Red Hat Enterprise Linux 6

High Availability

Обзор дополнения High Availability для Red Hat Enterprise Linux

Редакция 6

Last Updated: 2017-10-29

Red Hat Enterprise Linux 6 High Availability

Обзор дополнения High Availability для Red Hat Enterprise Linux

Редакция 6

Юридическое уведомление

Аннотация

В этом документе приведена обзорная информация о дополнении High Availability для Red Hat Enterprise Linux 6.

Содержание

ВВЕДЕНИЕ

В этом документе приведены общие сведения о дополнении **High Availability** для **Red Hat Enterprise Linux 6.**

Поскольку приведенная здесь информация является лишь обзорной, она предназначена для опытных администраторов **Red Hat Enterprise Linux,** знакомых с концепциями серверных вычислений.

За информацией о **Red Hat Enterprise Linux** обратитесь к документации:

- *Руководство по установке* ***Red Hat Enterprise Linux 6***
- *Руководство по развертыванию* предоставляет информацию о конфигурации и администрированию **Red Hat Enterprise Linux 6.**

Информацию о дополнениях и других решениях **Red Hat Enterprise Linux 6** можно найти в документах:

- *Конфигурация и управление дополнением* ***High Availability*** обсуждает вопросы эксплуатации решения **High Availability** (также известного как **Red Hat Cluster**) более подробно.
- *LVM* содержит информацию об управлении логическими томами (**LVM, Logical Volume Manager**), в том числе— в кластерном окружении.
- *GFS2* предлагает информацию об установке, настройке и эксплуатации файловой системы **GFS2** (**Global File System 2**).
- *DM Multipath* предлагает информацию о многопутевых возможностях **Red Hat Enterprise Linux 6.**
- *Распределение нагрузки* предоставляет информацию о настройке высокопроизводительных систем и служб с помощью дополнения **Red Hat Load Balancer** (раньше известного как виртуальный сервер **Linux**).
- *Примечания к выпуску* содержат краткий обзор последнего выпуска **Red Hat.**

ПРИМЕЧАНИЕ

В статье «Приемы развертывания кластера **Red Hat Enterprise Linux** с применением **High Availability** и **GFS2**» обсуждаются подходы к построению отказоустойчивого кластера с файловой системой **GFS2.**

Полный каталог документов в форматах **HTML, PDF** и **RPM** можно найти на диске документации **Red Hat Enterprise Linux** и на сайте https://access.redhat.com/site/documentation/.

1. ОТЗЫВЫ И ПРЕДЛОЖЕНИЯ

Чтобы оставить отзыв или сообщить об опечатках, создайте запрос в **Bugzilla** по адресу http://bugzilla.redhat.com/, выбрав продукт **Red Hat Enterprise Linux 6** , компонент *doc-High_Availability_Add-On_Overview* и версию **6 . 6.**

Чтобы облегчить поиск ошибок, укажите номер раздела и параграф.

ГЛАВА 1. ОБЗОР ВЫСОКОДОСТУПНЫХ РЕШЕНИЙ

Red Hat High Availability реализует кластерную схему с высоким уровнем доступа, масштабирования и отказоустойчивости в критических окружениях. Основные функции и компоненты этого дополнения обсуждаются в следующих секциях:

- Раздел 1.1, «Основы кластеризации»
- Раздел 1.2, «Знакомство с Red Hat High Availability»
- Раздел 1.3, «Инфраструктура кластера»

1.1. ОСНОВЫ КЛАСТЕРИЗАЦИИ

Кластер включает как минимум два компьютера, называемых *узлами* или *элементами*, которые совместно решают поставленные задачи. Существует четыре категории кластеров:

- Хранение данных
- Высокая доступность
- Распределение нагрузки
- Высокая производительность

Кластеры хранения обеспечивают бесперебойный доступ к файловой системе, разрешая вести параллельную запись и чтение данных с разных узлов. Управление хранилищем облегчается за счет того, что программы и исправления устанавливаются только один раз в единой файловой системе. В кластерных файловых системах необходимость в создании избыточных копий программных данных отпадает, а процессы создания резервных копий и восстановления существенно упрощаются. Кластерное хранилище создается средствами Red Hat High Availability в комплексе с Red Hat GFS2 (в комплекте Resilient Storage).

