -v showprimary mount_point
```

現在のノードが CVM マスターノードかどうかを確認するには、次のコマンドを入力します。

```
# vxdctl -c mode
```

ファイルシステムのサイズを実際に増やすには、次の 2 つのコマンドを実行します。マスター CVM ノードで、次のコマンドを入力します。

```
# vxassist -g shared_disk_group growto volume_name newlength
```

SFCFS ノードで、次のコマンドを入力します。

```
# fsadm -F vxfs -b newsize -r device_name mount_point
```

fstab ファイル

vfstab により開始されるマウントはクラスタ設定の開始前に行われるため、/etc/vfstab ファイルではどのクラスタファイルシステムについても起動時のマウントを指定しないでください。クラスタマウントの場合は、再ブートの後に、VCS 設定ファイルを使って有効にするファイルシステムを特定してください。

クラスタ内の負荷分散

クラスタ内の作業負荷の分散により、処理効率およびフェールオーバーの機能が向上します。

たとえば、8 つのファイルシステムと 4 つのノードがある場合、ノードごとに 2 つのファイルシステムをプライマリとして指定すると効果的です。プライマリシップは、どのノードが最初にファイルシステムをマウントするかによって決まります。また、fsclustadm を使って、SFCFS プライマリを指定することもできます。fsclustadm setprimary コマンドでも、現在のプライマリで障害が発生した場合にプライマリシップを引き受ける順序を定義できます。セットアップ後、クラスタ内の 1 つ以上のノードでファイルシステムがマウントされている間、このポリシーは有効です。

GUI の使用

ローカルファイルシステムとクラスタファイルシステムの両方で、ファイルシステムの作成とマウントなどの各種の VxFS 機能を Veritas Enterprise Administrator（VEA）を使って実行できます。

SFCFS HA では、VCS Cluster Manager の GUI を使って SFCFS を設定し、監視できます。VCS の GUI により、LLT イベントと GAB イベントをデバッグするためのログファイルが提供されます。

Storage Foundation Cluster File System でのスナップショット

スナップショットは、VxFS ファイルシステムの PIT（point-in-time）イメージを提供します。スナップショットに読み取り専用モードでマウントされたファイルシステムとしてアクセスすると、ファイルシステムのオンラインバックアップを効率よく実行できます。スナップショットは、スナップファイルシステムでデータブロックが上書きされるとデータブロックを逐次コピーするコピーオンライトセマンティクスを実装しています。

『Veritas File System 管理者ガイド』を参照してください。

クラスタファイルシステムのスナップショットは、クラスタ内の任意のノードから発生する I/O のコピーオンライト機構を拡張します。

クラスタスナップショットの特性

1. マウントしたクラスタファイルシステムのスナップショットは、クラスタ内の任意のノードにマウントできます。マウントしたファイルシステムは、プライマリ、セカンダリ、またはセカンダリ専用にできます。ファイルシステムの安定したイメージを任意のノードからの書き込み用に提供できます。
2. クラスタファイルシステムの複数のスナップショットを、クラスタ内の同じノードまたは異なるノードにマウントできます。
3. スナップショットは、スナップショットをマウントしているノードでのみアクセスできます。スナップショットデバイスを 2 つの異なるノードに同時にマウントすることはできません。
4. スナップショットをマウントするためのデバイスとしてローカルディスクまたは共有ボリュームを使えます。共有ボリュームは、スナップショットマウントによって排他的に使われ、スナップショットがそのデバイスでアクティブである間は、クラスタ内の他のノードから使えません。
5. スナップショットをマウントしているノードでは、スナップショットがマウントされている間はスナップショットを取られたオリジナルのファイルシステムのマウントを解除できません。

6 SFCFS スナップショットは、スナップショットのマウントが解除された場合、またはスナップショットをマウントしているノードに障害が発生した場合、消失します。ただし、他のどのノードがクラスタから離れたり、クラスタに参加したりしても、スナップショットはその影響を受けません。

7 読み取り専用でマウントされたファイルシステムのスナップショットは作成できません。クラスタファイルシステムのスナップショットは、クラスタファイルシステムが crw オプションでマウントされている場合のみ、スナップショットを取得できます。

処理効率の考慮事項

バックアップ用にスナップショットファイルシステムをマウントすると、コピーオンライトを実行し、スナップショットからデータブロックを読み取るためにリソースが使われるため、システム上の負荷が増大します。このような場合、クラスタのスナップショットを使ってオフホストバックアップを行うことができます。オフホストバックアップは、プライマリサーバーでのバックアップアプリケーションによる負荷を軽減します。スナップショット全体のオーバーヘッドに比べて、リモートスナップショットからのオーバーヘッドは小さくて済みます。したがって、比較的負荷の少ないノードからスナップショットをマウントして、バックアップアプリケーションを実行すると、クラスタ全体の処理効率に効果があります。

Storage Foundation Cluster File System でのスナップショットの作成

次の例は、SFCFS 管理インターフェースのコマンドを使って 2 ノードクラスタでスナップショットを作成し、マウントする方法を示しています。

クラスタファイルシステムでスナップショットを作成するには

1 共有 VxVM ボリューム上で VxFS ファイルシステムを作成します。

```
# mkfs -F vxfs /dev/vx/rdsk/cfsdg/vol1
レイアウトバージョン 7（version 7 layout）
104857600 セクタ、52428800 ブロック（サイズ 1024）、ログサイズ 16384
ブロック（104857600 sectors, 52428800 blocks of size 1024,
log size 16384 blocks）
i ノード数無制限、largefiles はサポートしていません
（unlimited inodes, largefiles not supported）
52428800 データブロック、52399152 空き データブロック
（52428800 data blocks, 52399152 free data blocks）
1600 アロケーションユニット（32768 ブロック）、32768 データブロック
（1600 allocation units of 32768 blocks, 32768 data blocks）
```

2 すべてのノード上で（前の例に続いて、system01 と system02 上で）ファイルシステムをマウントします。

```
# cfsmntadm add cfsdg vol1 /mnt1 all=cluster
# cfsmount /mnt1
```

cfsmntadm コマンドで Cluster Manager 構成にエントリを追加してから、cfsmount コマンドですべてのノード上でファイルシステムをマウントします。

3 前に作成したボリューム（この例では snapvol）上で、クラスタマネージャ構成にスナップショットを追加します。

```
# cfsmntadm add snapshot cfsdg snapvol /mnt1 /mnt1snap \
system01=ro
```

メモ: クラスタファイルシステムのスナップショットには、そのスナップショットが作成されたノード上でのみアクセスできます。スナップショットファイルシステム自体はクラスタマウントできません。

---

4 スナップショットをマウントします。

```
# cfsmount /mnt1snap
```

5 スナップされたファイルシステムは、そのスナップショットがすべてマウント解除されるまではマウント解除できません。スナップされたクラスタファイルシステムをマウント解除する前に、スナップショットをマウント解除します。

```
# cfsumount /mnt1snap
```

4

フェンシングの管理

I/O フェンシング

クラスタ設定は、I/O フェンシングが有効な状態で行うことをお勧めします。I/O フェンシングには、SCSI-3 PR（Persistent Reservation）をサポートするための共有デバイスが必要です。I/O フェンシングを有効にすると、スプリットブレインによってデータが破損する可能性を防ぐことができます。

Veritas Storage Foundation Cluster File System は、I/O フェンシングを有効にしていない状態でサポートされています。ただし、I/O フェンシングを有効にしないと、スプリットブレイン状態になりデータが破損する場合があります。

I/O フェンシングは、アクティブなクラスタのメンバーへの書き込みアクセスを許可し、メンバー以外へのアクセスをブロックします。I/O フェンシングの物理コンポーネントは、データディスクとコーディネータディスクです。それぞれが固有の目的を持ち、異なる物理ディスクデバイスを使います。

『Veritas Cluster Server インストールガイド』を参照してください。

http://support.veritas.com/docs/283161 のハードウェア互換性リスト（HCL）を参照してください。

データディスク

データディスクは、データストレージ用に使われる標準ディスクデバイスです。物理ディスクまたは RAID 論理ユニット（LUN）を使えます。これらのディスクは、SCSI-3 PGR をサポートする必要があります。データディスクは、標準の VxVM および CVM ディスクグループに組み込まれます。CVM がディスクグループ単位でデータディスクを遮蔽する役割を持ちます。VxVM は I/O フェンシングを有効にするので、その他のいくつかの機能が提供されます。グループに追加されたディスクは、デバイスへの新しいパスなので、自動的に遮蔽されます。

コーディネータディスク

コーディネータディスクは、特殊な目的を持つディスクです。コーディネータディスクは 3 つ（または 3 より大きい奇数）の標準ディスクまたは LUN で構成され、クラスタの再設定時に I/O フェンシングのために使うように別に用意されています。

コーディネータディスクは、クラスタの再設定時にグローバルなロック装置として機能します。このロック機構により、他のノードからデータドライブを遮蔽するノードが決定されます。別の視点から見ると、システムは、データドライブからピアを遮蔽する前に、コーディネータディスクからそのピアの登録を解除する必要があります。コーディネータディスクの制御に関するこのような「競合」の概念は、スプリットブレインの回避におけるフェンシングの役割を理解するために重要です。

コーディネータディスクは他の用途には使えません。データを保存することも、ユーザーデータ用のディスクグループに入れることもできません。コーディネータディスクには、SCSI-3 PGR をサポートする任意の 3 つのディスクを使えます。コーディネータディスクには、最小限の LUN を使うことをお勧めします。コーディネータディスクはデータを保存しないので、クラスタノードは、コーディネータディスクに登録するだけです。コーディネータディスクを予約する必要はありません。

コーディネータディスクを設定する前の注意

I/O フェンシングが機能するには、各クラスタシステムがアクセスできるディスクグループ内にコーディネータディスクが設定されている必要があります。コーディネータディスクを使うことにより、`vxfen` ドライバが潜在的なスプリットブレイン状態を解決し、データの破損を防止することができます。コーディネータディスクがデータストレージとして使われることはありません。そのため、余分な領域が消費されないように、ディスクアレイでは最小限の LUN を使って設定することをお勧めします。

コーディネータディスクの必要条件は次のとおりです。

- ✔ 3 つ以上のコーディネータディスクが存在し、コーディネータディスクの合計数が奇数である必要があります。これは、ディスクの定足数を確実に計算できるようにするためです。
- ✔ 各コーディネータディスクは物理的に別のディスクまたは LUN を使う必要があります。
- ✔ 可能な場合、各コーディネータディスクは別のディスクアレイに置いてください。
- ✔ ディスクアレイ内のコーディネータディスクはハードウェアベースのミラー化を使う必要があります。

✔ コーディネータディスクは SCSI-3 PR をサポートする必要があります。SCSI-3 PR のサポートのテストを行う vxfentsthdw ユーティリティの使用には、1 MB 以上のディスクが必要であることに注意してください。それより容量が小さいディスクは手動でテストできます。手順については、Veritas テクニカルサポート（http://support.veritas.com）までお問い合わせください。

コーディネータディスクのディスクグループの設定

コーディネータディスクとして使う予定のディスクをすでに追加および初期化済みの場合は、次の手順 4 から開始できます。

コーディネータディスクとして使うディスクグループを設定するには

1 3 つのディスクをコーディネータディスクとして使うために物理的に追加します。すべてのクラスタノードは、これらを物理的に共有する必要があります。余分な領域が消費されないように最小サイズのディスクまたは LUN を使うことをお勧めします。

2 必要に応じて、vxdisk scandisks コマンドを使ってディスクドライブとその属性をスキャンしてください。このコマンドは VxVS デバイスのリストを更新し、新しいデバイスで DMP を設定します。次に例を示します。

```
# vxdisk scandisks
```

3 vxdisksetup コマンドでディスクを VxVM ディスクとして初期化します。次のコマンド例では、CDS 形式を指定します。

