Red Hat Enterprise Linux 6

High Availability Add-On の概要

Red Hat Enterprise Linux 向け High Availability Add-On の概要
エディッション 6

Last Updated: 2017-10-29

Red Hat Enterprise Linux 6 High Availability Add-On の概要

Red Hat Enterprise Linux 向け High Availability Add-On の概要
エディッション 6

法律上の通知

Copyright © 2014 Red Hat, Inc. and others.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

High Availability Add-On Overview は、Red Hat Enterprise Linux 6 向けの High Availability Add-On の概要を説明しています。

目次

はじめに

本ガイドは、Red Hat Enterprise Linux 6 向けの High Availability Add-On の全体的な概要を説明しています。

本ガイドの内容は概要ですが、Red Hat Enterprise Linux について高度な運用知識があり、サーバーコンピューティングの概念を理解している方を対象としています。

Red Hat Enterprise Linux 6 の使用についての詳細情報は、以下の資料をご参照ください。

- 『Red Hat Enterprise Linux インストールガイド』 — Red Hat Enterprise Linux 6 のインストールに関する情報を提供しています。
- 『Red Hat Enterprise Linux 導入ガイド』 — Red Hat Enterprise Linux 6 の導入、設定、管理に関する情報を提供しています。

Red Hat Enterprise Linux 6 の本製品と関連製品についての詳細情報は、以下の資料をご参照ください。

- 『High Availability Add-On の設定と管理』 は、Red Hat Enterprise Linux 6 向けの High Availability Add-On (Red Hat Cluster という別名でも知られている) の設定と管理について説明しています。
- 『論理ボリュームマネージャの管理』 — 論理ボリュームマネージャ (LVM) について説明しており、クラスター化された環境における LVM の実行に関する情報を含みます。
- 『Global File System 2: 設定と管理』 — Resilient Storage アドオンに含まれている Red Hat GFS2 (Red Hat Global File System 2) のインストール、設定、および維持に関する情報を提供します。
- 『DM Multipath』 — Red Hat Enterprise Linux 6 のデバイスマッパーマルチパスの機能の使用法に関する情報を提供します。
- 『Load Balancer Administration』 — Red Hat Load Balancer アドオン (以前の名称は Linux 仮想サーバー (LVS)) でパフォーマンス性の高いシステム郡およびサービス群を構成する方法について説明しています。
- 『リリースノート』 — Red Hat 製品の現在のリリースに関する情報を提供します。

注記

High Availability Add-On および Red Hat Global File System 2 (GFS2) を使用する Red Hat Enterprise Linux クラスターのデプロイおよびアップグレードのベストプラクティスについての情報は、Red Hat カスタマーポータルに掲載の "Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices" というタイトルの記事を参照してください。 . https://access.redhat.com/kb/docs/DOC-40821

本ガイドおよびその他の Red Hat ドキュメントは、Red Hat Enterprise Linux ドキュメント CD およびオンライン http://access.redhat.com/documentation/docs から HTML、PDF、RPM のバージョンで入手できます。

1. フィードバック

本ガイド内に誤字、脱字を発見された場合や、本ガイドを改善するためのご意見、ご提案がございましたら弊社にぜひご報告ください。その際は、Bugzilla (http://bugzilla.redhat.com/) よりご報告をお願いいたします。製品には **Red Hat Enterprise Linux 6**、コンポーネントには 『doc-High_Availability_Add-On_Overview』、バージョン番号には **6.6** を選択してください。

本ガイドを改善するためのご意見/ご提案をお持ちの場合は、可能な限り具体的にご説明いただくようお願いします。また、エラーを発見された場合は、弊社で対象箇所を容易に見つけることができるように、そのセクション番号と周辺の文章も含めてご報告いただくようお願いします。

第1章 HIGH AVAILABILITY ADD-ON の概要

High Availability Add-On は、基幹実稼働サービスへ信頼性、スケーラビリティーおよび可用性を提供するクラスター化されたシステムです。以下のセクションでは、High Availability Add-On のコンポーネントおよび機能の概要を説明します。

- 「クラスターの基礎」
- 「High Availability Add-On とは」
- 「クラスターインフラストラクチャー」

1.1. クラスターの基礎

クラスターとは、ひとつのタスクを行うために連動して機能する 2 つ以上のコンピューターを指します (*ノード* または *メンバー* と呼ばれる)。クラスターには主に以下の 4 つのタイプがあります。

- ストレージ
- 高可用性
- 負荷分散
- ハイパフォーマンス

ストレージクラスターは、クラスター内の複数のサーバーに対して一貫性のあるファイルシステムイメージを提供し、ひとつの共有ファイルシステムに対する読み書きを同時に行えるようにします。アプリケーションのインストールやパッチの適用を任意のファイルシステムに制限することでストレージの管理を簡略化します。また、クラスター全体に一つのファイルシステムを持たせることでアプリケーションデータのコピーを重複して持たせる必要性が解消されるためバックアップや障害からの復元が簡略化されます。High Availability Add-On では、Red Hat GFS2 (Resilient Storage Add-On の一部) と併用することでストレージのクラスタ化を提供します。

高可用性クラスターは、単一点障害を排除し、ひとつのクラスターノードが動作不能になった場合にサービスを 1 つのクラスターノードから別のクラスターノードにフェイルオーバーすることにより、連続的にサービスの可用性を提供します。通常、高可用性クラスター内のサービスはデータの読み取りや書き込みを行います (read-write でファイルシステムがマウントされる)。従って、1 つのクラスターノードが別のクラスターノードからサービスの制御を引き継ぐ時点で、高可用性クラスターはデータの整合性を維持しなければなりません。高可用性クラスター内のノードの障害はクラスターの外側にあるクライアントからは見えません (高可用性クラスターはフェイルオーバークラスターと呼ばれることもあります)。High Availability Add-On では、高可性サービス管理コンポーネントの **rgmanager** を介して高可用性のクラスタ化を提供します。

負荷分散クラスターは、ネットワークサービス要求を複数のクラスターノードに分配して、クラスターノード間の要求負荷のバランスを取ります。負荷要件に応じてノード数を調節することができるため、対費用効果の高いスケーラビリティーを提供します。負荷分散クラスター内の 1 つのノードが動作不能になると、負荷分散ソフトウェアにより障害が検出され、要求が別のクラスターノードにリダイレクトされます。負荷分散クラスター内のノードの障害は、クラスターの外側にあるクライアントからは見えません。負荷分散機能は Load Balancer Add-On で利用できます。

ハイパフォーマンスクラスターは、クラスターノード群を使用して並列演算を行います。複数のアプリケーションが並行して動作できるため、アプリケーションのパフォーマンスが向上します (ハイパフォーマンスクラスターは演算クラスターまたはグリッドコンピューティングとも呼ばれます)。

注記

前述のクラスタータイプの要約は基本的な設定を説明しています。ニーズに応じて、複数のクラスタータイプを組み合わせる必要がある場合があります。

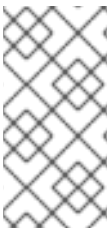
また、Red Hat Enterprise Linux High Availability Add-On に含まれるのは**高可用性サーバーの設定および管理に対するサポートのみ**になります。ハイパフォーマンスクラスターには**対応していません**。

1.2. HIGH AVAILABILITY ADD-ON とは

High Availability Add-On は、パフォーマンス、高可用性、負荷分散、スケーラビリティー、ファイル共有、対費用効果など、ニーズに応じてさまざまな設定で導入できるソフトウェアコンポーネントの統合セットです。

High Availability Add-On は、以下の主要なコンポーネントから構成されています。

- クラスターインフラストラクチャー— 設定ファイルの管理、メンバーシップの管理、ロックの管理、フェンス機能など、複数のノードを 1 つのクラスターとして一緒に動作させるための基本的な機能を提供します。
- 高可用性サービス管理— 1 つのクラスターノードが動作不能になった場合、そのノードから別のノードへのサービスのフェイルオーバーを提供します。
- クラスター管理ツール— High Availability Add-On のセットアップ、設定および管理を行うための設定および管理ツール。これらのツールは、クラスターインフラストラクチャーのコンポーネント、高可用性およびサービス管理のコンポーネントおよびストレージで使用されます。



注記

現時点で完全対応しているのは単一サイトのクラスターのみになります。物理的に離れた複数のサイトに散在するクラスターについては正式には対応していません。複数サイトのクラスターについては Red Hat のセールス担当者またはサポート担当者にご相談ください。

以下のコンポーネントで、High Availability Add-On を補足することができます。

- Red Hat GFS2 (Global File System 2) — Resilient Storage Add-On の一部で、High Availability Add-On で使用するクラスターファイルシステムを提供します。ストレージがあたかも各クラスターノードにローカルに接続されているかのように、複数ノードにブロックレベルでストレージを共有させることができるようになります。GFS2 クラスターファイルシステムを使用する場合はクラスターインフラストラクチャーが必要になります。
- CLVM (Cluster Logical Volume Manager) — Resilient Storage Add-On の一部で、クラスターストレージのボリューム管理を提供します。CLVM に対応させる場合にもクラスターインフラストラクチャーが必要になります。
- Load Balancer Add-On — IP 負荷分散のためルーティングを行うソフトウェアです。Load Balancer Add-On は、仮想サーバー群の背後にある実サーバーにクライアント要求を均等に分配する 2 台 1 組の冗長な仮想サーバーで実行されます。

1.3. クラスターインフラストラクチャー

High Availability Add-On のクラスターインフラストラクチャーは、コンピューター (*ノード*または*メン*

バーと呼ばれる) の集合体が 1 つのクラスターとして連動して機能するための基本的な機能を提供します。クラスターインフラストラクチャーを使用してクラスターを形成した後、ニーズに合わせて他のコンポーネントを使用します (例えば、GFS2 ファイルシステムでファイルを共有するためクラスターをセットアップする、サービスのフェイルオーバーを設定するなど)。クラスターインフラストラクチャーでは次のような機能を実行できます。

- クラスターの管理
- ロックの管理
- 隔離機能
- クラスター設定の管理

第2章 CMAN を使用したクラスター管理

クラスター管理とはクラスターの定足数とメンバーシップの管理を指します。Red Hat Enterprise Linux の High Availability Add-On でクラスター管理を行うのが CMAN (クラスターマネージャーの略) です。CMAN は分散型クラスターマネージャーのためそれぞれのクラスターノードで実行されます。クラスター管理はクラスター内の全ノードに分散されます。