Высокодоступные кластеры (их также называют отказоустойчивыми), как и следует из определения, обеспечивают постоянный доступ к службам за счет исключения единой точки отказа, перенося службы с одного узла на другой в случае его неисправности. Обычно сервисы в таком кластере осуществляют чтение и запись данных (файловые системы смонтированы в режиме чтения и записи), поэтому при отказе узла кластер должен обеспечить их целостность при передаче управления другому узлу. Сбои узлов в отказоустойчивых кластерах обрабатываются незаметно для клиентов за их пределами. За обеспечение непрерывного доступа к кластерным сервисам отвечает менеджер ресурсов `rgmanager` (входит в комплект Red Hat High Availability).

Кластеры распределения нагрузки, как и следует из названия, распределяют запросы обслуживания между узлами. Балансировка является экономичным способом масштабирования, поскольку в зависимости от нагрузки вы сможете задействовать необходимое число узлов. В случае отказа узла запросы будут перенаправляться другим узлам, поэтому его сбой останется незаметным для клиентов за пределами кластера. Эта функциональность реализована в дополнении Red Hat Load Balancer.

В высокопроизводительных кластерах происходит распараллеливание вычислений между узлами, что существенно повышает скорость обработки данных. Такие кластеры иногда называют вычислительными кластерами и схемами распределенных вычислений.

ПРИМЕЧАНИЕ

Перечисленные типы можно комбинировать, создавая на их базе комплексные кластерные схемы.

Red Hat High Availability не поддерживает высокопроизводительные кластеры.

1.2. ЗНАКОМСТВО С RED HAT HIGH AVAILABILITY

Red Hat High Availability представляет собой набор программных компонентов, позволяющий построить самые разнообразные схемы с высоким уровнем доступа в зависимости от необходимого уровня производительности, отказоустойчивости, степени распределения нагрузки, масштабируемости, общего доступа к файлам и рентабельности.

Основные составляющие и функции:

- Инфраструктура кластера — базовая функциональность для объединения узлов в кластер, включая управление файлами конфигурации, элементами кластера, блокировками, а также функции изоляции узлов.
- Непрерывный доступ — в случае отказа узла его сервисы будут перенесены на другой узел.
- Средства администрирования — утилиты конфигурации и управления кластером.

ПРИМЕЧАНИЕ

Географически распределенные кластеры пока официально не поддерживаются. За подробной информацией обратитесь к представителю Red Hat.

Функциональность отказоустойчивого кластера может быть дополнена:

- Red Hat GFS2 (Global File System 2) — кластерная файловая система предоставляется в комплекте Red Hat Resilient Storage. Позволяет совместно использовать пространство данных на блочном уровне, как будто устройство подключено к узлу напрямую. GFS2 создается на базе кластерной инфраструктуры.
- CLVM (Cluster Logical Volume Manager) — управление кластерным хранилищем.
- Load Balancer — комплект распределения нагрузки устанавливается на двух избыточных виртуальных серверах, которые будут распределять запросы обслуживания между физическими серверами.

1.3. ИНФРАСТРУКТУРА КЛАСТЕРА

Инфраструктура кластера предоставляет базовую функциональность для объединения компьютеров (также называемых *узлами*) в кластер. Готовый кластер можно дополнить другими компонентами для обеспечения совместного доступа к файлам в GFS2, переноса служб в случае отказа узла и т.п. Основные функции кластерной инфраструктуры, которые будут обсуждаться в этом руководстве:

- управление кластером;
- управление блокировками;
- изоляция неисправных узлов;

- управление конфигурацией кластера.

ГЛАВА 2. УПРАВЛЕНИЕ КЛАСТЕРОМ

В отказоустойчивом кластере Red Hat задачи управления его элементами и формирования кворума решает менеджер CMAN (Cluster MANager), работающий на всех узлах и реализующий распределенный механизм управления.