```
# vxdisksetup -i vxvm_device_name format=cdsdisk
```

次に例を示します。

```
# vxdisksetup -i /dev/rdsk/c2t0d2s2 format=cdsdisk
```

コーディネータディスクとして使う予定の各ディスクに対してこのコマンドを実行します。

4 1 つのノードで、vxfencoorddg という名前でディスクグループを作成します。このグループには、奇数の数のディスクまたは LUN で構成される、3 つ以上のディスクが含まれている必要があります。コーディネータディスクとして使うディスクは 3 つのみとし、ディスク領域を占有しないように最小サイズのディスクまたは LUN を使うことをお勧めします。

さらに、ディスクに c1t1d0、c2t1d0、c3t1d0 という名前のデバイスがあるとします。

5 任意のノードで、このディスクのいずれかのデバイス名を指定してディスクグループを作成します。

```
# vxdg -o coordinator=on init vxfencoorddg c1t1d0
```

6 ディスクグループに他の 2 つのディスクを追加します。

```
# vxdg -g vxfencoorddg adddisk c2t1d0
# vxdg -g vxfencoorddg adddisk c3t1d0
```

『Veritas Volume Manager 管理者ガイド』を参照してください。

コーディネータディスクグループを確認する際の必要条件

vxfentsthdw ユーティリティの実行

SCSI-3 のサポートを確認するための次のガイドラインを確認します。

- このユーティリティでは、2 種類のシステムからコーディネータディスクにアクセスできることが必要条件です。たとえば、4 システムクラスタの場合は、テスト用に 2 つのシステムを選択します。
- クラスタノードに ssh（SSH クライアント）を設定している場合、ノード間でパスワード入力と確認なしで ssh コマンドが実行できる場合に限り、VXFENTSTHDW が使えます。
  ssh を設定していない場合は、各ノードに対して、インストール中とディスク検証中に他のノードにアクセスできるようにリモート rsh アクセスを許可してください。各ノードで、/.rhosts ファイルの 1 行目に「+」の文字を指定すると、インストールプログラムを実行するシステムにリモートアクセスできるようになります。リモートアクセスは特定のノードにのみ許可できます。詳しくは /.rhosts ファイルのマニュアルページを参照してください。インストールとディスク検証の処理が終了したら、リモート rsh アクセス権限を削除します。
- ssh はデフォルトですが、rsh は vxfentsthdw -n コマンドを実行した場合にのみ有効になります。
- 両ノードが同じディスクに接続していることをこのテスト中に確認するには、vxfenadm -i *diskpath* コマンドでディスクのシリアル番号を検証します。
- vxfentsthdw ユーティリティには、これ以外にも多数のディスクをテストするのに適したオプションがあります。-r オプションを指定すると、データを破損せずにディスクをテストできます。ディスクグループ（-g）とファイル（-f）に一覧されているディスクをテストするオプションについては、この後で詳しく説明します。

コーディネータディスクグループのテスト

セットアップ後、コーディネータディスクグループをテストします。

コーディネータディスクグループをテストするには

1 いずれかのノードで、次のユーティリティを起動します。

```
# /opt/VRTSvcs/vxfen/bin/vxfentsthdw
```

システム対システムの通信が適切に稼動していることを確認してから、この手順を実行してください。

vxfentsthdw（1M）のマニュアルページを参照してください。

2 概要とディスク上のデータの上書きに関する警告について確認してから、処理の継続を確定し、ノード名を入力します。

3 検査するディスクの名前を入力します。
たとえば、/dev/rdsk/c4t8d0s2 のように入力します。

vxfendg ファイルの作成

セットアップとコーディネータディスクグループのテストが終了したら、使用するための設定を行います。

vxfendg ファイルを作成するには

1 ディスクグループをデポートします。

```
# vxdg deport vxfencoorddg
```

2 ノードが再起動されるときに自動的にインポートしないように、-t オプションを使ってディスクグループをインポートします。

```
# vxdg -t import vxfencoorddg
```

3 ディスクグループをデポートします。この操作によってコーディネータディスクが他の目的で動作することを防ぎます。

```
# vxdg deport vxfencoorddg
```

4 すべてのノードで、次のように入力します。

```
# echo "vxfencoorddg" > /etc/vxfendg
```

「vxfencoorddg」の引用符間にスペースを含めないでください。
このコマンドは、コーディネータディスクグループの名前を含む /etc/vxfendg ファイルを作成します。/etc/vxfendg ファイルの内容に基づいて、rc スクリプトは、システムが再起動されるときに vxfen ドライバによって使われる /etc/vxfentab ファイルを作成します。rc スクリプトは、vxfenconfig コマンドも起動します。このコマンドは、vxfen ドライバを設定して、/etc/vxfentab に記述されているコーディネータディスクを起動して使います。/etc/vxfentab ファイルは生成されるファイルです。このファイルを変更しないでください。

VCS 設定におけるフェンシングの有効化

すべてのクラスタノードで I/O フェンシングを設定したら、サンプルの vxfenmode ファイルを /etc/vxfenmode ファイルにコピーします。

```
# cp /etc/vxfen.d/vxfenmode_scsi3_dmp /etc/vxfenmode
```

フェンシングを有効にするには、様々な操作が必要です。具体的には、VCS 設定ファイル（main.cf）での UseFence 属性の編集、ファイル構文の確認、main.cf の他のノードへのコピー、フェンシングドライバを起動するすべてのノードとフェンシングを有効化した VCS の再ブートです。

I/O フェンシングを有効化するには

1 既存の VCS 設定ファイル、/etc/VRTSvcs/conf/config/main.cf を次のように保存します。

```
# haconf -dump -makero
```

2 次のように、すべてのノード上の VCS を停止します。

```
# hastop -all
```

3 main.cf ファイルのバックアップコピーを作成します。

```
# cd /etc/VRTSvcs/conf/config
# cp main.cf main.orig
```

4 いずれかのノードで、vi または他のテキストエディタを使って main.cf ファイルを編集します。UseFence 属性を追加してその値 SCSI3 を割り当てることで、クラスタ属性のリストを変更します。

```
cluster rac_cluster1 (
    UserNames = { admin = "cDRpdxPmHpzS." }
    Administrators = { admin }
    HacliUserLevel = COMMANDROOT
    CounterInterval = 5
    UseFence = SCSI3
    )
```

5 ファイルを保存して閉じます。

6 /etc/VRTSvcs/conf/config/main.cf の構文を次のように確認します。

```
# hacf -verify /etc/VRTSvcs/conf/config
```

7 rcp または他のユーティリティを使って、galaxy などのノードの VCS 設定ファイルを残りのクラスタノードにコピーします。残りの各ノードで、次のように入力します。

```
# rcp galaxy:/etc/VRTSvcs/conf/config/main.cf
  /etc/VRTSvcs/conf/config
```

8 すべてのノードの /etc/vxfenmode を次のように disabled から scsi3 に変更します。

```
# vxfenmode=scsi3
```

9 次のように、すべてのノード上の VCS を停止します。

```
# hastop -all
```

10 各システムの設定ファイルを使って各ノードを停止してから再起動します。たとえば、次のように入力します。

```
# shutdown -y -i6 -g0
```

I/O フェンシングを適切に停止するには、reboot コマンドではなく shutdown コマンドを使います。

コーディネータディスクの追加と削除

調整ディスクを追加する前に、ディスクが SCSI-3 Persistent Reservation をサポートしていることを確認します。

1 任意のクラスタノードに root としてログインします。

2 コーディネータディスクグループをインポートします。/etc/vxfendg ファイルには、コーディネータディスクを含むディスクグループの名前が記述されています。そのため、次のコマンドを使います。次のように入力します。

```
# vxdg -tfC import 'cat /etc/vxfendg'
```

このコマンドについての説明は、次のとおりです。

-t は、システムが再ブートするまでの間のみ、ディスクグループをインポートすることを指定します。

-f は、強制的にインポートすることを指定します。このオプションは、アクセスできないディスクがある場合に必要です。

-C は、すべてのインポートブロックの削除を指定します。

3 ディスクグループにディスクを追加したり、ディスクグループからディスクを削除するには、vxdiskadm（VxVM ディスク管理ユーティリティ）を使います。

4 ディスクが追加または削除されたら、ディスクグループをデポートします。

```
# vxdg deport 'cat /etc/vxfendg'
```

5 クラスタ内の各システムを再ブートして、コーディネータディスクにアクセスできるようにします。

フェンシング設定の確認

管理者は、vxfenadm コマンドを使ってフェンシング設定のテストとトラブルシューティングを行います。次のコマンドオプションがあります。

-d　現在の I/O フェンシングモードの表示
-g　キーの読み取りと表示
-i　デバイスからの SCSI INQUIRY 情報の読み取り
-m　ディスクへのキーの登録
-n　ディスクへの予約
-p　他のシステムから行った登録キーの削除
-r　予約領域の読み取り
-x　登録キーの削除

登録キーの書式

ディスクグループに関連付けられた VxVM によって定義されたキー構造は、最大 7 バイトです。VxVM がこのキーの先頭にシステム固有の ID を付けると、キーはシステム間で一意になります。したがって、I/O フェンシングで使われるキーは 8 バイトから構成されます。

0	1	2	3	4	5	6	7
ノード ID	VxVM 定義	VxVM 定義	VxVM 定義	VxVM 定義	VxVM 定義	VxVM 定義	VxVM 定義

vxfenadm -g /dev/*device_name* コマンドを使うと、現在ディスクに割り当てられているキーを表示できます。たとえば、ノード ID が 1 のシステムで次のコマンドを入力して、*device_name* ディスクのキーを表示できます。

```
# vxfenadm -g /dev/device_name
SCSI 登録キーを読み込んでいます ... (Reading SCSI Registration
Keys...)
デバイス名 : device_name (Device Name: device_name)
キーの総数 : 1 (Total Number of Keys: 1)
キー [0] : (key[0]:)
キー値 [ 数値形式 ]: 65,80,71,82,48,48,48,48 (Key Value [Numeric
Format]: 65,80,71,82,48,48,48,48)
```

vxfenadm の -g オプションでは、8 バイトからなるキー値が 2 つの形式で表示されます。数値形式の場合、ノード ID を表す最初のバイトは、システム ID に 65 を加えた値です。残りのバイトには、キーの文字の ASCII 値が含まれます。この例では PGR0000 です。次の行で、ノード ID 0 は A で表され、ノード ID 1 は B で表されます。

I/O フェンシングの無効化

次の場合は、フェンシングを無効にする必要があります。

- クラスタが最新の SFCFS スタックに更新され、ストレージが SCSI-3 PGR 機能をサポートしていない場合。
- インストール中にフェンシングをオンにしたが、後で無効にする場合。

デフォルトでは、I/O フェンシングが有効な状態で vxfen ドライバが作動します。コーディネータディスクを削除せずにこの機能を無効にするには、/etc/vxfenmode ファイルを作成する必要があります。さらに、vxfen ドライバを通知し、ドライバを開始および停止するための文字列をこのファイルに記述します。