CMAN は他のクラスターノードからのメッセージを監視することでメンバーシップを追跡します。クラスターメンバーシップに変化が生じると、クラスターマネージャーによって他のインフラストラクチャーコンポーネントに対して通知が送られ、各コンポーネントは変化に応じた動作をとることになります。一定時間内にメッセージを送信しないクラスターノードがあると、そのクラスターマネージャーはクラスターからそのノードを取り除き、他のクラスターインフラストラクチャーのコンポーネントにノードがメンバーではないことを通知します。他のクラスターインフラストラクチャーのコンポーネントはノードがクラスターのメンバーではなくなったという通知を受けると、それに応じて実行すべき動作を決定します。例えば、隔離機能の場合ならメンバーではなくなったノードの切断を行います。

CMAN ではクラスターノード数を監視することでクラスターの定足数を追跡します。半数以上のノードが正しく動作している場合、クラスターは定足数を獲得します。半数 (または半数以下) の場合は定足数に満たないためクラスターの全アクティビティが停止されます。同じクラスターの別々のインスタンスが同時に実行してしまう split-brain の発生を防ぐのがクラスター定足数です。split-brain 状態が発生すると、インスタンス同士が互いを認識できない状態でクラスターのリソースにアクセスするため、クラスターの整合性が失われることになります。

2.1. クラスター定足数

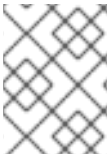
定足数とは CMAN で使用される投票アルゴリズムです。

クラスターが正しく機能するのは互いにメンバーの状態に関して一般的な同意がある場合のみです。大多数のノードが正しく動作し、他の動作中のメンバーと通信、同意している場合、定足数を獲得していると表現します。たとえば、13 台のノードを持つクラスターの場合、7 台以上のノードが通信を行っている場合にのみ定足数に達したと言います。13 台のうち 7 台が停止してしまうと定足数を失ってしまうためクラスターは正しく機能しなくなります。

split-brain の問題を防ぐには定足数を維持する必要があります。定足数を実施していなかった場合、上述の 13 台のノード上で通信エラーが発生すると、13 台のうちの 6 台と別の 6 台が別々に同じ共有ストレージで動作してしまような状態を招く可能性があります。通信エラーのため、2 つに分離されたクラスターの一部同士がディスク領域を上書きするためファイルシステムが破損します。定足数のルールを実施すると、片方のみが共有ストレージを使用するためデータの整合性を確保することができます。

定足数が split-brain 状態を防ぐわけではありませんが、定足数によりクラスター内で優先して動作させるノードが決定されます。split-brain が発生した場合には、複数のノードグループが同時に動作しないよう防ぎます。

定足数はクラスターノード間で行われるイーサネット経由のメッセージ通信で確定されます。オプションでイーサネットに加え **定足数ディスク** も組み合わせたメッセージ通信で確定することもできます。イーサネット経由の定足数の場合、定足数とは単純に過半数になります (全ノード数の 50% + 1)。定足数ディスクを設定する場合は、定足数はユーザーが指定した条件になります。



注記

デフォルトでは各ノード 1 つずつ定足数投票数を獲得します。オプションで複数の投票数を獲得できるよう設定することもできます。

2.1.1. 定足数ディスク

定足数ディスクまた定足数パーティションとはクラスタープロジェクトのコンポーネントで使用するため設定したディスクの任意のセクションです。例を示しながらその目的を説明していきます。

ノード A とノード B があるとします。ノード A はノード B からクラスターマネージャーの「ハートビート」パケットのいくつかを取得できませんでした。ノード A ではパケットを受信できなかった理由は認識できませんが次のような可能性が考えられます。「ノード B に障害が発生した」、「ネットワークのスイッチまたはハブに障害が発生した」、「ノード A のネットワークアダプターに障害が発生した」、「単にノード B の使用率が非常に高いためパケットを送信できなかった」など。クラスターのサイズが非常に大規模な場合、システムの使用率が非常に高い場合、ネットワークが不安定な場合などに発生する可能性があります。

ノード A 側ではパケットを受信できなかった理由、また問題がノード A 自体にあるのかノード B の方にあるのかわかりません。特に 2 台のノードで構成されるクラスターの場合、互いに接続できなくなり、お互いに他方を隔離しようとすることがあるため問題となります。

このため、ノードの隔離を行う前に、接続できないように見えるノードが実際に動作しているかどうかを確認する別の方法を用意しておくのが適切です。定足数ディスクでこれを行うことができます。接続できなくなっているノードを隔離する前に、クラスターソフトウェアを使用すると、そのノードが定足数パーティションにデータの書き込みを行っているかどうかでノードが動作しているかどうかを判断することができます。

2 ノードシステムの場合、定足数ディスクはタイブレーカーとしても機能します。ノードに定足数ディスクおよびネットワークへのアクセスがある場合、2 票獲得としてカウントされます。

ネットワークまたは定足数ディスクに接続できなくなったノードは 1 票失うため、安全に隔離することができるということになります。

定足数ディスクのパラメータの設定方法については 『Cluster Administration (クラスターの管理)』 ガイドにある Conga と **ccs** の管理に関する章を参照してください。

2.1.2. タイブレーカー

タイブレーカーは付加的な解決方法で、同数割れしてしまった場合に定足数に達しているかどうかを隔離を行う前にクラスターパーティションで決定できるようにする方法です。標準的なタイブレーカーの構成は ping ノードとも呼ばれる IP タイブレーカーです。

こうしたタイブレーカーを使用すると、ノード同士の監視に加えクラスター通信を行う際、同じパスの上流に位置するルーターも監視するようになります。2 ノードが互いに接続できなくなった場合、この上流のルーターに ping できるノードが存続することになります。一方、switch -loop などのように 2 ノードいずれも上流のルーターには ping できるのにノード同士の通信はできないため spllit-brain 状態に陥ってしまう場合も考えられます。このため、タイブレーカーを使用している場合でも、隔離機能が正しく設定されているか必ず確認することが重要になります。

さらに、(定足数ディスクと呼ばれることの多い) 共有パーティションから追加詳細が提供される別のタイプのタイブレーカーもあります。clumanager 1.2.x (Red Hat Cluster Suite 3) には、ネットワークがダウンした場合でも両方のノードが共有パーティション経由で通信し続けている限り操作を許可するディスクタイブレーカーが含まれていました。

QDisk (linux-cluster の一部) など、より複雑なスキームを持つタイブレーカーもあります。QDisk では任意の解決法を指定できます。各ノード自体にクラスターへの参加に適した状態であるかを判断させることができます。ただし、これは多くの場合、単純な IP タイブレーカーとして使用されます。詳細は qdisk(5) のマニュアルページをご覧ください。

CMAN にはいくつかの理由によりタイブレーカーは内蔵されていませんが API を使用すれば実装できます。API を使用すると定足数デバイスの登録と更新を行うことができます。サンプル使用例は QDisk のソースコードを参照してください。

以下のような場合、タイブレーカーが必要になるかもしれません。

- 2 ノード構成、フェンスデバイスが接続されクラスター通信で使用するパスとは別のネットワークパス上に配置している
- 2 ノード構成、隔離機能はファブリックレベルで設定 (とくに SCSI 予約の場合)。

ただし、クラスターにネットワークと隔離機能が正しく設定されている場合は、異常なケースを除き、タイブレーカーの使用はさらに複雑になるだけです。

第3章 RGMANAGER

RGManager は、サービスと呼ばれるクラスターリソースやリソースグループまたはリソースツリーなどの集合体に対するフェイルオーバー機能を管理し、これらを提供します。これらのリソースグループはツリー構造に編成されており、親/子の依存関係と継承関係を各サブツリー内に有します。

RGManager では、管理者がクラスターサービスの定義、設定および監視を実行できます。ノードに障害が発生すると、RGManager はクラスター化されたサービスを別のノードに再配置して、サービスの中断を最小限に抑えます。また、**mysql** を別個のノード・セットに制限したり **httpd** を 1 つのノードグループに制限するなど、サービスを特定ノードに制限することもできます。

RGManager は、各種のプロセスとエージェントを組み合わせて使用されます。以下のリストは、それらの分野の概要を示しています。

- フェイルオーバードメイン - RGManager フェイルオーバードメインシステムの機能
- サービスポリシー - Rgmanager のサービス起動とリカバリーポリシー
- リソースツリー - rgmanager のリソースツリーの機能 (起動/停止の順序および継承を含む)
- サービスオペレーションの動作 - rgmanager のオペレーション機能とそれぞれの状態の意味
- 仮想マシンの動作 - rgmanager クラスターで VM を実行する際に留意すべき特記事項
- ResourceActions - RGManager が使用するエージェントのアクションと **cluster.conf** ファイルからそれらの動作をカスタマイズする方法
- イベントスクリプト - rgmanager のフェイルオーバーおよびリカバリーポリシーが環境に適さない場合に、このスクリプトサブシステムを使用して独自にカスタマイズできる。

3.1. フェイルオーバードメイン

フェイルオーバードメインは、サービスがバインドされる可能性のあるメンバーの順序付けられたサブセットです。フェイルオーバードメインは、クラスターのカスタマイズには役立ちますが操作に必須ではありません。

以下に、フェイルオーバードメインの動作に影響を与える複数の設定オプションについて、これらを規定するセマンティクスをリストします。

- Preferred (優先) ノードまたはメンバー: 優先ノードは、オンライン時に所定サービスを実行するように指定されるメンバーです。1 つのメンバーの順序付けや制限のないフェイルオーバードメインを指定すると、この動作をエミュレートすることができます。
- Restricted (制限あり) ドメイン: このドメインにバインドされるサービスは、フェイルオーバードメインのメンバーであるクラスターメンバー上でのみ実行できます。フェイルオーバーのメンバーがいずれも使用できない場合、サービスは停止状態に置かれます。複数メンバーを含むクラスターでは、制限ありのフェイルオーバードメインを使用すると、サービスを実行するすべてのメンバーに同一設定が必要なクラスターサービス (httpd など) の設定が容易になります。クラスターサービスを実行するクラスター全体を設定する代わりに、クラスターサービスに関連した、制限ありフェイルオーバードメイン内のメンバーのみをセットアップすることが必要になります。
- Unrestricted (制限なし) ドメイン: デフォルトの動作。このドメインにバインドされるサービスはすべてのクラスターメンバー上で実行できますが、ドメインのメンバーが利用可能になる場合は常にドメインのメンバー上で実行されます。つまり、Failback (フェイルバック) を設定し