CMAN наблюдает за работой кластера, отслеживая поток сообщений между узлами. Если узел не отвечает на протяжении заданного периода времени, CMAN отключит его от кластера и сообщит об этом другим компонентам кластерной инфраструктуры, чтобы они смогли выбрать подходящую линию поведения.

CMAN следит за числом рабочих узлов, исходя из которого формируется кворум: если активно более половины узлов, кворум достигнут. Если кворума нет, работа кластера будет приостановлена. Также кворум поможет принять решение при расщеплении кластера на две равные части и избежать ситуации, в которой обе части будут продолжать работать, считая, что другая часть неработоспособна.

2.1. КВОРУМ

CMAN определяет наличие кворума методом голосования.

По умолчанию каждый узел имеет один голос. Кворум достигается большинством голосов. Так, например, в кластере с 13 узлами кворум будет набран при наличии 7 активных узлов, но при отказе еще одного узла он будет потерян, и кластер не сможет продолжить работу.

Принуждение кворума помогает решить проблему, возникающую при распадении кластера на одинаковые сегменты вследствие потери связи между узлами. Оба сегмента будут продолжать работать и изменять данные независимо друг от друга, что может нарушить целостность данных в файловой системе. Правила кворума помогут выбрать сегмент, которому будет разрешено использовать общее пространство данных.

Такой подход не предотвращает подобные ситуации, но помогает принять решение о том, какой сегмент продолжит работу.

Активность узлов определяется наличием Ethernet-трафика между узлами, а кворум принимается большинством голосов, то есть в кластере должно быть активно 50% узлов плюс 1. Чтобы решить проблему распадения кластера на равные сегменты, можно дополнительно добавить кворумный диск.

ПРИМЕЧАНИЕ

По умолчанию узел имеет один голос, но это можно изменить и выделить ему произвольное число голосов.

2.1.1. Кворумный диск

При расколе кластера кворумный диск (или раздел) поможет выбрать рабочий сегмент.

Представим кластер с двумя узлами. Если узел A вдруг перестал получать пакеты с узла B, причин этому может быть несколько: узел B вышел из строя; произошла ошибка на уровне сетевого коммутатора или шлюза; ошибка сетевого адаптера узла A; узел B просто сильно загружен (что не исключено в больших кластерах или при нестабильном сетевом соединении).

Узел A не знает, заключается ли причина в узле B или в нем самом. В результате оба узла могут попытаться изолировать друг друга.

В этой ситуации следует убедиться, действительно ли узел B вышел из строя. Для этого и нужен кворумный диск: если узел может записывать данные на этот диск, значит он работоспособен.

В кластере с двумя узлами кворумный диск выступает в роли арбитра: если узел успешно записывает данные на кворумный диск и у него есть доступ к сети, он получает дополнительный голос. Узел, набравший меньше голосов, будет отключен.

Если же узел действительно вышел из строя, он потеряет свой голос, и его можно будет изолировать.

Подробную информацию о кворумных дисках можно найти в руководстве *Администрирование кластера*.

2.1.2. Арбитражные алгоритмы

Арбитражные алгоритмы используют эвристические правила для проверки работоспособности узлов.

Один из таких алгоритмов использует ping для проверки связи с маршрутизатором, использующим те же каналы подключения, что и остальные узлы кластера. Если узлы не могут связаться друг с другом, выиграет тот, который сможет подключиться к маршрутизатору. Конечно, может оказаться так, что оба узла успешно подключаются к маршрутизатору, — именно поэтому необходимо верно настроить правила изоляции.

Другие алгоритмы используют общий раздел (кворумный диск). Например, clumanager 1.2.x в Red Hat Cluster Suite 3 не ограничивал работу узлов даже при отсутствии подключения к сети при условии, что они могли взаимодействовать через общий раздел.

Более сложные схемы, такие как QDisk (в составе linux-cluster), позволяют настроить эвристические правила для отдельных узлов. Подробную информацию можно найти на справочной странице qdisk(5).

У CMAN нет собственных арбитражных алгоритмов, но он может использовать внешние API, что позволит зарегистрировать кворумное устройство или получить доступ к исходному коду QDisk.