```
# echo "vxfen_mode=disabled" > /etc/vxfenmode
# /etc/init.d/vxfen stop
# /etc/init.d/vxfen start
```

さらに、フェンシングを後で再び有効にする場合は、/etc/vxfendg ファイルの削除をお勧めします。

様々なイベントにおける I/O フェンシングの動作

次の表は、様々なエラーシナリオにおいて、データの破損を防止するための I/O フェンシングの動作について説明しています。各イベントごとに、対応処置を示しています。

表 4-3

イベント	ノード A: 現象	ノード B: 現象	対応
すべてのプライベートネットワークでエラーが発生する。	ノード A は、コーディネータディスクの定足数を獲得しようとします。 ノード A は、コーディネータディスクの競合に勝利すると、ノード B を共有ディスクの登録から除外して処理を続行します。	ノード B は、コーディネータディスクの定足数を獲得しようとします。 ノード B は、コーディネータディスクの競合に敗退すると、クラスタから自分自身を切り離します。	ノード B がクラスタから切り離されたら、ノード B を復帰させる前にプライベートネットワークを修復します。
前述のイベント後にすべてのプライベートネットワークが再び機能する。	ノード A は処理を続行します。	ノード B はクラッシュしています。単に再起動しただけでは、ノード B はデータディスクに書き込めないため、データベースを起動できません。	プライベートネットワークのリストア後にノード B を再ブートします。

表 4-3

イベント	ノード A: 現象	ノード B: 現象	対応
一方のプライベートネットワークでエラーが発生する。	ノード A は、I/O フェンスのエラーメッセージをコンソールに表示し、処理を続行します。	ノード B は、jeopardy に関するメッセージをコンソールに表示し、処理を続行します。	プライベートネットワークを修復します。ネットワークの修復後に、両方のノードは修復したネットワークを自動的に使います。
ノード A がハングする。	ノード A は、何らかの理由で異常なビジー状態になっているか、カーネルデバッガに制御されています。 ノード A がハング状態またはカーネルデバッガの制御から解放されると、キューイングされていたデータディスクへの書き込みは、ノード A の登録が解除されているため失敗します。ノード A は、自身の登録が解除されたことを示すメッセージを GAB から受信すると、クラスタから自分自身を切り離します。	ノード B はノード A とのハートビートを失い、コーディネータディスクの定足数を獲得しようとします。 ノード B はコーディネータディスクの競合に勝利すると、ノード A の登録を共有ディスクから解除します。	プライベートネットワークの動作を確認し、ノード A を再ブートします。

表 4-3

イベント	ノード A: 現象	ノード B: 現象	対応
ノード A、ノード B の電源およびプライベートネットワークが遮断される。コーディネータディスクおよびデータディスクの電源は維持される。 両方のノードには電源が入り、再ブートされるが、プライベートネットワークは疎通しない。	ノード A が再ブートされ、I/O フェンシングドライバ（vxfen）はノード B がコーディネータディスクに登録されていることを検出します。プライベートネットワークがダウンしているため、ドライバはノード B がクラスタのメンバーとして一覧表示されていることを認識しません。このため、I/O フェンシングデバイスドライバはノード A がクラスタに参加することを拒否します。ノード A のコンソールは次の内容を表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	ノード B が再ブートされ、I/O フェンシングドライバ（vxfen）はノード A がコーディネータディスクに登録されていることを検出します。プライベートネットワークがダウンしているため、ドライバはノード A がクラスタのメンバーとして一覧表示されていることを認識しません。このため、I/O フェンシングデバイスドライバはノード B がクラスタに参加することを拒否します。ノード B のコンソールは次の内容を表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	既存のスプリットブレイン状態を解決するための手順については、「トラブルシューティング」の章の該当する項を参照してください。

表 4-3

イベント	ノード A: 現象	ノード B: 現象	対応
ノード B がダウンしている間にノード A がクラッシュする。ノード B は再起動されるが、ノード A は停止したままである。	ノード A はクラッシュしています。	ノード B が再ブートされ、ノード A がコーディネータディスクに登録されていることを検出します。ドライバは、ノード A がクラスタのメンバーとして一覧表示されていることを認識しません。I/O フェンシングデバイスドライバはコンソールに次のメッセージを表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	既存のスプリットブレイン状態を解決するための手順については、「トラブルシューティング」の章の該当する項を参照してください。
3 つのコーディネータディスクのうちの 2 つを含むディスクアレイの電源が遮断される。 ノード B はクラスタから切り離され、ディスクアレイは電源が切れたままである。	ノード A は、クラスタから切り離されるノードがない限り動作を続行します。 ノード A は、コーディネータディスクの定足数を獲得しようとします。3 つのコーディネータディスクのうち 1 つしか使えないため、ノード A は定足数の確保に失敗します。ノード A はクラスタから自分自身を切り離します。	ノード B は、クラスタから切り離されるノードがない限り動作を続行します。 ノード B はクラスタから切り離されます。	失敗したディスクアレイの電源を入れ、I/O フェンシングドライバを再起動して、ノード A をすべてのコーディネータディスクに登録できるようにします。

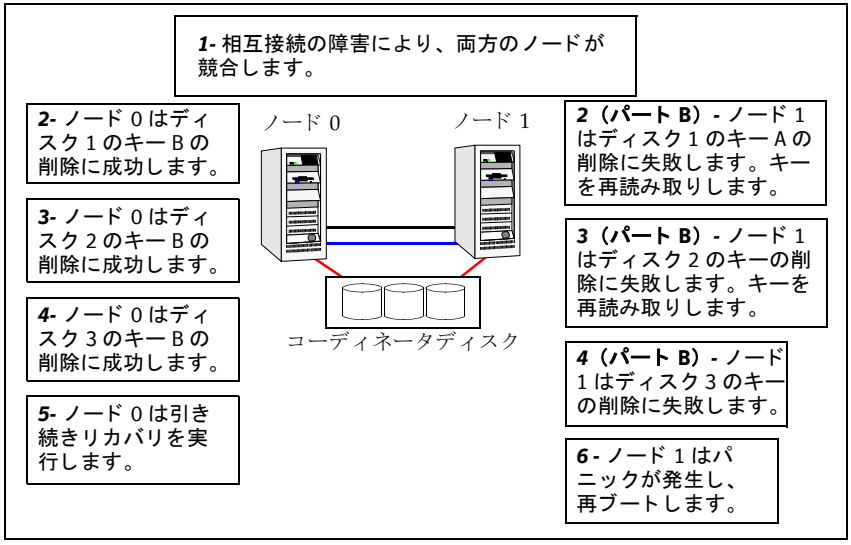
フェンシング設定のトラブルシューティング

次の例は、フェンシングが有効な環境でのネットワークの分割を説明しています。

『Veritas Cluster Server ユーザーズガイド』を参照してください。

既存のネットワーク分割（スプリットブレイン）の例

2 ノードクラスタの相互接続が切断され、クラスタが一時的にスプリットブレイン状態になったところを次の図に示します。



フェンシングモジュールは各システムで同じ処理を行うため、両方のノードは、もう一方のノードに障害が発生したと見なしますが、フェンシング操作を実行して他のノードが削除されているか確認を行います。各ノードの VCS GAB モジュールは、ハートビートの喪失によってピアが停止したと判断し、フェンシングモジュールにメンバーシップの変更を渡します。

各ノードは、コーディネータディスクの制御を取得しようとして競合します。登録済みのノードだけが別のノードの登録を削除できるため、一方のノードだけが各ディスクでコマンドを正常に完了します。

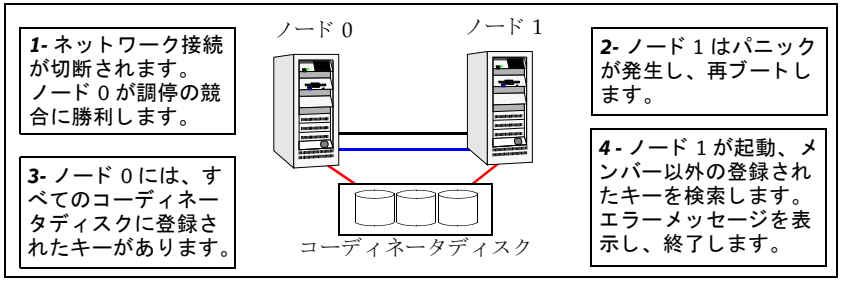
コーディネータディスクの大半からピアを正常に削除したノードが勝利します。次に、勝者側のフェンシングモジュールは、VCS と、フェンシングモジュールに登録されているより高レベルの他のパッケージにメンバーシップの変更を渡し、VCS がリカバリを実行できるようにします。敗者側は、カーネルにパニックが発生して再ブートします。

既存のネットワーク分割（スプリットブレイン）からのリカバリ

フェンシングモジュール vxfen は、ネットワーク分割後のノードの起動と、それによって発生するノードのパニックおよび再ブートを防止します。

シナリオ I

他に同様の症状が発生する状況は、2 ノードクラスタでメンテナンスのために一方のノードを停止した場合です。電源を切っている間は、プライベート相互接続ケーブルが取り外されます。



この状況では、次の動作が行われます。

- ✔ ノード 0 は、ネットワーク障害の後の調停の競合に勝利します。
- ✔ ノード 1 はパニックが発生し、再ブートします。
- ✔ ノード 0 には、コーディネータディスクに登録されたキーがあります。ノード 1 が起動すると、ノード 0 のキーを参照できますが、現在の GAB メンバーシップにノード 0 はありません。一時的な既存のスプリットブレインを検出して、vxfen モジュールがコンソールにエラーメッセージを表示します。vxfen モジュールはフェンシングの起動を防止し、これにより、VCS がオンラインになることも防止されます。
  対処法 : ノード 1 を終了し、ケーブルを再接続してノード 1 を再起動します。

シナリオ II

シナリオ I と同様に、2 ノードクラスタ内でプライベート相互接続ケーブルを取り外すと、ノード 1 はクラスタから切り離され、パニックが発生して再ブートします。プライベート相互接続ケーブルを再接続する前にノード 1 をクラスタに参加させると、ノード 0 は（パニックが発生して）再ブートします。プライベートネットワークが修復されるまで、ノードはデータディスクに書き込むことができません。これは、GAB メンバーシップが形成されず、したがってクラスタも形成されないためです。

対処法 : 両方のノードを終了し、ケーブルを再接続してノードを再起動します。

シナリオ III

シナリオ II と同様に、2 ノードクラスタ内でプライベート相互接続ケーブルを取り外すと、ノード 1 はクラスタから切り離され、パニックが発生して再ブートします。プライベート相互接続ケーブルを再接続する前にノード 1 をクラスタに参加させると、ハードウェア障害のためノード 0 にパニックが発生します。ノード 0 は再起動されず、ノード 1 はクラスタに参加できません。

対処法 : ノード 1 を終了し、ケーブルを再接続してノードを再起動します。次に、コーディネータディスクからノード 0 の登録を解除する必要があります。

1 ノード 1 で、次のように入力します。