ていない限り、サービスがドメイン外で実行されていてもドメインのメンバーがオンラインになると、サービスはそのメンバーに移行します。

- Ordered (優先順あり) ドメイン: この設定で指定される順序がドメイン内のメンバーの優先順を決定します。ドメインの最もランキングが高いメンバーは、オンラインの場合に常にサービスを実行します。つまり、メンバー A のランクがメンバー B のランクよりも高い場合、A がオフラインからオンラインに移行するとサービスは B で実行されていても A に移行します。

- Unordered (優先順なし) ドメイン: デフォルトの動作。ドメインのメンバーには優先順が設定されません。任意のメンバーがサービスを実行できます。サービスは、可能な場合は常にフェイルオーバードメインのメンバーに移行します (ただし、優先順なしのドメイン)。

- Failback (フェイルバック): 優先順ありフェイルオーバードメインのメンバー上のサービスは、ノード障害の前に元々実行していたノードへフェイルバックする必要があります。これは、失敗を繰り返すノードとフェイルオーバーノード間の頻繁なサービスの切り替えを防ぐのに役立ちます。

Ordering (順序付け)、restriction (制限)、nofailback (フェイルバックなし) はフラグであり、ほぼすべての組み合わせ (ordered+restricted、unordered+unrestricted など) が可能です。これらの組み合わせは、初回の定足数形成後のサービスを起動する場所や、サービスが失敗した場合にどのクラスターメンバーがサービスを引き継ぐかに影響を与えます。

3.1.1. 動作の例

クラスターが以下のメンバーセットから構成されるとします。{A, B, C, D, E, F, G}

優先順あり、制限ありフェイルオーバードメイン {A, B, C}

nofailback の設定解除: サービス「S」は、メンバー「A」がオンラインになり、定足数が設定されている場合に常にメンバー「A」上で実行されます。{A, B, C} のすべてのメンバーがオフラインの場合サービスは実行されません。サービスが「C」上で実行されている場合に「A」がオンラインに移行すると、サービスは「A」に移行します。

nofailback の設定: サービス「S」は、定足数が形成されると最も優先順位の高いクラスターメンバー上で実行されます。{A, B, C} のすべてのメンバーがオフラインの場合サービスは実行されません。サービスが「C」上で実行されている場合に「A」がオンラインに移行すると、サービスは「C」が失敗しない限り「C」上に残ります。「C」が失敗するとその時点で、サービスは「A」にフェイルオーバーします。

優先順なし、制限ありフェイルオーバードメイン {A, B, C}

サービス「S」は、定足数が設定されており、かつ {A, B, C} の 1 つ以上のメンバーがオンラインの場合にのみ実行されます。ドメインの他のメンバーがオンラインに移行してもサービスは再配置されません。

優先順あり、制限なしフェイルオーバードメイン {A, B, C}

nofailback の設定解除: サービス「S」は定足数が設定されている場合に常に実行されます。フェイルオーバードメインのメンバーがオンラインになる場合、サービスは優先順位の最も高いメンバー上で実行されます。そうでない場合は、サービスを実行するためにクラスターのメンバーがランダムに選択されます。つまり、サービス「A」がオンラインの場合は常に「A」で実行され、その後に「B」が続きます。

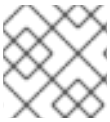
nofailback の設定: サービス「S」は、定足数が設定されている場合に常に実行されます。定足数の形成時にフェイルオーバードメインのメンバーがオンラインになっている場合、サービスはフェイルオーバードメインの優先順位の最も高いメンバー上で実行されます。つまり、「B」がオンラインになっている (「A」はオンラインではない) 場合、サービスは「B」で実行されます。後に「A」がクラスターに入る場合も、サービスは「A」に再配置されません。

優先順なし、制限なしフェイルオーバードメイン {A, B, C}

これは、「優先メンバーのセット」とも言われます。フェイルオーバードメインの 1 つ以上のメンバーがオンラインの場合、サービスはフェイルオーバードメインの不特定のオンラインメンバー上で実行されます。フェイルオーバードメインの別のメンバーがオンラインに移行しても、サービスは再配置されません。

3.2. サービスポリシー

RGManager には、管理者がサービスごとカスタマイズできる 3 つのサービスリカバリーポリシーがあります。



注記

これらのポリシーは、仮想マシンリソースにも適用されます。

3.2.1. 起動ポリシー

デフォルトで、 RGManager は、rgmanager がブートし、かつ定足数が存在する場合にすべてのサービスを起動します。管理者はこの動作を変更することができます。

- autostart (デフォルト) - rgmanager がブートし、定足数が形成される際にサービスを起動します。「0」に設定される場合は、クラスターはサービスを起動せず、代わりにサービスを無効状態にします。

3.2.2. リカバリーポリシー

リカバリーポリシーは、サービスが特定ノードで失敗する場合に rgmanager が取るデフォルトのアクションです。以下のリストで定義されるように 3 つの選択可能なオプションがあります。

- restart (デフォルト) - サービスを同じノードで再起動します。これ以外のリカバリーポリシーが指定されていない場合は、このリカバリーポリシーが使用されます。再起動が失敗すると、rgmanager はサービスを再配置するように切り替えます。
- relocate - クラスター内の他のノード上でサービスの起動を試みます。他のノードがサービスを正常に起動できない場合、サービスは停止状態になります。
- disable - 何も実行しません。サービスは無効状態になります。
- restart-disable - 実行済みのサービスの再起動を試みます。再起動に失敗すると、サービスは無効状態になります。

3.2.3. 再起動ポリシーの拡張

再起動リカバリーポリシーが使用される場合、所定の時間内に実行できる再起動の回数についての最大しきい値を追加で指定できます。これを管理するパラメーターとしてサービス用に利用できる、max_restarts と restart_expire_time という 2 つのパラメーターがあります。

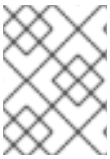
max_restarts パラメーターは、サービスの放棄およびクラスター内の別のホストへの再配置を行う前の再起動の最大回数を指定する整数です。

restart_expire_time パラメーターは、rgmanager に対して再起動イベントの存続期間を指示します。

これらの 2 つのパラメーターを一緒に使用すると、指定期間に許容される再起動の回数についてのスライディングウィンドウを作成できます。以下が例になります。

```
<service name="myservice" max_restarts="3" restart_expire_time="300" ...>
  ...
</service>
```

上記のサービスの許容レベルは、5 分間に 3 回の再起動です。300 秒後に 4 回目のサービスの失敗が発生すると、rgmanager はサービスを再起動せず、代わりにサービスをクラスター内の他の使用可能なホストに再配置します。



注記

両方のパラメーターを一緒に指定する必要があります。どちらか一方のパラメーターの使用方法は単独では定義されません。

3.3. リソースツリーの基礎/定義

以下は、リソースツリーの構造を図示しており、各領域の定義を以下にリストします。

```
<service name="foo" ...>
    <fs name="myfs" ...>
        <script name="script_child"/>
    </fs>
    <ip address="10.1.1.2" .../>
</service>
```

- リソースツリーは、リソース、リソース属性、親/子および兄弟の関係についての XML 表現です。リソースツリーのルートは、ほとんど常にサービスと呼ばれる特殊タイプのリソースになります。通常、リソースツリー、リソースグループおよびサービスはこの wiki では同意のものとして使用されます。rgmanager の観点では、リソースツリーはアトミック単位であり、リソースツリーのすべてのコンポーネントは同じクラスターノード上で起動します。
- fs:myfs と ip:10.1.1.2 は兄弟です。
- fs:myfs は script:script_child の親です。
- script:script_child は fs:myfs の子です。

3.3.1. 親/子関係、依存関係および起動の順序付け

以下のように、リソースツリー内の親/子関係のルールはかなり単純です。

- 親は子の前に開始する。
- 親が停止する前に、すべての子が正常に停止していなければならない。
- 上記の 2 つのルールから、子リソースはその親リソースに依存していると言えます。
- リソースが健全な状態にあると見なされるには、依存するすべての子が健全である必要があります。

3.4. サービスの操作および状態

以下の操作は、仮想マシンにのみ有効な移行操作を除き、サービスと仮想マシンの両方に適用されます。

3.4.1. サービスの操作

サービスの操作は、以下のリストに定義する 5 つの選択可能なアクションのいずれかを適用するためにユーザーが呼び出すことのできる有効なコマンドです。

- enable — サービスを起動します。オプションとして、優先ターゲット上での実行や、フェイルオーバードメインのルールに従った実行が可能です。これらの指定がない場合は、clusvcadm が実行されるローカルホストがサービスを起動します。初回の起動が失敗すると、サービスは relocate 操作が要求されたかのように動作します (以下を参照)。操作が成功すると、サービスは起動状態になります。

- disable — サービスを停止し、無効状態にします。これは、サービスが失敗状態にある場合にのみ許可される操作です。

- relocate — サービスを別のノードに移動します。オプションとして、管理者はサービスを受信する優先ノードを指定できますが、サービスがそのホストで実行できない場合 (例えば、サービスが起動に失敗するか、またはホストがオフラインの場合) でも、再配置は継続し、別のノードが選択されます。Rgmanager は、クラスター内の許可されたすべてのノードでサービスの起動を試みます。クラスター内の許可されたターゲットノードがサービスを起動できない場合、再配置は失敗となり、サービスは元のオーナーで再起動を試行します。元のオーナーがサービスを再起動できない場合、サービスは停止状態になります。

- stop — サービスを停止し、停止状態に切り替わります。

- migrate — 仮想マシンを別のノードに移行します。管理者はターゲットノードを指定する必要があります。移行時の障害の内容によっては、仮想マシンは失敗状態になるか、または元のオーナー上で起動状態になる可能性があります。

3.4.1.1. `freeze` 操作

RGManager はサービスを凍結できます。これにより、ユーザーは rgmanager が管理するサービスのダウンタイムを最小に抑えた上で、システム上の rgmanager、CMAN またはその他のソフトウェアをアップグレードすることができます。

さらに、rgmanager サービスの複数部分のメンテナンスが可能になります。例えば、単一の rgmanager サービスにデータベースと web サーバーがある場合、rgmanager サービスを凍結してから、データベースを停止し、メンテナンスを実行し、データベースを再起動してからサービスの凍結解除を行うことができます。

3.4.1.1.1. 凍結時のサービスの動作

- ステータスチェックが無効になります。

- Start （起動) 操作が無効になります。

- Stop （停止) 操作が無効になります。

- (サービスオーナーの電源オフにしても) フェイルオーバーは起こりません。

重要

以下のガイドラインに従わなかった場合、リソースが複数ホスト上で割り振られる可能性があります。

- rgmanager の再起動の前にホストをリブートする予定でない限りは、サービスの凍結時に rgmanager のすべてのインスタンスを停止しないでください。
- 報告を受けたサービスオーナーがクラスターに再度参加して rgmanager を再起動するまでは、サービスを凍結解除しないでください。