Арбитражный алгоритм требуется:

- В схемах с двумя узлами, если сетевой путь к устройству изоляции отличается от пути к кластеру.
- В схемах с двумя узлами, где изоляция неисправных узлов осуществляется на уровне коммутаторов (особенно с резервированием SCSI).

Стоит помнить, что арбитражный алгоритм значительно усложняет конфигурацию кластера.

ГЛАВА 3. RGMANAGER

RGManager — менеджер кластерных ресурсов, отвечающий за настройку, управление и восстановление групп ресурсов, также называемых сервисами. Группа ресурсов представляет собой древовидную структуру, связи в которой построены по принципу семейных взаимосвязей.

RGManager включает собственные инструменты для конфигурации и мониторинга сервисов. Так, в случае отказа узла RGManager сможет перенести сервис на другой узел с минимальными помехами для рабочего процесса. Дополнительно можно определить подмножество узлов, на которых будет работать сервис, — например `httpd` привязать к одной группе узлов, а `mysql` — к другой.

В этой главе обсуждаются основные понятия и рабочие процессы RGManager:

- Резервный домен — подмножество узлов, на которых может работать кластерный сервис.
- Сервисная политика — правила запуска и восстановления сервисов.
- Дерево ресурсов — иерархическая структура ресурсов, определяющая порядок их запуска и остановки.
- Управление сервисами — управляющие команды, изменяющие состояние сервиса.
- Управление виртуальными машинами — особенности управления виртуальными машинами в кластере.
- Команды управления ресурсами — команды агентов, которые использует RGManager, и их настройка в `cluster.conf`.
- Сценарии событий — если стандартные сервисные политики rgmanager вас не устраивают, их можно адаптировать при помощи сценариев.

3.1. РЕЗЕРВНЫЙ ДОМЕН

Резервный домен представляет собой подмножество узлов, на которых может работать кластерный сервис. Наличие резервных доменов не является обязательным условием для нормального функционирования кластера, но поможет создать гибкую конфигурацию.

Поведение домена определяется несколькими характеристиками:

- Предпочтительный узел — узел, на котором сервис запускается в первую очередь. Выбор предпочтительного узла можно эмулировать, создав неупорядоченный, неограниченный домен с единственным элементом.
- Ограниченный домен — сервис может работать только в пределах этого домена. Если в домене не осталось свободных узлов, сервис будет приостановлен. Такая организация может упростить конфигурацию сервисов, аналогичных httpd, который требует, чтобы все узлы, на которых он выполняется, были настроены одинаково. В этом случае его можно закрепить за доменом, содержащим узлы с идентичной конфигурацией.
- Неограниченный домен (по умолчанию) — сервис может работать на любом узле кластера, но предпочтение будет отдаваться узлам в домене. Если сервис выполняется за пределами домена, и в это время в домене появился свободный узел, он будет перенесен на этот узел (это поведение можно отключить, установив флаг «nofailback»).
- Упорядоченный домен — позволяет определить предпочтительный порядок выбора узлов. Это означает, что при активации узла A, имеющего более высокий приоритет по сравнению с узлом B,

сервис, работающий на узле B, будет перенесен на узел A. Таким образом, сервис всегда будет работать на узле с наивысшим приоритетом.

- Неупорядоченный домен (по умолчанию) — сервис может работать на любом узле в домене без каких-либо предпочтений.

- С возвратом — возвращает сервис на исходный узел, где он работал до сбоя. Обычно используется в упорядоченном домене при частых сбоях узла и помогает сократить число переносов, восстанавливая сервис не на промежуточном, а сразу на исходном узле.

Флаги «ordering», «restriction», «nofailback» помогают создать гибкие комбинации для контроля поведения сервисов при любых обстоятельствах.

3.1.1. Примеры доменов

В качестве примера рассмотрим кластер из нескольких узлов: {A, B, C, D, E, F, G}.

Упорядоченный, ограниченный домен {A, B, C}

Флаг «nofailback» не установлен: изначально сервис S будет запущен на узле A, так как он имеет наивысший приоритет. Если же сервис выполняется на узле C, и в это время узел A снова стал доступен — он будет перенесен с C обратно на A. Если все три узла отключены, S не сможет продолжить работу.