```
# /opt/VRTSvcs/vxfen/bin/vxfenclearpre
```

2 ノードを再起動します。

5

Veritas Volume Manager のクラスタ機能の管理

クラスタは、ディスクの集合を共有する多数のホストまたはノードで構成されます。クラスタ設定の主なメリットは次のとおりです。

- 可用性 - 1 つのノードが失敗しても、他のノードは共有ディスクに引き続きアクセスできます。適したソフトウェアが設定されている場合、ミッションクリティカルなアプリケーションは、実行をクラスタのスタンバイノードに転送することによって、動作し続けることができます。冗長ハードウェアに切り替えることで連続的で途切れることのないサービスを提供できるこの機能を、一般にフェールオーバーと呼びます。
  フェールオーバーは、データベースやファイル共有の、ユーザーや上位レベルのアプリケーションに対して透過的です。システムやサービスを監視して、ハードウェアの障害またはソフトウェアのエラーが発生した場合に別のノードでアプリケーションを再開するには、VCS などのクラスタ管理ソフトウェアを設定する必要があります。VCS では、ノードのクラスタへの参加または切り離しなど、一般的な管理タスクの実行もできます。
- オフホスト処理 - クラスタ内のより低負荷のノードでバックアップ、意思決定支援、レポート生成などのアクティビティを実行することで、システムリソースの競合を減少させることができます。これにより、企業はクラスタシステムへの投資以上の価値を導出できます。

CVM を利用すると、クラスタ内の最大 32 個のノードが VxVM 制御下のディスク（VM ディスク）の集合に同時にアクセスし、管理できるようになります。すべてのノード上でディスク構成とその任意の変更の同じ論理表示を利用できます。クラスタ機能が有効なとき、クラスタ内のすべてのノードが VxVM オブジェクトを共有できます。この章では、VxVM で提供されているクラスタ機能について説明します。

VxVM と CVM の詳細情報の参照先は次のとおりです。VxVM マニュアルセットのオンラインバージョンは /opt/VRTS/docs ディレクトリにインストールされています。

『Veritas Volume Manager 管理者ガイド』を参照してください。

VCS の詳細情報の参照先は次のとおりです。VCS マニュアルセットのオンラインバージョンは /opt/VRTS/docs ディレクトリにインストールされています。

『Veritas Cluster Server ユーザーズガイド』を参照してください。

クラスタボリューム管理の概要

企業規模のミッションクリティカルなデータ処理では、密結合されたクラスタシステムがますます一般的になってきました。クラスタの第一のメリットは、ハードウェア障害に対する保護です。プライマリノードに障害が発生、または利用不能になった場合、アプリケーションは、実行をクラスタのスタンバイノードに転送することで、動作し続けることができます。冗長ハードウェアに切り替えることでサービスの連続的な可用性を提供するこの機能を、一般にフェールオーバーと呼びます。

また、クラスタ化されたシステムのもう一つの大きなメリットとして、バックアップ、意思決定支援、レポート生成などのアクティビティにより、システムリソースの競合を減少させる機能があります。このような操作を、サービスの要求に応答している高負荷のノードではなく、クラスタ内の低負荷のノードで実行することにより、クラスタシステムからより一層の価値を導出できます。一部の操作を低負荷のノードで実行するこのような機能を、一般に負荷分散と呼びます。

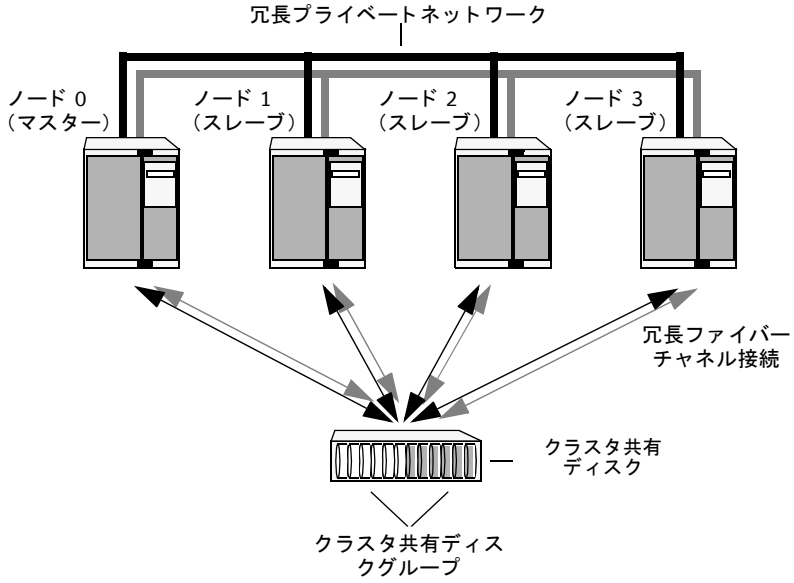
クラスタ機能を実装するために、VxVM はホストのオペレーティングシステムまたは VCS によって提供されるクラスタモニタデーモンと連携します。クラスタモニタは、クラスタメンバーシップの変更を VxVM に通知します。VxVM の各ノードは独自に起動し、クラスタモニタ、OS とクラスタ機能に対応した VxVM のコピーをそれぞれ独自に備えています。あるノードがクラスタに参加すると、そのノードは共有ディスクにアクセスできるようになります。クラスタから切り離されたノードは、それ以降共有ディスクにアクセスできなくなります。クラスタモニタをノード上で起動すると、そのノードはクラスタに参加します。

注意 : VxVM のクラスタ機能は、VxVM と連携して使うように正しく設定されたクラスタモニタと組み合わせて使うときにのみサポートされます。

63 ページの「4 ノードクラスタの例」の図は、類似した、または同じハードウェア特性（CPU、RAM、ホストアダプタ）を持ち、同一のソフトウェア（オペレーティングシステムを含む）で設定された 4 つのノードで構成される単純なクラスタ配置の例です。ノードはプライベートネットワークで互いに接続され、ファイバーチャネル経由で共有外部ストレージ（ディスクアレイまたは JBOD（just a bunch of disks））とも個別に接続されます。各ノードはこれらのディスク（1 つ以上のクラスタ共有ディスクグループで構成される）への 2 つの独立したパスを持ちます。

プライベートネットワークにより、ノードはシステムリソースと互いの状態についての情報を共有できます。プライベートネットワークを使って、すべてのノードから他の現在アクティブなノード、クラスタに参加しているノード、クラスタから切り離されているノード、障害が発生しているノードを認識できます。プライベートネットワークは、チャネルの 1 つで障害が発生した場合に備えた冗長性を確保するために、少なくとも 2 つの通信チャネルを必要とします。1 つのチャネルのみが使われていた場合、その障害はノードの障害と区別が付きません。これは「ネットワーク分割」として知られる状況です。

図 5-2　　4 ノードクラスタの例



クラスタモニタにとって、すべてのノードが等価です。共有ディスクグループ内に設定された VxVM オブジェクトは、潜在的に、クラスタに参加しているすべてのノードからアクセスされる可能性があります。ただし、VxVM のクラスタ機能では、ノードの 1 つがマスターノードとして機能し、クラスタ内の他のノードすべてがスレーブノードとして機能することが要求されます。任意のノードがマスターノードになり、特定の VxVM のアクティビティの調整を担当することができます。

メモ : VxVM オブジェクトを設定または再設定するコマンドは、マスターノード上で実行する必要があります。マスターノードから開始する必要がある作業には、共有ディスクグループの設定、ボリュームの作成と再設定、スナップショット操作の実行などがあります。

VxVM は、クラスタに最初に参加したノードがマスターノードの機能を実行する仕様です。マスターノードがクラスタから切断されると、スレーブノードの 1 つが新しいマスターに選択されます。「4 ノードクラスタの例」では、ノード 0 がマスターノード、ノード 1、2、3 がスレーブノードです。

専有および共有ディスクグループ

2 種類のディスクグループが定義されます。

- 専用ディスクグループ - 1 つのノードにのみ所属します。専用ディスクグループのインポートは、1 つのシステムでのみ行われます。専用ディスクグループ内のディスクには、物理的には 1 つ以上のシステムからアクセスできますが、アクセスは 1 つのシステムに制限されます。ブートディスクグループ（通常は `bootdg` という名前の予約済みディスクグループがエイリアスとして設定される）は、常に専用ディスクグループです。
- 共有ディスクグループ - すべてのノードに共有されます。共有（またはクラスタ共有）ディスクグループは、すべてのクラスタノードにインポートされます。共有ディスクグループ内のディスクは、クラスタに参加可能なすべてのシステムから、物理的にアクセスできる必要があります。

クラスタでは、大多数のディスクグループが共有されます。共有ディスクグループ内のディスクは、クラスタ内のすべてのノードからアクセスでき、複数のクラスタノード上のアプリケーションが同じディスクに同時にアクセスすることもできます。共有ディスクグループ内のボリュームは、ライセンスおよびディスクグループアクティブ化モードによる制約の範囲内で、クラスタ内の複数のノードから同時にアクセスできます。

`vxdg` コマンドを使って、ディスクグループをクラスタ共有に設定することができます。

『Veritas Volume Manager 管理者ガイド』を参照してください。

ディスクグループを 1 つのノードに対してクラスタ共有としてインポートすると、各ディスクヘッダーにはクラスタ ID が設定されます。各ノードを順次クラスタに参加していくと、ディスクグループをクラスタ共有であると認識しインポートします。共有ディスクのインポートまたはデポートを随時実行できます。この操作は、すべてのノードに対して分散方式で実行されます。

各物理ディスクは、一意のディスク ID でマークされます。マスターノード上で VxVM のクラスタ機能を起動すると、マスターノードはすべての共有ディスクグループを（noautoimport 属性セットが設定されている場合を除き）インポートします。スレーブノードがクラスタへの参加を試みると、マスターノードはインポートされているディスク ID の一覧をスレーブノードに送信し、スレーブノードはそのすべてにアクセスできるかをチェックします。スレーブノードからアクセスできないディスクがある場合には、クラスタへの参加は中断されます。一覧内のディスクすべてにアクセスできる場合には、スレーブノードは、マスターノードと同じ共有ディスクグループをインポートして、クラスタに参加します。ノードがクラスタから切断されると、そのノードにインポートされていた共有ディスクグループはすべてデポートされますが、それらのディスクグループは残りのノードからはデポートされません。

共有ディスクグループの再設定は、すべてのノードで協調して実行されます。ディスクグループの設定の変更は、すべてのノード上で同時に実行され、各ノード上の変更は同一になります。これらの変更には原子的な性質があります。これは、この変更がすべてのノード上で同時に発生するか、あるいはまったく発生しないかのどちらかであることを意味します。

クラスタのすべてのメンバーがクラスタ共有ディスクグループへの読み書きアクセスを実行できるかどうかは、「共有ディスクグループのアクティブ化モード」で説明する（クラスタ共有ディスクグループの）アクティブ化モードの設定によって異なります。クラスタ内の少なくとも 1 つのノードがアクティブ状態であれば、クラスタ共有ディスクグループ内に格納されているデータは使用可能です。あるクラスタノードの障害が、残りのアクティブノードのアクセスに影響を与えることはありません。どのノードからアクセスしても、クラスタ共有ディスクグループの設定は同一に見えます。

メモ : 各ノード上で稼動しているアプリケーションは、同時に VM ディスク上のデータにアクセスできます。VxVM は、複数のノードによる共有ボリュームへの同時書き込みに対する保護を行いません。アプリケーション側で（たとえば DLM（Distributed Lock Manager）を使って）整合性を管理することが前提となっています。

共有ディスクグループのアクティブ化モード

共有ディスクグループのアクティブ化は、ノード上のアプリケーションからの I/O に、このディスクグループへのアクセスを許可するため、そのノード上で実行する必要があります。アプリケーションがボリュームに対して読み取りや書き込みができるかどうかは、共有ディスクグループのアクティブ化モードで指定されます。共有ディスクグループの有効なアクティブ化モードには、exclusive-write（排他書き込み）、read-only（読み取り専用）、shared-read（共有読み取り）、shared-write（共有書き込み）、および off（非アクティブ化）があります。アクティブ化モードについては、66 ページの「共有ディスクグループのアクティブ化モード」の表で説明しています。

メモ : 共有ディスクグループのデフォルトのアクティブ化モードは shared-write（共有書き込み）です。

高可用性（High Availability）アプリケーションやオフホストバックアップなどのクラスタの特殊な用途では、ディスクグループのアクティブ化を使って、クラスタ内の個別のノードからのボリュームアクセスを明示的に制御できます。

ディスクグループのアクティブ化モードは、クラスタ内の各ノードからのボリューム I/O を制御します。指定されたノードのディスクグループが、クラスタ内の別のノード上で競合するモードでアクティブにされている場合、そのディスクグループをアクティブにできません。

表 5-4　　共有ディスクグループのアクティブ化モード

アクティブ化モード	説明
exclusive-write（ew）	ノードはディスクグループに対する排他書き込みのアクセス権を持ちます。書き込みアクセスを実行するためにディスクグループをアクティブ化できる他のノードはありません。
read-only（ro）	ノードはディスクグループに対する読み取りのアクセス権を持ち、クラスタ内の他のすべてのノードの書き込みアクセスを拒否します。ノードはディスクグループに対する書き込みのアクセス権は持ちません。書き込みモードのいずれかを他のノード上で実行するためにディスクグループのアクティブ化を試みると失敗します。
shared-read（sr）	ノードはディスクグループに対する読み取りのアクセス権を持ちます。ノードはディスクグループに対する書き込みのアクセス権は持ちませんが、他のノードは書き込みアクセスを取得できます。
shared-write（sw）	ノードはディスクグループに対する書き込みのアクセス権を持ちます。

表 5-4 共有ディスクグループのアクティブ化モード

アクティブ化モード	説明
off	ノードはディスクグループに対する読み取りと書き込みのいずれのアクセス権も持ちません。ディスクグループに対するクエリー操作は許可されます。

次の表は、共有ディスクグループのアクティブ化モードの対応関係の概略を示します。

表 5-5 アクティブ化モードの対応関係

クラスタでのディスクグループのアクティブ化モード	別のノードでディスクグループをアクティブにする際のモード			
	exclusive-write	read-only	shared-read	shared-write
exclusive-write	失敗する	失敗する	成功する	失敗する
read-only	失敗する	成功する	成功する	失敗する
shared-read	成功する	成功する	成功する	成功する
shared-write	失敗する	失敗する	成功する	成功する

共有ディスクグループは、ディスクグループの作成中、手動または自動インポート中に任意のモードで自動的にアクティブ化することができます。共有ディスクグループの自動アクティブ化を制御するには、デフォルトファイル /etc/default/vxdg を作成する必要があります。

デフォルトファイル /etc/default/vxdg には次の行が含まれている必要があります。