3.4.2. サービスの状態

以下に、RGManager によって管理されるサービスの状態の定義をリストします。

- Disabled — 管理者がサービスを再度有効にするか、またはクラスターが定足数を失う (この時点で autostart パラメーターが評価される) まで、サービスは無効状態のままになります。管理者はこの状態からサービスを有効にすることができます。
- Failed — サービスは停止していると見なされます。リソースの停止操作が失敗すると常にこの状態になります。管理者は、disable (無効) 要求を発行する前に、割り振られたリソース (例えば、マウント済みのファイルシステム) が存在していないことを確認しなければなりません。サービスがこの状態になった時に実行できるのは disable (無効) 操作のみです。
- Stopped — 停止した状態で、次のサービスまたはノードに遷移後にサービスを起動するかどうかが評価されます。これは一時的な状態です。管理者は、この状態からサービスを無効/有効にすることができます。
- Recovering — クラスターはサービスのリカバリーを試行中です。管理者は、必要な場合はリカバリーを防止するためにサービスを無効にすることができます。
- Started — サービスのステータスチェックが失敗した場合、サービスリカバリーポリシーに従ってサービスのリカバリーを行います。サービスを実行するホストが失敗する場合、フェイルオーバードメインおよび排他的なサービスルールに従ってサービスのリカバリーを行います。管理者は、この状態からサービスの停止、無効、および (仮想マシンの場合は) 移行を行うことができます。

注記

`starting` および **`stopping`** などの他の状態は、 **`started`** の状態の特殊な遷移状態です。

3.5. 仮想マシンの動作

RGManager は、VM 以外の他のサービスとは少し異なる方法で仮想マシンを扱います。

3.5.1. 通常の操作

rgmanager が管理する VM は、clusvcadm または他のクラスター対応のツールによってのみ管理する必要があります。動作の大半は通常のサービスに共通するものです。これには以下が含まれます。

- Starting (enabling)
- Stopping (disabling)

- Status monitoring
- Relocation
- Recovery

高可用性の仮想サービスについて詳しくは、*7章 仮想化と高可用性* を参照してください。

3.5.2. 移行

通常のサービス操作のほかにも、仮想マシンは、他のサービスがサポートしていない移行動作をサポートします。移行動作により、クラスター内で仮想マシンの場所を変更するために起動/停止を行う必要がなくなるため、仮想マシンのダウンタイムを最小限に抑えます。

rgmanager がサポートする移行には 2 つのタイプがあり、これらは VM ごとに移行の属性に応じて選択されます。

- live (デフォルト) — 仮想マシンは、ほとんどのメモリーコンテンツが移動先ホストにコピーされる間も継続的に実行されます。これにより、移行時の VM のパフォーマンスと移行完了までにかかる合計時間を犠牲にして VM のアクセス不能状態を最小限に抑えることができます (通常は 1 秒未満)。
- pause - 仮想マシンは、メモリーコンテンツが移動先ホストにコピーされている間にメモリーに凍結されます。これにより、仮想マシンの移行完了までにかかる時間を最小限に抑えることができます。

どの移行スタイルを採用するかは、可用性とパフォーマンスの要件によって変わります。例えば、live (ライブ) 移行の場合、29 秒間のパフォーマンス低下と 1 秒間の完全な使用不可状態が発生する可能性がありますが、pause (一時停止) 移行の場合は 8 秒間の完全な使用不可状態が発生する可能性はあるものの、パフォーマンスの低下は予想されません。

重要

仮想マシンをサービスのコンポーネントとして使用することもありますが、その場合は、すべての移行および以下の大半の便利な機能が無効になります。

さらに KVM を使った移行の使用には、ssh を注意深く設定することが必要です。

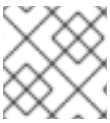
3.5.3. RGManager の仮想マシン機能

以下のセクションでは、RGManager が仮想マシンの管理を容易にする様々な方法を示します。

3.5.3.1. 仮想マシンの追跡

VM がすでに実行されている場合、**clusvcadm** を使って仮想マシンを開始すると、rgmanager はクラスターから VM を検索して VM を見つけると VM に **started** とマーク付けします。

管理者が誤って **virsh** などの非クラスターツールを使ってクラスターノード間で VM を移行した場合、rgmanager はクラスターからVM を検索して VM を見つけると VM に **started** とマーク付けします。



注記

VM が複数の場所で実行されていても、RGManager は警告を出しません。

3.5.3.2. 一時的なドメインのサポート

Rgmanager は、libvirt によってサポートされる一時的な仮想マシンをサポートします。これにより、rgmanager は仮想マシンをオンザフライで作成し、削除できるため、非クラスターツールの使用による仮想マシンの二重起動の可能性を軽減できます。

さらに、一時的な仮想マシンのサポートにより、クラスター化されたファイルシステム上に libvirt の XML 記述ファイルを保管できるため、クラスター全体での **/etc/libvirt/qemu** の同期を手動で維持する必要がなくなります。

3.5.3.2.1. 管理機能

cluster.conf から VM を追加または削除しても、VM の開始または停止は行われません。これにより、rgmanager は VM に注意を向け始めるか、または注意しなくなるかのいずれかになります。

フェイルバック (優先度の高いノードへの移行) は、ダウンタイムを最小限に抑えるため、移行を使用して実行されます。

3.5.4. 処理されない動作

以下の条件およびユーザーアクションは、RGManager ではサポートされません。

- クラスターが仮想マシンを管理している間に、非クラスター対応ツール (virsh または xm など) を使用して仮想マシンの状態や設定を操作すること。仮想マシンの状態はチェックすることができます (例: virsh list、virsh dumpxml)。
- クラスターが管理する VM を非クラスターノードや rgmanager を実行していないクラスター内のノードに移行する。Rgmanager は直前の場所で VM を再起動するため、VM の 2 つのインスタンスが実行されてしまい、ファイルシステムが破損します。

3.6. リソースアクション

RGManager は、リソースエージェントから以下の戻り値を予期します。

- start - リソースの起動
- stop - リソースの停止
- status - リソースの状態のチェック
- metadata - OCF RA XML メタデータのレポート

3.6.1. 戻り値

OCF には監視操作についての各種の戻りコードが含まれますが、rgmanager がステータスを呼び出すため、OCF はほぼ例外なく SysV-style 戻りコードに依存します。

0 - 成功

停止後の停止、または未実行時の停止により成功を戻します。

開始後の開始、または実行時の開始により成功を戻します。

ゼロ以外 - 失敗

停止操作がゼロ以外の値を戻す場合、サービスは失敗状態になり、サービスのリカバリーを手動で行う必要があります。

第4章 フェンシング

フェンシングとは、 クラスターの共有ストレージから任意のノードを切断することを指します。 フェンシングによって共有ストレージからの I/O を遮断することでデータの整合性を確保します。 クラスターのインフラストラクチャーではフェンスデーモン **fenced** を使ってフェンシングを実施します。

CMAN はノードに障害が発生していることを検出した場合、 その障害が発生しているクラスタインフラストラクチャの他のコンポーネントとの通信を行ないます。 CMAN より障害の通知を受け取った **fenced** は、 障害が発生しているノードの切断を行ないます。 他のコンポーネントでも行なうべき動作の確定が行なわれます (つまり、 必要とされる復元の動作を実行します)。 たとえば、 ノード障害の通知を受け取ると、 DLM や GFS2 では障害ノードに対する **fenced** のフェンシング完了が確認されるまではアクティビティの一時停止を行います。 障害ノードのフェンシング (切断) が確認されると、 DLM、 GFS2 は復元を実行します。 つまり、 DLM の場合は障害ノードのロックが解除され、 GFS2 の場合は障害ノードのジャーナル復元が行なわれます。

フェンシングプログラムでは、 使用するフェンシングメソッドをクラスター設定ファイルから確定します。 フェンシングメソッドはクラスター設定ファイル内の 2 種類のキーエレメント、 フェンスエージェントとフェンスデバイスで定義されます。 フェンシングプログラムは、 クラスター設定ファイル内で定義されるフェンスエージェントに対して呼び出しを行ないます。 呼び出しが行なわれると、フェンスエージェントによるフェンスデバイスを使ったノードの切断が行なわれます。 フェンシングが完了すると、フェンシングプログラムからクラスターマネージャーに通知が行なわれます。

High Availability Add-On では次のようなフェンシングメソッドを提供しています。

- 電源のフェンシング — 電力コントローラを使用するフェンシングメソッドで動作不能なノードの電源を切ります。
- ストレージのフェンシング — 動作不能なノードにストレージを接続しているファイバーチャンネルを無効にするフェンシングメソッドです。
- その他のフェンシング — IBM Bladecenter、 PAP、 DRAC/MC、 HP ILO、 IPMI、 IBM RSA II など、 動作不能なノードの I/O を無効にするまたは電源を切るフェンシングメソッドが他にもいくつかあります。

図4.1「電源のフェンシング例」 に電源のフェンシングの例を示します。 この例では、 ノード A 内のフェンシングプログラムでパワーコントローラによるノード D の電源オフを行なっています。 図4.2「ストレージのフェンシング例」 では、 ストレージフェンシングの例を示しています。 この例では、 ノード A 内のフェンシングプログラムでファイバーチャンネルスイッチによるノード D のポートの無効化が行なわれ、 ストレージからノード D を切断しています。