Флаг «nofailback» установлен: если сервис выполняется на узле C, и в это время узел A становится доступен — сервис останется на C. Если же C выйдет из строя, сервис будет восстановлен на узле A. Если все три узла отключены, S не сможет продолжить работу.

Неупорядоченный, ограниченный домен {A, B, C}

Для работы сервиса, закрепленного за этим доменом, должен быть доступен хотя бы один узел (например, B). Подключение узлов A и C никак не повлияет на работу сервиса, и он продолжит работу на узле B.

Упорядоченный, неограниченный домен {A, B, C}

Флаг «nofailback» не установлен: изначально сервис S будет работать на узле A, так как он имеет наивысший приоритет. Если A недоступен, будет выбран узел B, и наконец — C. Если узел A снова станет доступен, сервис будет перенесен с C обратно на A. Если отключены все три узла, будет выбран любой другой узел за пределами домена.

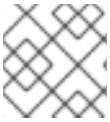
Флаг «nofailback» установлен: S изначально будет работать на узле с наивысшим приоритетом. Если узел A недоступен, будет выбран узел B, и наконец — C. Если позднее узел A снова станет доступен, сервис не будет переноситься обратно и останется на узле C.

Неупорядоченный, неограниченный домен {A, B, C}

Такой домен определяет предпочтительный круг узлов для запуска сервиса. Все узлы в домене равноправны, поэтому сервис не будет возвращаться на узел, где он работал до сбоя.

3.2. СЕРВИСНАЯ ПОЛИТИКА

Правила запуска сервисов на узлах кластера определяются сервисной политикой RGManager. В зависимости от индивидуальных требований правила могут корректироваться.



ПРИМЕЧАНИЕ

Приведенные здесь правила также применимы к ресурсам виртуальных машин.

3.2.1. Запуск сервиса

По умолчанию сервисы запускаются при запуске **rgmanager**.

- **`autostart`** (по умолчанию) — запуск сервиса при запуске **rgmanager**. Нулевое значение отключит сервис.

3.2.2. Возобновление работы

Эта политика определяет поведение сервиса в случае сбоя узла.

- **`restart`** (по умолчанию) — перезапуск на том же узле. Если не удалось, сервис будет перенесен на другой узел (см. **`relocate`**).
- **`relocate`** — запуск на другом узле. Если не удалось, сервис будет остановлен.
- **`disable`** — остановка сервиса.
- **`restart-disable`** — попытка перезапуска сервиса с отключением в случае неудачи.

3.2.3. Параметры перезапуска

Для политики перезапуска можно настроить дополнительные параметры:

`max_restarts` — максимальное число попыток, после чего сервис будет перенесен на другой узел.

`restart_expire_time` — время, на протяжении которого **rgmanager** будет пытаться перезапустить сервис.

Вместе эти значения контролируют число попыток на протяжении заданного периода времени.

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

В этом примере **rgmanager** попытается перезапустить сервис три раза в течение **5** минут. После четвертой попытки, которая будет предпринята по истечении **300** секунд, сервис будет восстановлен на другом узле.

ПРИМЕЧАНИЕ

Эти параметры не могут использоваться по отдельности.

3.3. ДЕРЕВО РЕСУРСОВ

Структура группы ресурсов может быть представлена в виде дерева:

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

```
        </fs>
        <ip address="10.1.1.2" .../>
    </service>
```

- Дерево ресурсов представляет собой XML-представление с корневым элементом **<service>**. Еще раз напомним, что в этом руководстве термины «дерево ресурсов», «группа ресурсов» и «сервис» являются взаимозаменяемыми. С точки зрения **rgmanager**, дерево представляет собой единый объект, элементы которого запускаются на одном узле.
- В приведенном примере **<fs name="myfs" ...>** и **<ip address="10.1.1.2" .../>** расположены на одном уровне иерархии.
- **<fs name="myfs" ...>** является родителем **<script name="script_child"/>**.
- **<script name="script_child"/>** является потомком **<fs name="myfs" ...>**.