```
enable_activation=true
default_activation_mode=activation-mode
```

アクティブ化モードは、exclusive-write（排他書き込み）、read-only（読み取り専用）、shared-read（共有読み取り）、shared-write（共有書き込み）、off（非アクティブ化）のいずれかです。

注意 : デフォルトファイルを使ってアクティブ化を有効にする場合、そのデフォルトファイルをクラスタ内のすべてのノードで同じにすることをお勧めします。そうしないと、アクティブ化の結果が予測不能になります。

共有ディスクグループが作成またはインポートされると、指定したモードにアクティブにされます。ノードがクラスタに参加すると、そのノードからアクセス可能なすべての共有ディスクグループが、指定したモードでアクティブにされます。

vxconfigd デーモンがすでに稼動しているときにデフォルトファイルを編集する場合は、デフォルトファイルへの変更を有効にするために、vxconfigd プロセスを再起動する必要があります。

注意 : デフォルトのアクティブ化モードが off 以外のディスクグループに対して、クラスタ内の別のノードが競合するモードでアクティブにすると、クラスタへの参加、またはディスクグループの作成やインポートに続いて行われるアクティブ化の適用に失敗することがあります。

共有ディスクグループのアクティブ化モードを表示するには、vxdg list *diskgroup* コマンドを使います。

また、vxdg コマンドを使って、共有ディスクグループのアクティブ化モードを変更することもできます。

共有ディスクグループの接続性ポリシー

クラスタ内の各ノードは、ディスクの状態に関して常に合意する必要があります。特に、1 つのノードが指定したディスクに書き込めない場合、書き込み操作の結果が呼び出し元に返される前に、すべてのノードがそのディスクへのアクセスを停止する必要があります。このため、あるノードがディスクにコンタクトできない場合は、別のノードにコンタクトしてディスクの状態をチェックします。ディスクに障害が発生している場合、そのディスクにはどのノードからもアクセスできないためノード間で合意が成立し、ディスクを切断できるようになります。ディスクではなく、いくつかのノードからのアクセスパスに障害が発生している場合、ディスクの状態に関するノード間の合意が成立しません。この種類の不具合を解決するために、次のポリシーのいずれかを適用できます。

- グローバル接続性ポリシーでは、クラスタ内のいずれかのノードがディスク障害を報告した場合は、クラスタ全体で切断を実行します。これはデフォルトのポリシーです。
- ローカル接続性ポリシーでは、ディスクの障害の範囲をその障害を認識した特定のノードに限定します。ディスクの可用性を確認するために、クラスタ内のすべてのノードとの通信を試行し、すべてのノードがそのディスクに関して問題を報告した場合には、クラスタ全体で切断を実行します。

vxdg コマンドを使って、ディスク切断とディスクグループ（dg）障害のポリシーを設定できます。dgfailpolicy では、マスターノードが共有ディスクグループ内の設定とログのコピーへの接続を失った場合の、ディスクグループの障害ポリシーを設定します。この属性では、ディスクグループのバージョンが 120 以上である必要があります。サポートされるポリシーは次のとおりです。

- dgdisable - マスターノードは、すべてのユーザーまたはカーネルに開始されたトランザクションに対してディスクグループを無効にします。最初の書き込みと最後のクローズは失敗します。これはデフォルトのポリシーです。

- leave - ユーザーまたはカーネルに開始されたトランザクション（最初の書き込みと最後のクローズを含む）に対するログの更新に失敗すると、マスターノードにパニックが発生し、ディスクグループは無効になりません。ログコピーへのアクセスの失敗が全体にわたる場合、すべてのノードがマスターノードになり、すべてのノードで順番にパニックが発生します。

ディスクグループの障害ポリシー

設定データベースとログのコピーが含まれるすべてのディスクへのアクセスをマスターノードが失った場合の適切な動作を判別するには、自身によるローカル切断ポリシーだけでは不十分です。このような場合、ディスクグループは無効になります。その結果、クラスタ内の他のノードもこのボリュームへのアクセスを失います。リリース 4.1 では、ディスクグループの障害ポリシーが導入されており、このような場合にマスターノードで必要な動作を判別します。次の表に示すように、このポリシーには 2 種類の設定があります。

表 5-6　　複数の障害ポリシーに対するマスターノードの動作

I/O 障害の種類	Leave（dgfailpolicy=leave）	Disable（dgfailpolicy=dgdisable）
マスターノードはすべてのログのコピーへのアクセスを失います。	マスターノードのパニックが発生し、カーネルによるトランザクションのエラーに対しては［カーネルログの更新に失敗しました（klog update failed）］というメッセージが表示され、ユーザーによるトランザクションのエラーに対しては［cvm config の更新に失敗しました（cvm config update failed）］というメッセージが表示されます。	マスターノードはディスクグループを無効にします。

ディスクグループ障害ポリシーの状態にあるマスターノードの動作は、ディスク切断ポリシーの設定に依存しません。ディスクグループ障害ポリシーが `leave` に設定されていると、通常は起こらない事態である、すべてのノードがログのコピーにアクセスできないパニック状態となります。

共有ディスクグループの制限

メモ: ブートディスクグループ（通常は bootdg がエイリアス）をクラスタ共有にすることはできません。ルートディスクグループは専用ディスクグループである必要があります。

VxVM のクラスタ機能のみを使う場合、RAW デバイスだけがアクセス可能です。Veritas Storage Foundation Cluster File System などの適切なソフトウェアがインストールされ設定されている場合を除き、共有ボリューム内のファイルシステムへの共有アクセスはサポートされていません。

VxVM のクラスタ機能は、RAID 5 ボリュームまたはクラスタ共有ディスクグループに対するタスクの監視をサポートしていません。ただし、これらの機能は、クラスタの特定のノードに接続されている専用ディスクグループに対しては使えます。

専用ディスクグループに共有可能にする必要がある RAID 5 ボリュームがある場合、最初に、このボリュームを stripe-mirror または mirror-stripe などのサポートされているボリュームタイプに再レイアウトする必要があります。オンラインでの再レイアウトは、サポートされていますが、RAID 5 ボリュームは除きます。

サポートされていないオブジェクトが共有ディスクグループに含まれている場合は、そのオブジェクトをデポートしてからクラスタノードのうちの 1 つで専用ディスクグループとして再インポートします。共有ディスクグループに対してサポートされているレイアウトにボリュームを再構成し、いったんデポートした後、共有グループとして再インポートします。

5

Storage Foundation Cluster File System のエージェント

エージェントは、事前定義済みのリソースタイプを管理するプロセスです。エージェントは VCS から設定情報を取得し、それを使ってリソースを起動します。その後、リソースを定期的に監視し、リソースの状態に応じて VCS を更新します。一般的なエージェントは、次の処理を行います。

- リソースをオンラインにする
- リソースをオフラインにする
- リソースを監視し、状態の変化を VCS に報告する

VCS 付属エージェントは VCS の一部であり、VCS のインストール時にインストールされます。クラスタ機能エージェントは、Veritas File System と Veritas Volume Manager（VxVM）に用意されている VCS のアドオンリソースです。クラスタ機能エージェントとリソースタイプは `VRTScavf` パッケージの一部であり、`cfscluster config` コマンドを実行すると設定されます。

『Veritas Cluster Server 付属エージェントリファレンスガイド』を参照してください。

このガイドの PDF バージョンは `/opt/VRTS/docs` ディレクトリにあります。

この付録では、次の内容について説明します。

Storage Foundation Cluster File System エージェントの一覧

SFCFS エージェントには、次のものがあります。

- CFSMount エージェント
- CFSfsckd エージェント
- CVMCluster エージェント
- CVMVolDg エージェント

VCS クラスタコンポーネント

リソース、属性、サービスグループはクラスタ機能に不可欠なコンポーネントです。

『Veritas Cluster Server ユーザーズガイド』を参照してください。

リソース

リソースは、ハードウェアまたはソフトウェアのエンティティです。そのようなエンティティには、ディスク、ボリューム、ファイルシステムマウントポイント、ネットワークインターフェースカード（NIC）、IP アドレス、アプリケーション、データベースなどがあります。各リソースは連携し、クライアント / サーバーモデルのクライアントにサービスを提供するサーバーとして動作します。各リソースタイプは、そのリソースの属性とともに `types.cf` ファイルに定義されています。VCS 設定ファイル `main.cf` には、リソースの属性の値が含まれます。`main.cf` ファイルは、`include` 指示語を使って、`types.cf` に記載されたリソースを組み込みます。

属性

属性には、クラスタ、ノード、サービスグループ、リソース、リソースタイプ、およびエージェントに関する属性値が含まれます。リソースは、特定の属性の指定された値によって設定され、特定の方法で機能します。リソースの属性値を変更すると、VCS エージェントによるリソースの管理方法が変わります。属性は定義と値で構成されます。属性を定義するには、属性のデータ形式と値の種類を指定します。属性には、値が指定されないときに割り当てられるデフォルト値も定義することができます。

サービスグループ

サービスグループは、関連するリソースで構成されます。サービスグループをオンラインにすると、グループ内のすべてのリソースがオンラインになります。

エージェントとエージェントのリソースの変更

VCS Cluster Manager の GUI を使うか、または VCS コマンド（hastatus や haconf などの ha コマンド）をコマンドラインから入力することにより、エージェントによって管理されるリソースの設定を修正できます。main.cf ファイルを直接編集することもできますが、変更内容を有効にするには、システムを再ブートする必要があります。サンプルの main.cf ファイルは /etc/VRTSvcs/conf/sample_cvm ディレクトリにあります。

クラスタファイルシステムのリソース管理には、Veritas Cluster Server の GUI を使うことをお勧めします。

『Veritas Cluster Server インストールガイド』を参照してください。

ファイルシステムクラスタ機能のリソースとサービスグループ

VCS を通してクラスタマウントを管理するには、各種のリソースタイプ、リソース、サービスグループが必要です。Veritas Volume Manager クラスタ機能（CVM）で必要な VCS リソースタイプは次のとおりです。

- CVMCluster
- CVMVolDg

CVMCluster は CVM の操作全体を制御します。CVMCluster のエージェントは CVM クラスタを制御します。VCS 環境で必要な CVMCluster のリソースは 1 つだけです。CVMCluster リソースを追加するときは、CVM の標準的な設定手順を使うことをお勧めします。手順では cvm という名前のサービスグループを作成し、リソースを設定に追加します。

74 ページの「Storage Foundation Cluster File System の管理インターフェース」を参照してください。

SFCFS 機能に必要な VCS リソースタイプは、次のとおりです。

- CFSfsckd
- CFSMount

CFSfsckd は SFCFS 機能に必須のリソースタイプです。CFSfsckd エージェントはクラスタファイルシステムのチェック（fsck コマンド）デーモン vxfsckd を起動します。クラスタマウントを正しく行うには、このデーモンが動作している必要があります。CVMCluster に関しては、CFSfsckd 用に 1 つのリソースインスタンスのみが必要です。これらのリソースを SFCFS 設定プロセスを使って追加します。このプロセスでは、リソースを cvm サービスグループに追加します。

75 ページの「cfscluster コマンド」を参照してください。

各 CVMVolDg リソースが 1 つの共有ディスクグループとそのディスクグループの 1 つ以上の共有ボリュームを制御します。CVMVolDg リソースは、ディスクグループを有効にし、ディスクグループのアクティブ化モードを設定します。各 CFSMount リソースは指定したマウントポイントの共有ボリュームのクラスタマウントを制御します。また、CFSMount リソースは、マウントポイントに関連付けられたオプションも追跡します。

これらのリソースインスタンスは SFCFS 設定中に自動的には追加されません。SFCFS クラスタ管理コマンドを使って、必要に応じて追加する必要があります。

メモ : CFSMount リソースと CVMVolDg リソースは cvm サービスグループには追加されません。これらは別のサービスグループに追加する必要があります。

---

76 ページの「cfsmntadm コマンド」を参照してください。

リソースとサービスグループ間の依存関係

リソースとサービスグループの間の依存関係は、リソースとサービスグループがオンラインおよびオフラインになる順序を指定します。これらは正しい順序で行われる必要があります。SFCFS に必要な各種のリソースとサービスグループはこれらの依存関係（リンク）のルールに従う必要があります。

- CFSMount リソースは対応する CVMVolDg リソースに依存する
- CVMVolDg リソースを含むサービスグループは cvm サービスグループに依存する

Storage Foundation Cluster File System の管理インターフェース

SFCFS 管理インターフェースを使うと、SFCFS に必要なリソースを、正しい属性と属性間の正しいリンクで簡単に作成できます。

Storage Foundation Cluster File System のリソース管理コマンド

クラスタファイルシステムの機能を管理するためには、5 つの VCS エージェントが必要です。これらの各リソースはいくつかの属性と属性間の依存関係を持ちます。リソースの管理を容易にするために、5 つの SFCFS 管理コマンドが提供されています。クラスタファイルシステムの管理にはこれらのコマンドのみを使うことをお勧めします。コマンドは次のとおりです。

- `cfscluster` - クラスタ設定コマンド
- `cfsmntadm` - クラスタマウントされたファイルシステム上でポリシーを追加、削除、変更、設定する
- `cfsdgadm` - 共有ディスクグループをクラスタ設定に追加する、または構成から削除する
- `cfsmount`/`cfsumount` - 共有ボリューム上でクラスタファイルシステムをマウントまたはマウント解除する

cfscluster コマンド

`cfscluster` コマンドは主として、CVM と SFCFS を設定したり設定解除するために使われ、クラスタ内の任意のノードから実行できます。`cfscluster config` コマンドを実行する前に、VCS を起動しておく必要があります。`cfscluster config` コマンドはすべてのリソースタイプ定義を追加し、CVMCluster タイプと CFSfsckd タイプにリソースインスタンスを 1 つずつ追加します。`cfscluster config` コマンドはリソースをオンラインにし、`cfscluster status` コマンドを使って VCS の状態を問い合わせます。

73 ページの「ファイルシステムクラスタ機能のリソースとサービスグループ」を参照してください。

`cfscluster unconfig` コマンドはリソースをオフラインにし（CFSMount リソースを除く）、クラスタファイルシステムを管理するために使われていたすべてのリソースとサービスグループを削除します。

`cfscluster`（1M）のマニュアルページを参照してください。

`cfsumount` コマンドを使って CFSMount リソースを手動でオフラインにしてから、`cfscluster unconfig` コマンドを実行する必要があります。

cfsmntadm コマンド

各クラスタマウントを制御するために、1 つの CVMVolDg リソースと 1 つの CFSMount リソースが必要です。これらのリソースは cfsmntadm add を使って追加できます。cfsmntadm コマンドは、引数にマウントポイント、共有ボリュームおよび共有ディスクグループをとります。任意でサービスグループ名を指定することもできます。サービスグループ名を指定する場合、cfsmntadm コマンドは新しいサービスグループを作成し（そのサービスグループがまだない場合）、cvm サービスグループに依存するように設定します。サービスグループ名を指定しない場合、cfsmntadm add はデフォルトのサービスグループ cfs を作成します。次に、このコマンドは指定されたサービスグループに CVMVolDg を追加し、これを指定されたディスクグループと関連付けます（その種類のリソースが同じサービスグループにまだない場合）。続いて、cfsmntadm add は CFSMount リソースを追加し、これを CVMVolDg リソースとリンクします。その後、リソース属性に適切な値を設定します。すべてのマウントポイント（同じ共有ディスクグループ内にそのデバイスを持つ）を同じサービスグループに追加することをお勧めします。

73 ページの「ファイルシステムクラスタ機能のリソースとサービスグループ」を参照してください。

cfsmntadm を使うと、ファイルシステムスナップショットや Storage Checkpoint の追加、リソースの削除、表示、変更、クラスタマウントされたファイルシステムのプライマリ選択ポリシーの設定を行うこともできます。

cfsmntadm（1M）のマニュアルページを参照してください。

cfsdgadm コマンド

cfsdgadm コマンドは共有ディスクグループの管理インターフェースです。cfsdgadm を使うと、クラスタ設定への共有ディスクグループの追加、共有ディスクグループの削除、アクティブ化モードの変更、共有ディスクグループの設定情報の表示を行うことができます。cfsdgadm で指定する共有ディスクグループは、事前に存在する必要があります。

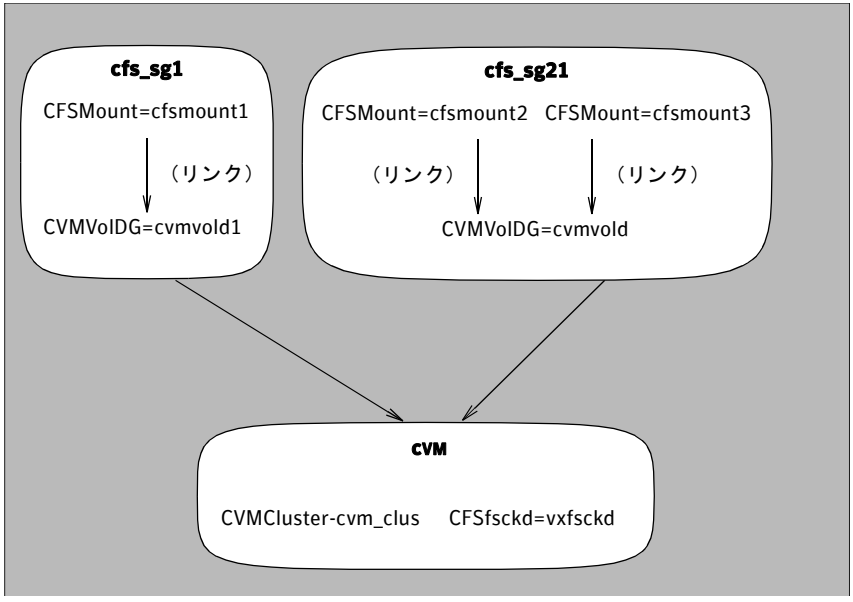
cfsdgadm（1M）のマニュアルページを参照してください。

cfsmount/cfsumount コマンド

cfsmount コマンドは、1 つ以上のノードの共有ボリュームにクラスタファイルシステムをマウントします。ノードを指定しない場合、クラスタファイルシステムは、クラスタ内のすべての関連付けられたノードにマウントされます。cfsmount を実行する前に、cfsmntadm add コマンドを使って、共有ボリュームのクラスタマウントインスタンスを定義しておく必要があります。cfsumount コマンドは 1 つ以上の共有ボリュームをマウント解除します。

cfsmount（1M）のマニュアルページを参照してください。

図 5-3　SFCFS サービスグループとリソース間の依存関係



main.cf ファイルの例