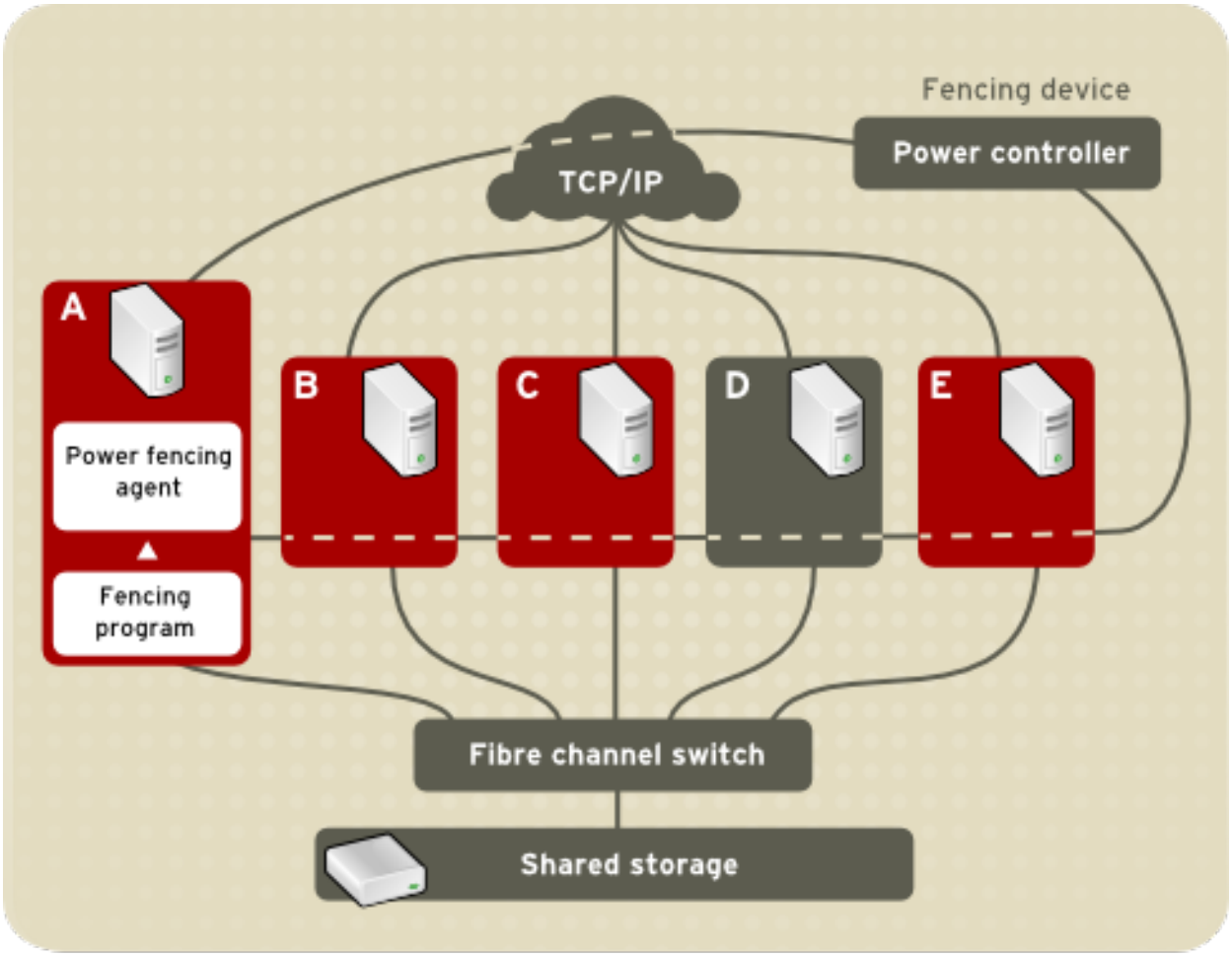


図4.1 電源のフェンシング例

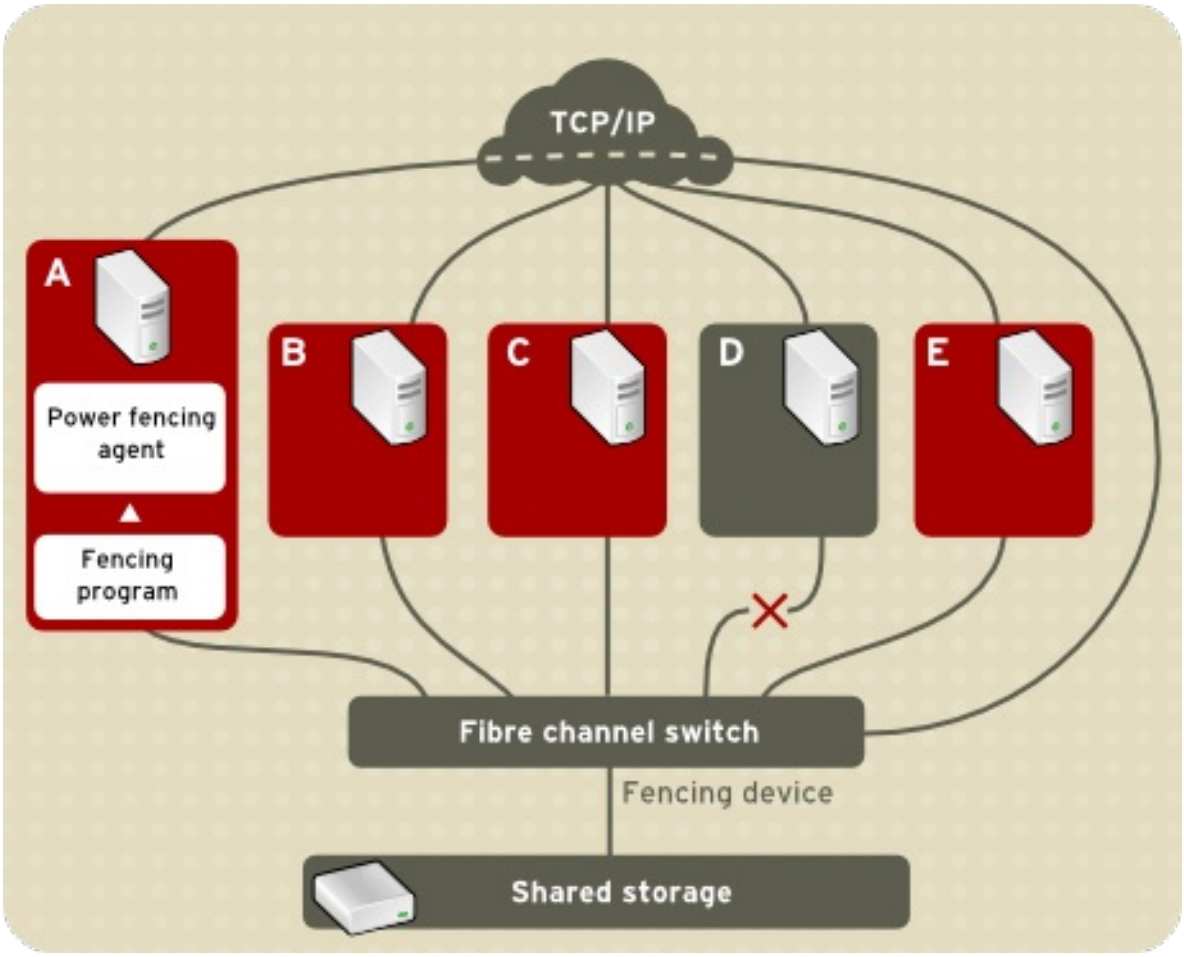


図4.2 ストレージのフェンシング例

クラスター設定ファイルを編集し、 クラスタ内の各ノードに対してフェンシングメソッド名、 フェンスエージェント、 フェンスデバイスをそれぞれ割り当てることでフェンシングメソッドを指定します。

フェンシングメソッドの指定方法は、 ノードが二重電源装置を備えているのか、 ストレージへのパスが複数あるのかによって異なってきます。 二重電源装置を備えている場合は、 ノードのフェンシングメソッドにはそれぞれの電源装置に対してひとつずつ少なくとも 2 つのフェンスデバイスを指定する必要があります (図4.3「二重電源装置を備えたノードのフェンシング」 を参照)。 同様に、 ファイバーチャンネルストレージへのパスが複数ある場合には、 ノードのフェンシングメソッドに各パスに対してひとつずつフェンスデバイスを指定する必要があります。 たとえば、 ファイバーチャンネルストレージへのパスが 2 つある場合なら、 フェンシングメソッドで各パスに対してひとつずつフェンスデバイスを指定します (図4.4「二重接続のファイバーチャンネルを備えたノードのフェンシング」 を参照)。