```
include "types.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"

cluster cfs_cluster (
    UserNames = { admin = HMNfMHmJNiNNlVNhMK }
    Administrators = { admin }
    CredRenewFrequency = 0
    HacliUserLevel = COMMANDROOT
    CounterInterval = 5
    )

system system01 (
    )

system system02 (
    )

group cvm (
    SystemList = { system01 = 0, system02 = 1 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { system01, system02 }
    )

    CFSfsckd vxfsckd (
        ActivationMode @system01 = { cfsdg = off }
        ActivationMode @system02 = { cfsdg = off }
        )

    CVMCluster cvm_clus (
        CVMClustName = omcluster
        CVMNodeId = { system01 = 0, system02 = 1 }
        CVMTransport = gab
        CVMTimeout = 200
        )

    CVMVxconfigd cvm_vxconfigd (
        Critical = 0
        CVMVxconfigdArgs = { syslog }
        )

    cvm_clus requires cvm_vxconfigd
    vxfsckd requires cvm_clus


    // resource dependency tree
    //
    //  group cvm
```

```
    //  {
    //  CFSfsckd vxfsckd
    //      {
    //      CVMCluster cvm_clus
    //          {
    //          CVMVxconfigd cvm_vxconfigd
    //          }
    //      }
    //  }


group vrts_vea_cfs_int_cfsmount1 (
    SystemList = { system01 = 0, system02 = 1 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { system01, system02 }
    )

    CFSMount cfsmount1 (
        Critical = 0
        MountPoint = "/mnt0"
        BlockDevice = "/dev/vx/dsk/cfsdg/vol1"
        NodeList = { system01 , system02 }
        RemountRes @system01 = DONE
        RemountRes @system02 = DONE
        )

    CVMVolDg cvmvoldg1 (
        Critical = 0
        CVMDiskGroup = cfsdg
        CVMActivation @system01 = off
        CVMActivation @system02 = off
        )

    requires group cvm online local firm
    cfsmount1 requires cvmvoldg1


    // resource dependency tree
    //
    //  group vrts_vea_cfs_int_cfsmount1
    //  {
    //  CFSMount cfsmount1
    //      {
    //      CVMVolDg cvmvoldg1
    //      }
    //  }
```

CVMTypes.cf ファイルの例

```
type CVMCluster (
    static int NumThreads = 1
    static int OnlineRetryLimit = 2
    static int OnlineTimeout = 400
    static str ArgList[] = { CVMTransport, CVMClustName,
CVMNodeAddr, CVMNodeId, PortConfigd, PortKmsgd, CVMTimeout }
    str CVMClustName
    str CVMNodeAddr{}
    str CVMNodeId{}
    str CVMTransport
    int PortConfigd
    int PortKmsgd
    int CVMTimeout
)

type CVMVolDg (
    static keylist RegList = { CVMActivation }
    static str ArgList[] = { CVMDiskGroup, CVMVolume, CVMActivation
}
    str CVMDiskGroup
    keylist CVMVolume
    str CVMActivation
    temp int voldg_stat
)

type CVMVxconfigd (
    static int FaultOnMonitorTimeouts = 2
    static int RestartLimit = 5
    static str ArgList[] = { CVMVxconfigdArgs }
    static str Operations = OnOnly
    keylist CVMVxconfigdArgs
)
```

CFSTypes.cf ファイルの例

```
type CFSMount (
    static keylist RegList = { MountOpt, Policy, NodeList, ForceOff,
SetPrimary }
    static int FaultOnMonitorTimeouts = 1
    static int OnlineWaitLimit = 1
    static str ArgList[] = { MountPoint, BlockDevice, MountOpt }
    str MountPoint
    str MountType
    str BlockDevice
    str MountOpt
    keylist NodeList
    keylist Policy
    temp str Primary
    str SetPrimary
    str RemountRes
    str ForceOff
)

type CFSfsckd (
    static int RestartLimit = 1
    str ActivationMode{}
)
```

CFSMount エージェント

表 5-7 に、CFSMount エージェントの機能を一覧表示します。

表 5-7　　CFSMount エージェント

説明	クラスタファイルシステムのマウントポイントをオンラインまたはオフラインにし、監視します。CFSMount エージェントの実行可能ファイルは /opt/VRTSvcs/bin/CFSMount/CFSMountAgent です。タイプ定義はファイル /etc/VRTSvcs/conf/config/CFSTypes.cf に格納されます。

表 5-7 CFSMount エージェント

エントリポイント	Online - ブロックデバイスまたはファイルシステムスナップショットをクラスタモードでマウントします。 Offline - ファイルシステムをマウント解除します（必要な場合強制マウント解除を行う）。 Monitor - ファイルシステムがマウントされているかどうかを調べます。fsclustadm コマンドでマウントの状態を確認します。 Clean - CFS をマウントする場合にヌル操作を実行します。 attr_change - 新しいマウントオプションでファイルシステムを再マウントします。ファイルシステムの新しいプライマリを設定します。ファイルシステムに fsclustadm ポリシーを設定します。

必須の属性	データ形式と値の種類	定義
BlockDevice	文字列 - スカラー	マウントポイントに指定するブロックデバイス。
MountPoint	文字列 - スカラー	マウントポイントのディレクトリ。
NodeList	文字列 - キーリスト	マウントするノードのリスト。

オプションの属性	データ形式と値の種類	定義
Policy	文字列 - スカラー	プライマリファイルシステム選択ポリシーのノードリスト

表 5-7　CFSMount エージェント

MountOpt	文字列 - スカラー	mount コマンドのオプション。有効な MountOpt 属性の文字列を作成するには、次の条件を満たす必要があります。 ■ VxFS 固有のオプションのみを使う ■ VxFS 固有のオプションを指定するために -o フラグを使わない ■ -F vxfs ファイルシステムタイプオプションを使わない ■ cluster オプションは必須ではない ■ 次の例のようにカンマ区切りリストでオプションを指定する ro ro,cluster blkclear,mincache=closesync
内部属性		
MountType、Primary、SetPrimary、RemountRes、ForceOff		ユーザー設定ではない - システムのみが使う

タイプの定義

```
type CFSMount (
    static int RestartLimit = 2
    static str LogLevel
    static str ArgList[] = {MountPoint, BlockDevice, MountOpt}
    NameRule = resource.MountPoint
    str MountPoint
    str BlockDevice
    str MountOpt
)
```

設定例

```
CFSMount testdg_test01_fsetpri (
    Critical = 0
    mountPoint = "/mnt1"
    BlockDevice = "/dev/vx/dsk/testdg/test01"
    )

CFSMount testdg_test02_fsetpri (
    Critical = 0
    MountPoint = "/mnt2"
    BlockDevice = "/dev/vx/dsk/testdg/test02"
    MountOpt = "blkclear,mincache=closesync"
    )
```

CFSfsckd エージェント

表 5-8 に、CFSfsckd エージェントの機能を一覧表示します。

表 5-8　CFSfsckd エージェント

説明	vxfsckd プロセスの開始、停止、監視を行います。CFSfsckd エージェントの実行可能ファイルは /opt/VRTSvcs/bin/CFSfsckd/CFSfsckdAgent です。タイプ定義はファイル /etc/VRTSvcs/conf/config/CFSTypes.cf に格納されます。設定は cfscluster config コマンドの実行後に main.cf ファイルに追加されます。
エントリポイント	Online - vxfsckd プロセスを開始します。 Offline - vxfsckd プロセスを強制終了します。 Monitor - vxfsckd プロセスが実行されているかどうかを調べます。 Clean - CFS をマウントする場合にヌル操作を実行します。

必須の属性	データ形式と値の種類	定義
なし		

オプションの属性	データ形式と値の種類	定義
なし		

タイプの定義

```
type CFSfsckd (
    static int RestartLimit = 2
    static str LogLevel
    static str ArgList[] = { }
    NameRule = ""
)
```

設定例

```
CFSfsckd vxfsckd (
)
```

CVMCluster エージェント

表 5-9 に、CFSCluster エージェントの機能を一覧表示します。

表 5-9　CVMCluster エージェント

説明	CVM と関連付けられたクラスタポート上のノードメンバーシップを制御します。CVMCluster リソースは CVMMultiNIC リソースを必要とし、CVMMultiNIC に依存するように設定する必要があります。CVMCluster エージェントの実行可能ファイルは `/opt/VRTSvcs/bin/CVMCluster/CVMClusterAgent` です。タイプ定義はファイル `/etc/VRTSvcs/conf/config/CVMTypes.cf` に格納されます。設定は `cfscluster config` コマンドの実行後に `main.cf` ファイルに追加されます。
エントリポイント	■ Online - CVM クラスタポートにノードを接続します。 ■ Offline - CVM クラスタポートからノードを削除します。 ■ Monitor - ノードの CVM クラスタメンバーシップ状態を監視します。 ■ Clean - CFS をマウントする場合にヌル操作を実行します。

必須の属性	データ形式と値の種類	定義
CVMClustName	文字列 - スカラー	クラスタ名
CVMNodeAddr	文字列 - 関連	ホスト名と IP アドレスのリスト
CVMNodeId	文字列 - 関連	ホスト名と LLT ノード番号のリスト
CVMTransport	文字列 - 関連	CVM トランスポートモード。`gab` または `udp` のどちらか。SFCFS の場合、有効なトランスポートモードは `gab` のみ。
PortConfigd	整数 - スカラー	vxconfigd レベルの通信に CVM が使うポート番号。
PortKmsgd	整数 - スカラー	カーネルレベルの通信に CVM が使うポート番号。
CVMTimeout	整数 - スカラー	クラスタの再構成時に CVM が使うタイムアウト。

タイプの定義

```
type CVMCluster (
    static int NumThreads = 1
    static int OnlineRetryLimit = 2
    static int OnlineTimeout = 400
    static str ArgList[] = { CVMTransport, CVMClustName,
CVMNodeAddr,
                        CVMNodeId,                 PortConfigd,
PortKmsgd, CVMTimeout }
    NameRule = ""
    str CVMClustName
    str CVMNodeAddr{}
    str CVMNodeId{}
    str CVMTransport
    int PortConfigd
    int PortKmsgd
    int CVMTimeout
)
```

設定例

```
CVMCluster cvm_clus (
        Critical = 0
        CVMClustName = vcs
        CVMNodeId = { system01 = 1, system02 = 2 }
        CVMTransport = gab
        CVMTimeout = 200
    )
```

CVMVolDg エージェント

表 5-10 に、CVMVolDg エージェントの機能を一覧表示します。

表 5-10　CVMVolDg エージェント

説明	ディスクグループ内の VxVM 共有ボリュームをオンラインまたはオフラインにし、監視します。CVMVolDg エージェントの実行可能ファイルは /opt/VRTSvcs/bin/CVMVolDg/CVMVolDg です。タイプ定義はファイル /etc/VRTSvcs/conf/config/CVMTypes.cf に格納されます。
エントリポイント	■ Online - 共有ディスクグループのアクティブ化モードを設定し、ボリュームをオンラインにします。 ■ Offline - 共有ディスクグループのアクティブ化モードを off に設定します。 ■ Monitor - ディスクグループとボリュームがオンラインかどうかを調べます。 ■ Clean - CFS をマウントする場合にヌル操作を実行します。 ■ attr_changed - 共有ディスクグループの指定されたアクティブ化モードを変更します。

必須の属性	データ形式と値の種類	定義
CVMDiskGroup	文字列 - スカラー	共有ディスクグループ名
CVMVolume	文字列 - キーリスト	共有ボリューム名
CVMActivation	文字列 - スカラー	ディスクグループのアクティブ化モード。共有書き込み（sw）に設定する必要があります。これはローカライズされた属性です。

タイプの定義

```
type CVMVolDg (
    static keylist RegList = { CVMActivation }
    static str ArgList[] = { CVMDiskGroup, CVMActivation }
    NameRule = ""
    str CVMDiskGroup
    str CVMActivation
)
```

設定例