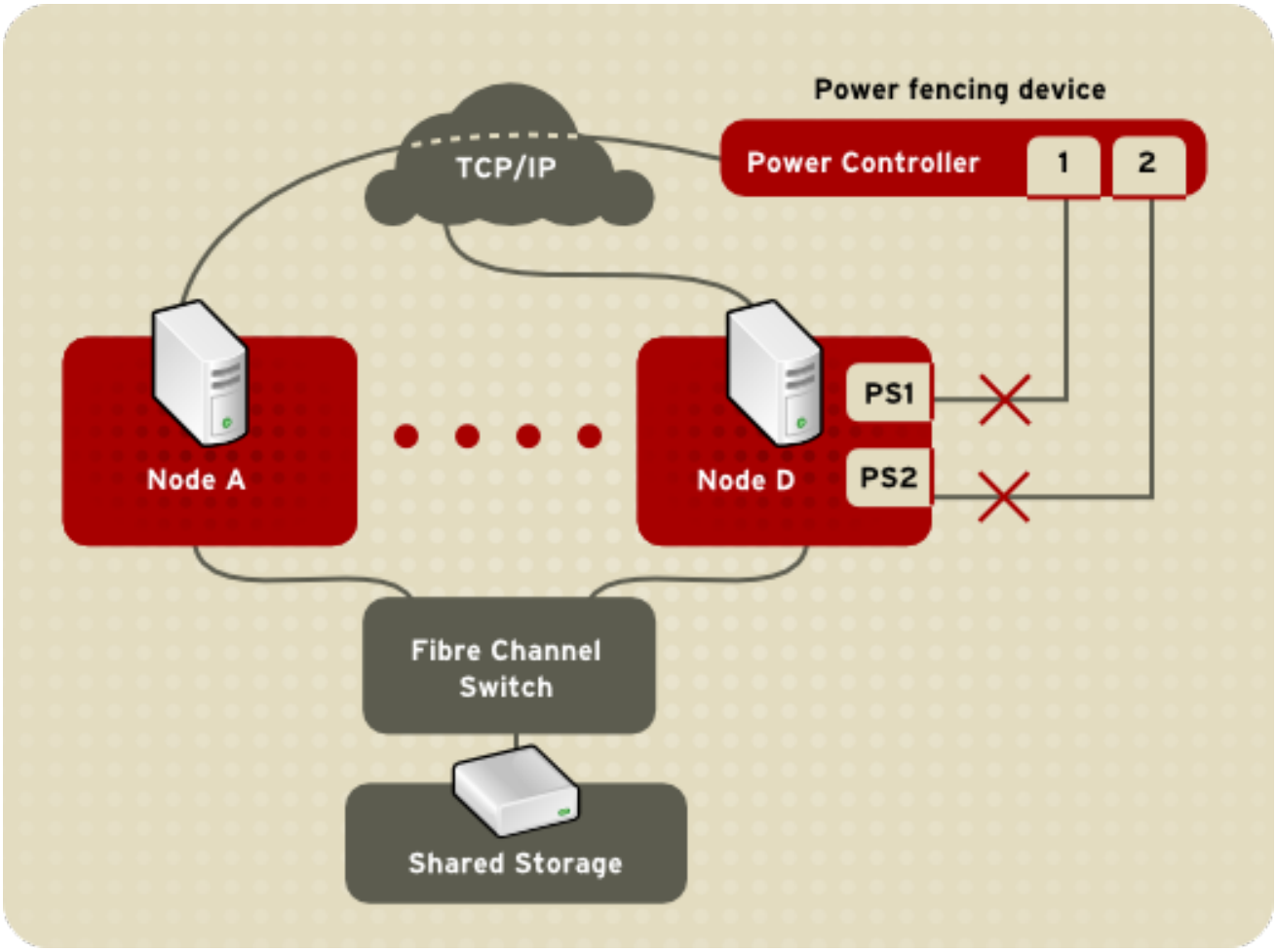


図4.3 二重電源装置を備えたノードのフェンシング

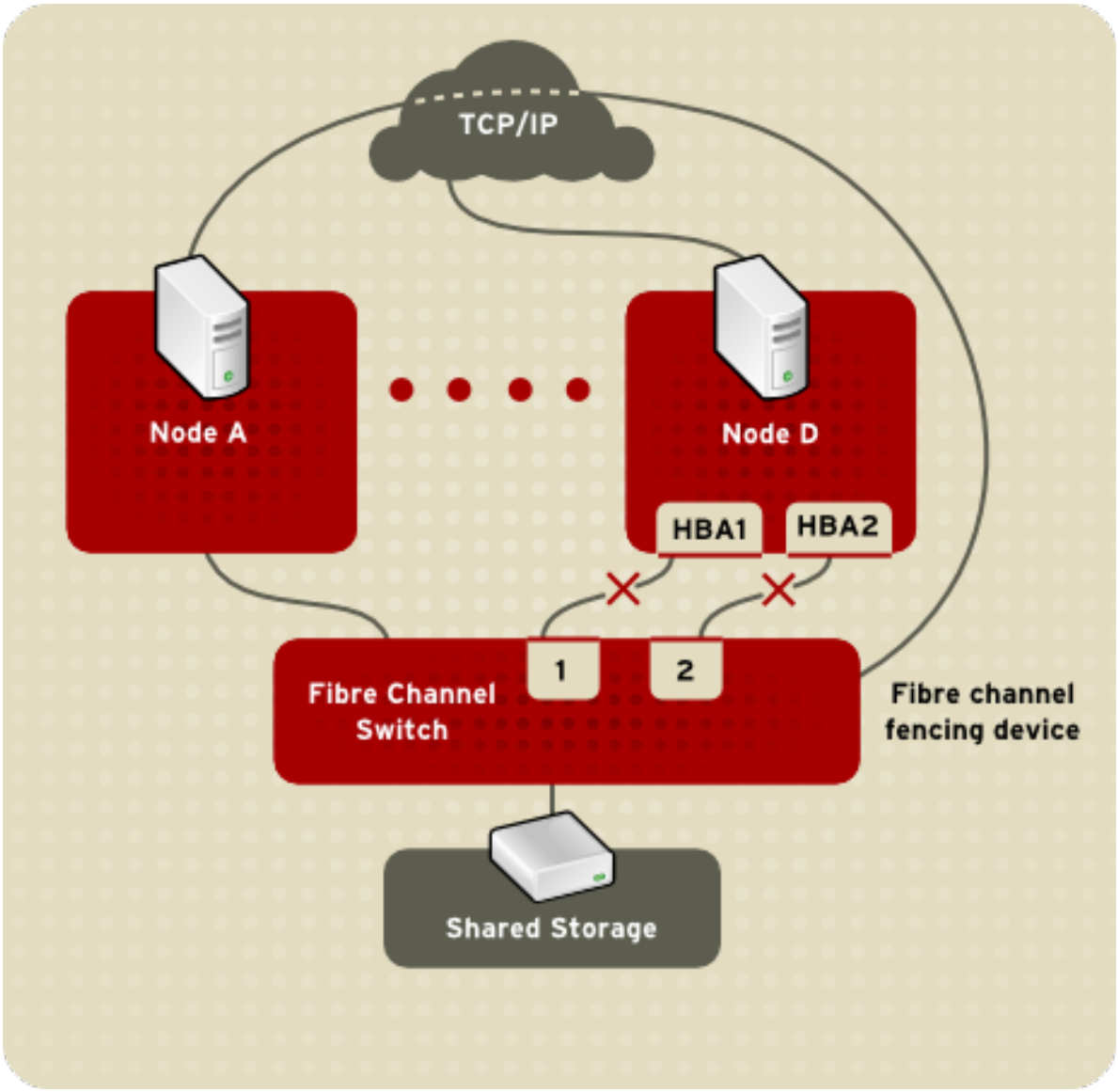


図4.4 二重接続のファイバーチャンネルを備えたノードのフェンシング

ノードに設定するフェンシングメソッドの数は一つでも複数でも構いません。フェンシングメソッドを一つだけ設定するということは、そのノードを切断できる方法がそのフェンシングメソッドのみということになります。複数のフェンシングメソッドを設定すると、クラスタ設定ファイルに指定されるフェンシングメソッドの順序に従って連結されることになります。ノードに障害が発生すると、 クラスター設定ファイルでそのノードに対して最初に指定されているフェンシングメソッドを使って切断が行なわれます。最初のフェンシングメソッドによる切断が失敗した場合、 次に指定されているフェンシングメソッドが使用されます。 いずれのフェンシングメソッドでも切断に失敗した場合には、 最初のフェンシングメソッドに戻り、ノードが切断されるまでクラスター設定ファイルに指定された順序で各フェンシングノードがエンドレスに使用されていきます。

フェンスデバイスの設定方法に関しては、 『クラスターの管理』 ガイドを参照してください。

第5章 ロック管理

ロック管理は、他のクラスターインフラストラクチャーコンポーネントが共有リソースへのアクセスを同期するために仕組みを提供する共通のクラスターインフラストラクチャーのサービスです。Red Hat クラスターでは、DLM (分散ロックマネージャー) がロックマネージャーです。

ロックマネージャーは、GFS ファイルシステムへのアクセスなどの、クラスター内のリソースへのアクセスを制御する Traffic Cop (交通を取り締まる警官) として機能します。ロックマネージャーがないと、共有ストレージへのアクセス制御がなくなり、クラスター内のノードが相互のデータを破損させる可能性があるので、ロックマネージャーは必要です。

名前に示されるように、DLM は分散ロックマネージャーであり、各クラスターノードで実行されます。ロック管理はクラスター内のすべてのノードを対象として分散されます。GFS2 および CLVM はロックマネージャーのロックを使用します。GFS2 は、ロックマネージャーのロックを使用して (共有ストレージ上の) ファイルシステムのメタデータへのアクセスを同期します。CLVM は、ロックマネージャーのロックを使用して (共有ストレージ上の) LVM ボリュームおよびボリュームグループへの更新を同期します。さらに、**rgmanager** は DLM を使用してサービスの状態を同期します。

5.1. DLM ロッキングモデル

DLM のロッキングモデルは、多数のロッキングモードのセットと、同期/非同期の両方の実行を提供します。アプリケーションは、ロックリソース上でロックを取得します。ロックリソースとロックの間には 1 対多の関係があります。単一のロックリソースには、これに関連付けられた複数のロックが含めることができます。

ロックリソースは、ファイル、データ構造、データベース、または実行可能なルーチンなどの実際のオブジェクトに対応させることができますが、必ずしもこれらのいずかに対応させる必要はありません。ロックリソースに関連付けるオブジェクトは、ロックの粒度を定めるものです。例えば、データベース全体のロックは粒度の粗いロックと見なされ、データベース内の各項目のロックは粒度の細かいロックと見なされます。

DLM のロッキングモデルは以下をサポートします。

- リソースへのアクセスを次第に制限する 6 つのロッキングモード
- 変換によるロックの昇格と降格
- ロック要求の同期完了
- 非同期完了
- ロック値ブロックによるグローバルデータ

DLM は、ロックトラフィックを管理するためのノード間通信や、ノード障害の発生後のロックのリマスタリングまたはノードがクラスターに入る際のロック移行を行うリカバリープロトコルなどのロック機能をサポートする独自の仕組みを提供します。ただし、DLM はクラスター自体を実際に管理する仕組みは提供しません。従って DLM は、以下の最低要件を満たす他のクラスターインフラストラクチャー環境と連動してクラスター内で動作することが予期されます。

- ノードはクラスターの一部である。
- すべてのノードがクラスターのメンバーシップについて一致しており、定足数を有している。
- IP アドレスはノード上で DLM と通信する必要がある。通常、DLM はノードごとの単一 IP アドレスに制限されるノード間通信用の TCP/IP を使用します (これはボンディングドライバーを使用して冗長化できます)。DLM は、1 ノードに複数の IP アドレスを許可するノード間トランス

ポート用として SCTP を使用するように設定できます。

DLM は、上記の最低要件を満たすすべてのクラスターインフラストラクチャー環境で機能します。オープンソースまたはクローズドソース環境のどちらを使用するについてはユーザーが決定できます。ただし DLM には、複数の環境を使って実施されるテスト量に関して主な制限があります。

5.2. ロックの状態

ロックの状態は、ロック要求の現在の状態を示します。ロックは常に以下の 3 つの状態のいずれかになります。

- Granted — ロック要求は成功し、要求されたモードを獲得しました。
- Converting — クライアントはロックモードの変更を試行しましたが、新規モードは既存のロックと互換性がありません。
- Blocked — 新規ロックの要求は、競合するロックがあるために許可されませんでした。

ロックの状態は、その要求されるモードと同じリソース上の他のロックのノードによって決定されます。

第6章 設定および管理ツール

クラスター設定ファイル **/etc/cluster/cluster.conf** は、High Availability Add-On 設定を指定します。設定ファイルは、以下のクラスター特性を記述する XML ファイルです。

- Cluster name — ノードがクラスターに加わるかまたはクラスターからフェンシングされる際に使用されるクラスター名、クラスター設定ファイルのリビジョンレベル、および基本的なフェンスタイミングのプロパティーを指定します。
- Cluster — クラスターの各ノードを指定し、ノードのノード名、ノード ID、定足数 投票数、およびフェンシングメソッドを指定します。
- Fence Device — クラスター内のフェンスデバイスを指定します。パラメーターはフェンスデバイスの種類によって変わります。例えば、フェンスデバイスとして使用されるパワーコントローラーの場合、クラスター設定ではパワーコントローラーの名前、その IP アドレス、ログインおよびパスワードを定義します。
- Managed Resources — クラスターサービスを作成するのに必要なリソースを指定します。管理対象リソースには、フェイルオーバードメイン、リソース (IP アドレスなど)、およびサービスの定義が含まれます。さらに管理対象リソースは、クラスターサービスおよびクラスターサービスのフェイルオーバー動作を定義します。