```
CVMVolDg testdg (
        CVMDiskGroup = testdg
        CVMActivation @system01 = sw
        CVMActivation @system02 = sw
        )
```

用語集

API

アプリケーションプログラミングインターフェース（Application Programming Interface）の略。

BLI バックアップ（Block-Level Incremental Backup）

ファイル全体の保存と取り込みを行わない Veritas のバックアップ機能。その代わりに、前回のバックアップ以降に変更されたデータブロックのみがバックアップされます。

bootdg

ブートディスクグループのエイリアスとして予約済みのディスクグループ名。

CVM

Veritas Volume Manager のクラスタ機能。

CVM マスター（CVM Master）

CVM（Cluster Volume Manager）にはマスターノードがあり、これがボリューム設定に対する変更を記録します。

GB

ギガバイトの略（2^{30} バイトまたは 1024 メガバイト）。

i ノード（inode）

ファイルシステム内部の各ファイルの一意の識別子で、そのファイルと関連付けられたデータとメタデータを含みます。

i ノードアロケーションユニット（inode allocation unit）

特定のファイルセットの i ノード割り当て情報を含む連続するブロックのグループ。この情報はリソースの概略と空き i ノードマップの形式です。

K

キロバイトの略（2^{10} バイトまたは 1024 バイト）。

MB

メガバイトの略（2^{20} バイトまたは 1024 キロバイト）。

MVS

MVS（Multi Volume Support）の略。

MVS ファイルシステム（multi-volume file system）

複数のボリュームにまたがって作成された 1 つのファイルシステム。各ボリュームに独自のプロパティを設定できます。

OLT レプリカ（object location table replica）

データの破損に備えた OLT のコピー。OLT レプリカは、メディア（またはディスク）上の固定位置に書き込まれます。

OLT（Object Location Table）
ファイルシステムの重要な構造要素の位置付けに必要な情報。OLT は、メディア（またはディスク）上の固定位置に書き込まれます。

Quick I/O for Databases
Veritas File System の機能の 1 つ。読み取りまたは書き込みロックを最小限に抑え、データの 2 重バッファリングを解消して、データベースの処理速度を向上させます。この機能では、オンライントランザクションで RAW ディスクデバイスを使った場合と同程度の高速処理を実現し、しかもファイルシステムの管理上の便宜性には影響しません。

Quick I/O ファイル（Quick I/O file）
`::cdev:vxfs:` 拡張子を使ってアクセスされる通常の VxFS ファイル。

SFCFS
Veritas Storage Foundation Cluster File System の略。

SFCFS プライマリ（SFCFS Primary）
クラスタ内の各ファイルシステムには、そのファイルシステムのメタデータの更新を受け持つプライマリノードがあります。

Storage Checkpoint
ファイルシステムまたはデータベースのイメージに対して、一貫して安定したイメージを提供し、最後の Storage Checkpoint 以降に変更されたデータブロックを追跡する機能。

TB
テラバイトの略（2^{40} バイトまたは 1024 GB）。

ufs
UNIX のファイルシステムタイプ。一部のコマンドで、パラメータとして使われます。

UFS
UNIX のファイルシステム。4.2 Berkeley Fast File System から派生。

VCS
Veritas Cluster Server の略。

vxfs
Veritas File System のファイルシステムタイプ。一部のコマンドで、パラメータとして使われます。

VxFS
Veritas File System の略。

VxVM
Veritas Volume Manager の略。

アクセス制御リスト（ACL）（access control list）
特定のユーザーまたはグループと、特定のファイルまたはディレクトリに対するそのアクセス権限を識別する情報。

アロケーションユニット（allocation unit）
ファイルシステム上の連続したブロックのグループ。リソースの概略、空きリソースマップ、データブロックが含まれています。アロケーションユニットには、スーパーブロックのコピーも含まれています。

インテントログ（intent logging）
ファイルシステム構造に対する保留中の変更を記録する方法。これらの変更は、循環型のインテントログファイルに記録されます。

エージェント（agent）
事前定義済みの Veritas Cluster Server（VCS）のリソースタイプを管理するプロセス。エージェントは、リソースをオンラインまたはオフラインにしたり、リソースを監視して VCS に状態の変化を報告します。起動すると、エージェントは VCS から設定情報を取得し、リソースを定期的に監視し、リソースの状態に応じて VCS を更新します。

エクステント（extent）
1 ユニットとして処理されるファイルシステムの連続したデータブロックのグループ。エクステントは、開始ブロックのアドレスと長さで定義されます。

エクステント属性（extent attribute）
ファイルにエクステントを割り当てる方法を決定するポリシー。

外部クォータファイル（external quotas file）
クォータファイル（ファイル名は quotas）。クォータ関連のコマンドを実行するには、ファイルシステムのルートディレクトリにクォータファイルが存在する必要があります。
「クォータファイル」と「内部クォータファイル」を参照してください。

カプセル化（encapsulation）
指定されたディスク上の既存のパーティションをボリュームに変換するプロセス。いずれかのパーティションにファイルシステムが含まれる場合、/etc/vfstab エントリが修正され、代わりにファイルシステムがボリューム上にマウントされます。一部のシステムでは、カプセル化を適用できない場合があります。

間接アドレスエクステント（indirect address extent）
ファイルデータそのものではなく、他のエクステントへの参照を含むエクステント。1 段間接アドレスエクステントでは、複数の間接データエクステントが参照されます。2 段間接アドレスエクステントでは、1 段間接アドレスエクステントが参照されます。

間接データエクステント（indirect data extent）
ファイルデータを含むエクステント。間接アドレスエクステントを介して参照されます。

共有ディスクグループ（shared disk group）
複数のディスクが複数のホストで共有されるディスクグループ（クラスタ共有ディスクグループとも呼ばれます）。

共有ボリューム（shared volume）
共有ディスクグループに属するボリューム。同時に複数のノードで使えます。

クォータ（quotas）
ファイルシステム上のファイルとデータブロックを使うユーザーに対するシステムリソースのクォータ限度。
「ハード制限」と「ソフト制限」を参照してください。

クォータファイル（quotas file）
クォータコマンドにより、外部クォータファイルが読み書きされ、使用制限が取得または変更されます。クォータが有効にされると、クォータ限度が外部クォータファイルから内部クォータファイルにコピーされます。
「クォータ」、「内部クォータファイル」、「外部クォータファイル」を参照してください。

クラスタサービス（Cluster Services）
SFCFS スタックの GAB（Group Atomic Broadcast）モジュールがファイルシステムにクラスタメンバーシップサービスを提供します。LLT により、カーネル間の通信が可能になり、ネットワーク通信が監視されます。
15 ページの「コンポーネント製品の役割」を参照してください。

クラスタマウントされたファイルシステム（cluster mounted file system）
複数のホストが同じファイルをマウントし、そのファイルに対してファイル操作を実行できるようにする共有ファイルシステム。クラスタマウントには、同じファイルシステムの他のクラスタマウントによってアクセス可能な共有ストレージデバイスが必要です。共有デバイスへの書き込みは、クラスタファイルシステムがマウントされた任意のホストから同時に実行できます。ファイルシステムをクラスタマウントにするためには、`mount -o cluster` オプションを使ってファイルシステムをマウントする必要があります。
「ローカルマウントされたファイルシステム」を参照してください。

原子操作（atomic operation）
完全に成功するか、または失敗してすべてが操作開始前の状態のままになる操作。成功すると、すべての操作がただちに有効になります。その変更過程はユーザーには認識されません。操作が一部でも失敗した場合、操作は中止され部分的変更はすべて破棄されます。

構造化ファイルセット（structural fileset）
ファイルシステムの構造を定義するファイル。これらのファイルをユーザーが参照したりアクセスすることはできません。

固定エクステントサイズ（fixed extent size）
ファイルシステムのデフォルトの割り当てポリシーに優先するエクステント属性。特定の固定サイズでファイルの割り当てを設定します。

事前割り当て（preallocation）
ファイルシステムの領域が足りなくなった場合でも、ファイルに対して指定された領域をアプリケーションで確実に使えるようにするための方法。

スーパーブロック（super-block）
ファイルシステムのタイプ、レイアウト、サイズなど、ファイルシステムに関する重要な情報が含まれているブロック。VxFS スーパーブロックは、常にファイルシステムの先頭から 8192 バイトに位置し、長さは 8192 バイトです。

スナップショットファイルシステム（snapshot file system）
マウントされたファイルシステムの特定時点における完全コピー。オンラインバックアップを実行するために使われます。

スナップファイルシステム（snapped file system）
スナップショットファイルシステムの作成にイメージが使われたファイルシステム。

スループット（throughput）
ファイルシステムの場合、特定の時間内に実行できる I/O 操作数。

ソフト制限（soft limit）
ソフト制限はハード制限より低い値になります。ソフト制限は、制限時間内であれば、超過することができます。ファイルとブロックに対して、それぞれ時間が制限されています。
「ハード制限」と「クォータ」を参照してください。

大容量ファイル（large file）
2 TB よりも大きなファイル。VxFS は、サイズが 256 TB までのファイルをサポートします。

大容量ファイルシステム（large file system）
2 TB よりも大きなファイルシステム。VxFS は、サイズが 256 TB までのファイルシステムをサポートします。

遅延（latency）
ファイルシステムの場合、ファイルシステムの特定の操作がユーザーに戻るまでの時間。

直接エクステント（direct extent）
i ノードで直接参照されるエクステント。

データ同期書き込み（data synchronous writes）
同期 I/O の一種。この処理では、ファイルデータは、書き込み操作が戻る前にディスクに書き込まれます。ただし、i ノードのみマーク付けされて後で更新されます。ファイルサイズが変更された場合にのみ、書き込み操作が戻る前に i ノードが書き込まれます。このモードでは、ファイルデータは書き込み操作が戻る前に必ずディスク上に保存されますが、システムがクラッシュすると i ノードの更新が失われる可能性があります。

データブロック（data block）
ファイルとディレクトリに属する実際のデータが含まれるブロック。

ディスカバードダイレクト I/O（discovered direct I/O）
ディスカバードダイレクト I/O はダイレクト I/O と類似しており、データブロックの整列条件に対して同じ制約があります。ただし、格納領域を割り当てる操作、ファイルのサイズを拡張する書き込み操作を実行する際に、i ノード更新の書き込みがアプリケーションに制御を戻す前に実行される必要がありません。

同期書き込み（synchronous writes）
ファイルデータをディスクに書き込み、i ノード最終更新時刻を更新し、更新した i ノードをディスクに書き込む同期 I/O の一種。書き込み操作が呼び出し元に戻ったときには、データと i ノードの両方がディスクに書き込まれています。

トランザクション（transaction）
ファイルシステム構造は、すべての更新を確実に完了させるためにグループ化されています。

内部クォータファイル（internal quotas file）
VxFS では、内部使用のために内部クォータファイルが保持されています。内部クォータファイルには、各ユーザーによって使われるブロックとインデックスの数が記述されています。
「クォータ」と「外部クォータファイル」を参照してください。

ノード（node）
クラスタ内のホストの 1 つ。

ノード参加（node join）
ノードがクラスタに参加し、共有ディスクにアクセスできるようにするプロセス。

ノード停止（node abort）
緊急の場合に、進行中の操作を停止することなくノードがクラスタから切り離される状況。

ハード制限（hard limit）
ファイルシステム上のファイルやデータブロックの使用に関して、各ユーザーのシステムリソースに対する絶対的な制限値。
「クォータ」を参照してください。

ハートビート（Heartbeats）
ハートビートメッセージは、クラスタメンバーシップの変更情報を取得するために、プライベートリンクを介して送信されます。ハートビートメッセージを 16 秒間送信しないノードがある場合、そのノードはメンバーシップから削除されます。コマンド lltconfig を使って、様々なハートビートパラメータに関する情報を取得できます。LLT モジュールがクラスタ内の通信サービスを提供します。

バッファ付き I/O（buffered I/O）
通常、データは、読み取りまたは書き込み操作時に中間メディアのカーネルバッファを経由して、ユーザーバッファとディスク間でコピーされます。同じデータが繰り返し読み取られたり書き込まれる場合、このカーネルバッファはキャッシュとして機能します。そのため、処理速度が向上します。
「非バッファ I/O」と「ダイレクト I/O」を参照してください。

非同期書き込み（asynchronous writes）
遅延された書き込み。この書き込みでは、システムのページキャッシュに存在するページにデータが書き込まれ、書き込み操作が呼び出し元に戻るまでの間にディスクに書き込まれません。そのため、処理効率が向上します。ただし、データがディスクにフラッシュされる前にシステムがクラッシュすると、データを失う可能性があります。

非バッファ I/O（unbuffered I/O）
カーネルキャッシュを回避して、I/O 処理効率を向上させる I/O。ダイレクト I/O に類似しています。ただし、ファイルの拡張時に、ダイレクト I/O では i ノードがディスクに同期して書き込まれますが、非バッファ I/O では i ノード更新が遅延します。
「バッファ付き I/O」と「ダイレクト I/O」を参照してください。

ブートディスク（boot disk）
システムをブートするために使われるディスク。

ブートディスクグループ（boot disk group）
システムがブートされる場合があるディスクが含まれた専用ディスクグループ。

ファイルシステムブロック（file system block）
ファイルシステムの割り当てで基本となる最小サイズ。一部の UNIX ファイルシステムでは、これは断片化サイズと等しくなります。

ファイルセット（fileset）
ファイルシステム内のファイルの集合。

プライマリファイルセット（primary fileset）
ユーザーが参照でき、アクセスできるファイル。

ページファイル（page file）
仮想アドレス空間の固定サイズのブロック。このブロックは、システムで使える物理アドレスにマップされます。

ボリューム（volume）
ファイルシステムやデータベースなど、アプリケーションで使われるディスクブロックのアドレス指定可能な範囲を表す仮想ディスク。

ボリュームセット（volume set）
複数の異なるボリュームのコンテナ。各ボリュームは、独自のジオメトリを所有できます。

ミラー（mirror）
ボリュームとそのボリューム内のデータの複製（サブディスクの順序付けられた集合）。各ミラーは、ミラーが対応付けられているボリュームの 1 つの複製です。

メタデータ（metadata）
ディスク上のファイル属性を記述する構造データ。

予約（reservation）
ファイルに対して領域を事前割り当てするために使われるエクステント属性。

ルートディスクグループ（root disk group）
システム上に常に存在する特別な専用ディスクグループ。ルートディスクグループの名前は `rootdg` です。

連続ファイル（contiguous file）
下位のメディア上でデータブロックが物理的に隣接しているファイル。

ローカルマウントされたファイルシステム（local mounted file system）
単一のホストにマウントされたファイルシステム。単一のホストが他のクライアントからストレージへのすべてのファイルシステム書き込みを調整します。ローカルマウントにするために、`mount -o cluster` オプションを使ってファイルシステムをマウントすることはできません。
「クラスタマウントされたファイルシステム」を参照してください。

索引

T

V

え

お

き

く

こ

さ

し