クラスターの設定は、スタートアップ時と設定が再ロードされた時に **/usr/share/cluster/cluster.rng** にあるクラススキーマに準じて自動的に妥当性が検証されます。また、**ccs_config_validate** コマンドを使用すると、いつでもクラスター設定の妥当性を検証できます。

注釈付きのスキーマは **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば **/usr/share/doc/cman-3.0.12/cluster_conf.html**) を参照してください。

設定の妥当性検証は、以下の基本的なエラーをチェックします。

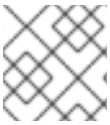
- XML 妥当性 — 設定ファイルが有効な XML ファイルであることをチェックします。
- 設定のオプション — オプション (XML の要素と属性) が有効であることをチェックします。
- オプションの値 — オプションに有効なデータが含まれていることをチェックします (限定的)。

6.1. クラスター管理ツール

Red Hat High Availability Add-On ソフトウェアの管理では、クラスターコンポーネント間の関係を指定するために設定ツールが使用されます。以下のクラスター設定ツールは、Red Hat High Availability Add-On で利用できます。

- **Conga** — Red Hat High Availability Add-On のインストール、設定および管理で使用される包括的なユーザーインターフェースです。**Conga** を使用して High Availability Add-On の設定および管理方法についての詳細は、『High Availability Add-On の設定と管理』を参照してください。
  - Luci — Conga のユーザーインターフェースを提供するアプリケーションサーバーです。これにより、ユーザーはクラスターサービスを管理でき、必要に応じてヘルプやオンラインドキュメントにアクセスできます。
  - Ricci — クラスター設定の分散を管理するサービスデーモンです。ユーザーパス設定の詳細情報に Luci インターフェースの使用方法が含まれ、この設定はクラスターノードへの分散用に corosync にロードされます。

- Red Hat Enterprise Linux 6.1 リリース以降の Red Hat High Availability Add-On は、**ccs** クラスター設定コマンドのサポートを提供しています。これにより、管理者は cluster.conf クラスター設定ファイルを作成し、修正し、表示することができます。**ccs** コマンドを使用して High Availability Add-On を設定し、管理する方法についての詳細は、『クラスターの管理』マニュアルを参照してください。



注記

system-config-cluster は RHEL 6 では利用できません。

第7章 仮想化と高可用性

さまざまな仮想化プラットフォームが、High Availability Add-On と Resilient Storage アドオンを使用する Red Hat Enterprise Linux 6 でサポートされています。Red Hat Enterprise Linux High Availability Add-On を使用する仮想化については、以下の 2 つのユースケースがサポートされています。

これには、仮想化プラットフォームとして使用できるベアメタルホスト上で実行される RHEL Cluster/HA が使用されます。このモードでは、クラスターリソースマネージャー (rgmanage) を設定し、仮想マシン (ゲスト) を高可用性リソースとして管理することができます。

- 高可用性リソース/サービスとしての VM
- ゲストクラスター

7.1. 高可用性リソース/サービスとしての VM

RHEL HA と RHEV はどちらも高可用性の仮想マシンを提供する仕組みを提供します。機能が重複していることを考慮すると、具体的なユースケースに適している製品を慎重に選択する必要があります。以下は、RHEL HA と RHEV のどちらを VM の高可用性を提供する製品として選択するかについての考慮すべきいくつかのガイドラインになります。

仮想マシンおよび物理ホスト数:

- 多数の物理ホストで多数の VM が高可用化されている場合、RHEV ではホスト CPU、メモリーおよび負荷情報などを考慮して VM 配置を管理する高度なアルゴリズムを提供するため、RHEV がより適したソリューションになるでしょう。
- 少数の物理ホストで少数の VM が高可用化されている場合、RHEL HA を使用すると追加される必要のあるインフラストラクチャーが少なくなるため、RHEL HA がより適したソリューションになるでしょう。RHEL HA の VM の最小ソリューションには、2 ノードクラスター用に 2 つの物理ホストが必要になります。一方、RHEV の最小ソリューションには RHEVM サーバーに高可用性を提供する 2 つのノードと、VM ホストとして機能する 2 つのノードの合計 4 ノードが必要です。
- どの程度のホストまたは VM が「多数」と見なされるかについての厳密なガイドラインはありませんが、単一の RHEL HA クラスター内の最大ホスト数は 16 であり、8 つ以上のホストを持つすべてのクラスターには、サポート容易性を判断するための Red Hat によるアーキテクチャーレビューが必要になることに注意してください。

仮想マシンの使用:

- お使いの高可用性 VM が共有インフラストラクチャーの提供用に使用されるサービスを提供している場合、RHEL HA または RHEV のいずれかを使用することができます。
- VM 内で実行中の小規模な重要なサービスセット向けに高可用性を提供する必要がある場合は、RHEL HA または RHEV を使用することができます。
- 仮想マシンの迅速なプロビジョニングを行うインフラストラクチャーを検討している場合には、RHEV を使用する必要があります。
    - RHEV の VM では、動的な高可用性が意図されています。新規の VM は RHEV の「クラスター」に簡単に追加でき、完全にサポートされます。
    - RHEL の VM の高可用性環境は高度に動的にならないものとされています。VM の固定されたセットを含むクラスターは、いったんセットアップされるとクラスターの有効期間中そのまま維持されるため、VM の追加または削除を行うことは推奨されません。

- RHEL の高可用性は、そのクラスター設定における静的な性質と、物理ノードの最大カウント数 (16 ノード) が比較的に少ないことを考慮すると、クラウドのような環境を作成するためのインフラストラクチャーを提供する目的で使用することはできません。

RHEL 5 は、2 つの仮想化プラットフォームをサポートします。Xen は RHEL 5.0 リリース以降サポートされており、RHEL 5.4 では KVM が導入されました。

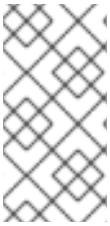
RHEL 6 は、仮想化プラットフォームとして KVM のみをサポートしています。

RHEL 5 AP クラスターは、ホストクラスターインフラストラクチャーで管理される仮想マシンの実行に、KVM と Xen の両方の使用をサポートします。

RHEL 6 HA は、ホストクラスターインフラストラクチャーで管理される仮想マシンの実行に使用される KVM をサポートしています。

以下は、Red Hat によって現在サポートされているデプロイメントシナリオのリストです。

- RHEL 5.0+ は、RHEL AP クラスターとの併用で Xen をサポートしています。
- RHEL 5.4 では、テクノロジープレビュー用の RHEL AP クラスター内の管理対象リソースとして、KVM 仮想マシンのサポートが導入されました。
- RHEL 5.5+ では、KVM 仮想マシンの完全サポートに向けてサポートがレベルアップしました。
- RHEL 6.0+ は、RHEL 6 High Availability Add-On の高可用性リソースとして KVM 仮想マシンをサポートしています。
- RHEL 6.0+ では、RHEL 6 が Xen をサポートしなくなったため、RHEL 6 High Availability Add-On での Xen 仮想マシンのサポートを行っていません。



注記

サポートされているデプロイメントシナリオの更新情報および特記については、以下の Red Hat ナレッジベースのエントリーを参照してください。

https://access.redhat.com/kb/docs/DOC-46375

管理対象リソースとして実行される仮想マシンがどの種類であるかは、問題にはなりません。RHEL で Xen または KVM のいずれかによってサポートされているすべてのゲストを高可用性ゲストとして使用できます。これには、各種の RHEL (RHEL3、RHEL4、 RHEL5) や Microsoft Windows のいくつかの種類が含まれます。RHEL のドキュメンテーションを参照し、各ハイパーバイザーでサポートされるゲストオペレーティングシステムの最新リストを確認してください。

7.1.1. 一般的な推奨事項

- RHEL 5.3 以前には、rgmanager は Xen domU (ゲスト) を管理するためにネイティブの Xen インターフェースを使用しました。RHEL 5.4 では、Xen と KVM の 2 つのタイプのハイパーバイザー間に一貫したインターフェースを提供する目的で、これらのハイパーバイザーの両方に libvirt を使用できるような変更が加えられました。このアーキテクチャーの変更のほかにも RHEL 5.4 と 5.4.z では数多くのバグ修正がリリースされたため、Xen の管理対象のサービスを設定する際には、少なくとも最新の RHEL 5.5 パッケージにホストクラスターをアップグレードすることをお勧めします。
- KVM の管理対象サービスについては、最初にこの機能の完全サポートを始めた RHEL のバージョン RHEL 5.5 にアップグレードする必要があります。

- クラスターをデプロイする前には最新の RHEL エラータを常に確認し、既知の問題やバグについての最新の修正が適用されていることを確認してください。

- 複数の異なるタイプのハイパーバイザーのホストを混在させることはサポートされていません。ホストクラスターはすべて Xen であるか、またはすべて KVM ベースであるかのいずれかである必要があります。

- ホストハードウェアのプロビジョニングでは、ホストによるメモリーのオーバーコミットや仮想 CPU のオーバーコミットなしに、ホストハードウェアが複数の失敗したホストから再配置されたゲストを吸収できるようにする必要があります。メモリーまたは仮想 CPU のいずれかのオーバーコミットを引き起こす障害が発生した場合、重大なパフォーマンスの低下やクラスター障害が発生する可能性があります。

- rgmanager の制御下にある (live migrate、stop.start) 仮想マシンの管理に xm または libvirt ツール (virsh、virt-manager) を直接使用することは、クラスター管理スタックのバイパスが生じる可能性があるためにサポートされておらず、推奨されていません。

- それぞれの VM の名前は、ローカル専用/非クラスター VM を含め、クラスター全体で一意の名前です。Libvirtd はホストごとに一意の名前のみを使用します。VM を手動で複製する場合、クローンの設定ファイルで名前を変更する必要があります。

7.2. ゲストクラスター

これは、各種の仮想化プラットフォーム上で仮想化されたゲスト内で実行される RHEL Cluster/HA に関連するものです。このユースケースでは、ゲスト内で実行されるアプリケーションの高可用性を維持するために RHEL クラスタリング/HA が主に使用されます。このユースケースは、RHEL クラスタリング/HA がベアメタルホストで常に使用されてきた方法と類似しています。相違点は、クラスタリングがゲスト内で実行される点です。

以下は、仮想化プラットフォームと RHEL Cluster/HA を使用してゲストクラスターを実行するために現在利用できるサポートレベルのリストです。以下のリストでは、RHEL 6 のゲストは、High Availability (コアクラスタリング) と Resilient Storage アドオン (GFS2、clvmd および cmirror) の両方をサポートします。

- RHEL 5.3+ Xen ホストは、ゲストのオペレーティングシステムも REHL 5.3 以上であるゲストクラスターを完全にサポートします。
    - Xen ゲストクラスターは、ゲストのフェンシングに fence_xvm または fence_scsi のいずれかを使用することができます。
    - fence_xvm/fence_xvmd を使用するには、fence_xvmd をサポートするためにホストクラスターを実行している必要があり、fence_xvm はすべてのクラスター化されたゲストでゲストフェンシングエージェントとして使用される必要があります。
    - 共有ストレージは、ホストブロックストレージまたはファイルレベルのストレージのいずれかでバックアップされる iSCSI または Xen の共有ブロックデバイスのいずれかによって提供されます (RAW イメージ)。
- RHEL 5.5+ KVM ホストは、ゲストクラスターの実行をサポートしません。
- RHEL 6.1+ KVM ホストは、ゲストのオペレーティングシステムが RHEL 6.1+ または RHEL 5.6+ のいずれかであるゲストクラスターの実行をサポートします。RHEL 4 ゲストはサポートされません。
    - ベアメタルのクラスターノードを仮想化されたクラスターノードと混在することは許可されています。

- RHEL 5.6+ ゲストクラスターは、ゲストフェンシング用に fence_xvm または fence_scsi のいずれかを使用できます。
- RHEL 6.1+ ゲストクラスターは、ゲストフェンシング用に fence_xvm (**fence-virt** パッケージ内) または fence_scsi のいずれかを使用できます。
- RHEL 6.1+ KVM ホストは、ゲストクラスターが fence_virt または fence_xvm をフェンスエージェントとして使用している場合に fence_virtd を使用する必要があります。ゲストクラスターが fence_scsi を使用している場合は、ホスト上に fence_virtd は不要です。
- fence_virtd は、 3 つのモードで稼働できます。
  - ホストとゲストのマッピングがハードコーディングされており、ゲストのライブマイグレーションが許可されないスタンドアロンモード。
  - クラスター化されたゲストのライブマイグレーションを追跡するために Openais チェックポイントサービスを使用する。これには、ホストクラスターが実行中である必要があります。
  - libvirt-qpid パッケージで提供される Qpid Management Framework (QMF) を使用する。QMF を使用することにより、完全なホストクラスターがなくてもゲストの移行を追跡できます。
- 共有ストレージは、ホストブロックストレージか、またはファイルレベルのストレージのいずれかでバックアップされる iSCSI または KVM の共有ブロックデバイスのいずれかで提供されます (RAW イメージ)。

- Red Hat Enterprise Virtualization Management (RHEV-M) バージョン 2.2+ および 3.0 では、RHEL 5.6+ および RHEL 6.1+ のクラスターゲストを現在サポートしています。
  - ゲストクラスターは同種である必要があります (すべてが RHEL 5.6+ ゲストか、またはすべてが RHEL 6.1+ ゲストのいずれか)。
  - ベアメタルのクラスターノードを仮想化されたクラスターノードと混在することは許可されています。
  - フェンシングは、RHEV-M 2.2+ の fence_scsi および RHEV-M 3.0 の fence_scsi と fence_rhevm の両方で提供されます。フェンシングは、以下に説明する fence_scsi を使用してサポートされます。
    - fence_scsi と iSCSI ストレージの使用は、preempt-and-abort コマンドを使って SCSI 3 永続予約をサポートする iSCSI サーバーに制限されます。すべての iSCSI サーバーがこの機能をサポートする訳ではありません。ストレージベンダーにご確認の上、お使いのサーバーが SCSI 3 永続予約サポートに準拠しているかを確認してください。Red Hat Enterprise Linux と共に出荷される iSCSI サーバーは、SCSI 3 永続予約を現在サポートしていないため、 fence_scsi の使用には適していないことに注意してください。

- VMware vSphere 4.1、VMware vCenter 4.1、VMware ESX および ESXi 4.1 は、ゲストのオペレーティングシステムが RHEL 5.7+ または RHEL 6.2+ であるゲストクラスターの実行をサポートします。VMware vSphere、vCenter、ESX および ESXi のバージョン 5.0 もサポートされています。ただし、Vmware vSphere 5.0 の初期リリースで提供された WDSL スキーマが未完成であったため、fence_vmware_soap ユーティリティーは、デフォルトのインストールでは機能しません。この問題を解決するための更新された手順については、Red Hat ナレッジベース https://access.redhat.com/knowledge/ を参照してください。

- ゲストクラスターは同種である必要があります (すべてが RHEL 5.7+ ゲストか、またはすべてが RHEL 6.1+ ゲストであるかのいずれか)。
- ベアメタルのクラスターノードを仮想化されたクラスターノードと混在することは許可されています。
- fence_vmware_soap エージェントには、サードパーティーの VMware perl API が必要です。このソフトウェアパッケージは、VMware の Web サイトからダウンロードしてから RHEL のクラスターゲストにインストールする必要があります。
- または、後述するように fence_scsi を使用してフェンシング機能を提供することができます。
- 共有ストレージは、iSCSI または VMware の RAW 共有ブロックデバイスによって提供されます。
- VMware ESX ゲストクラスターの使用は、fence_vmware_so_ap または fence_scsi のいずれかの使用によりサポートされています。

- Hyper-V ゲストクラスターの使用は、現時点ではサポートされていません。

7.2.1. fence_scsi および iSCSI 共有ストレージの使用

- 上記のすべての仮想化環境において、fence_scsi および iSCSI ストレージを、ネイティブの共有ストレージとネイティブのフェンスデバイスの代わりとして使用できます。
- iSCSI ターゲットが SCSI3 永続予約と preempt and abort コマンドを適切にサポートする場合、iSCSI で提供される共有ストレージ用の I/O フェンシングを提供するために fence_scsi を使用することができます。ストレージベンダーにお問い合わせの上、iSCSI ソリューションが上記の機能をサポートするかどうかをご確認ください。
- RHEL と共に出荷される iSCSI サーバーソフトウェアは SCSI 3 永続予約をサポートしないため、fence_scsi と共に使用することはできません。ただし、fence_vmware または fence_rhevm などの他のフェンスデバイスと併用して共有ストレージソリューションとして使用するのに適しています。
- すべてのゲスト上で fence_scsi を使用している場合に、ホストクラスターは不要です (RHEL 5 Xen/KVM および RHEL 6 KVM ホストのユースケースの場合)
- fence_scsi がフェンスエージェントとして使用される場合、すべての共有ストレージは iSCSI 上にある必要があります。iSCSI とネイティブの共有ストレージを混在させることはできません。

7.2.2. 一般的な推奨事項

- 前述のように、数多くの拡張機能とバグ修正があるため、仮想化機能を使用する前にホストとゲストの両方を最新の RHEL パッケージにアップグレードすることをお勧めします。
- ゲストクラスターの下に仮想化プラットフォーム (ハイパーバイザー) を混在させることはサポートされていません。すべての基礎となるホストには、同じ仮想化テクノロジーを使用する必要があります。
- 単一の物理ホスト上でゲストクラスター内のすべてのゲストを実行することはサポートされていません。単一ホストに障害が発生すると高可用性が提供されなくなるためです。ただしこの設定は、プロトタイプや開発の目的で使用することができます。

- ベストプラクティスには以下が含まれます。
  - ゲストごとに単一ホストを設定する必要はありませんが、この設定にすると、ホスト障害がクラスター内の単一ノードのみに影響を与えるために最も高い可用性が提供されます。2 対 1 のマッピング (1 物理ホストごとに単一クラスター内の 2 つのゲスト) の場合、単一ホストの障害が 2 つのゲストの障害に発展することを意味します。従って、可能な限り 1 対 1 のマッピングとすることをお勧めします。
  - 現時点で、物理ホストの同一のセット上で複数の独立したゲストを混在させることは、fence_xvm/fence_xvmd または fence_virt/fence_virtd フェンスエージェントを使用する際にはサポートされていません。
  - fence_scsi + iSCSI ストレージを使用するか、または fence_vmware + VMware (ESX/ESXi および vCenter) を使用する場合には、物理ホストの同一のセット上で複数の独立したゲストを混在させても問題なく機能します。
  - 物理ホストの同一のセット上にクラスター化されていないゲストをゲストクラスターとして実行することはサポートされていませんが、ホストクラスターが設定されている場合にホストは相互の物理的なフェンシングを行うため、これらの他のゲストもホストフェンシングの操作時に強制終了されます。
  - ホストハードウェアは、メモリーまたは仮想 CPU のオーバーコミットを防ぐことができるようにプロビジョニングする必要があります。メモリーまたは仮想 CPU のオーバーコミットはパフォーマンスを低下させます。パフォーマンスの低下が深刻になると、クラスターのハートビートが影響を受け、クラスターの障害が発生する可能性があります。

付録A 改訂履歴

改訂 1-15.2 **Fri Feb 6 2015**
翻訳ドラフト

改訂 1-15.1 **Fri Feb 6 2015**
翻訳ファイルを XML ソースバージョン 1-15 と同期

改訂 1-15 **Tue Dec 16 2014** **Steven Levine**
RHEL 6 スプラッシュページに sort_order を実装するため更新

改訂 1-13 **Wed Oct 8 2014** **Steven Levine**
Red Hat Enterprise Linux 6.6 の GA リリース

改訂 1-12 **Thu Aug 7 2014** **Steven Levine**
Red Hat Enterprise Linux 6.6 の Beta リリース

改訂 1-11 **Fri Aug 1 2014** **Steven Levine**
修正: #852720
編集上の問題

改訂 1-10 **Fri Jun 6 2014** **Steven Levine**
Red Hat Enterprise Linux 6.6 のドラフト版

改訂 1-7 **Wed Nov 20 2013** **John Ha**
Red Hat Enterprise Linux 6.5 の GA リリース

改訂 1-4 **Mon Feb 18 2013** **John Ha**
Red Hat Enterprise Linux 6.4 向けリリース

改訂 1-3 **Mon Jun 18 2012** **John Ha**
Red Hat Enterprise Linux 6.3 GA 向けリリース

改訂 1-2 **Fri Aug 26 2011** **John Ha**
6.2 リリース用に更新

改訂 1-1 **Wed Nov 10 2010** **Paul Kennedy**
初期リリース