3.3.1. Взаимосвязи и порядок запуска ресурсов

Правила взаимоотношений между родителями и потомками довольно просты:

- Родительские ресурсы запускаются перед потомками.
- Прежде чем остановить родительский ресурс, необходимо завершить работу его потомков.
- Из вышесказанного следует, что потомки полностью зависят от родительских ресурсов.
- При оценке состояния ресурса учитывается состояние его потомков.

3.4. УПРАВЛЕНИЕ СЕРВИСАМИ

Ниже обсуждаются операции управления кластерными сервисами.

3.4.1. Команды управления сервисами

Управление сервисами осуществляется при помощи нескольких простых команд. Обратите внимание, что команда **`migrate`** применима только к виртуальным машинам.

- **`enable`** — запуск сервиса в соответствии с сервисной политикой. Если не задан ни предпочтительный узел, ни резервный домен, сервис будет запущен на том же узле, где работает **`clusvcadm`**. Если попытка запуска завершилась неудачей, будет инициирована операция переноса (см. **`relocate`**).
- **`disable`** — остановка. Это единственно доступное действие для сервисов в состоянии *failed*.
- **`relocate`** — перенос сервиса на другой узел. Дополнительно можно определить предпочтительный узел. Если предпочтительный узел недоступен, будет выбран другой узел. Если не удалось возобновить работу сервиса на других узлах, он будет перезапущен на исходном узле. Если и эта попытка завершилась неудачей, сервис будет остановлен.
- **`stop`** — остановка сервиса с переходом в состояние *stopped*.
- **`migrate`** — миграция виртуальной машины на другой узел. Если миграция завершилась неудачей, виртуальная машина может остаться в состоянии *started* на исходном узле или перейти в состояние *failed*.

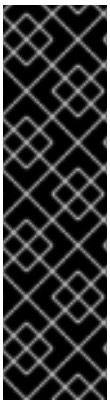
3.4.1.1. Freeze

Чтобы минимизировать простои при обновлении **rgmanager**, CMAN и других серверных программ, сервис можно временно заморозить.

Например, чтобы провести плановое обслуживание базы данных, можно заморозить сервис, остановить базу данных, выполнить работы по обслуживанию, перезапустить базу данных и возобновить работу сервиса.

3.4.1.1.1. Характеристики замороженного сервиса

- проверка состояния отключена;
- операции запуска не выполняются;
- операции остановки не выполняются;
- операции переноса между узлами не выполняются.



ВАЖНО

Ниже приведены правила, несоблюдение которых может привести к переносу ресурсов на разные узлы.

- Не останавливайте все экземпляры rgmanager при наличии замороженных сервисов (исключение составляет перезагрузка узлов перед перезапуском rgmanager).
- Прежде чем вывести сервис из замороженного состояния, убедитесь, что сервер, на котором он выполнялся, снова подключился и запустил rgmanager.

3.4.2. Состояние сервиса

Ниже приведен список возможных состояний сервисов под управлением RGManager.

- **disabled** – сервис отключен.
- **failed** – сервис переходит в нерабочее состояние при неудачной попытке остановки. Из этого состояния его выведет только команда **disable**. Прежде чем его полностью отключить, администратор должен будет освободить используемые им ресурсы (отключить файловые системы и т.п.)
- **stopped** – обычно сервис останавливается с целью проверки его состояния перед перезапуском на другом узле. Это временное состояние, которое может контролироваться администратором.
- **recovering** – означает, что кластер пытается восстановить сервис. Администратор может отключить сервис, чтобы его не восстанавливать.
- **started** – сервис запущен. Команды **relocate**, **stop**, **disable** и **migrate** могут изменить это состояние.

ПРИМЕЧАНИЕ

Состояние **started** включает промежуточные этапы **starting** и **stopping**.

3.5. УПРАВЛЕНИЕ ВИРТУАЛЬНЫМИ МАШИНАМИ

RGManager использует несколько другой подход к управлению виртуальными машинами по сравнению с другими кластерными сервисами.

3.5.1. Управляющие операции

Если виртуальные машины создаются как кластерные ресурсы, то и управлять ими следует при помощи clusvcadm или других кластерных утилит. Стандартные операции:

- запуск;
- остановка;
- проверка состояния;
- перенос;
- восстановление.

Глава 7, *Виртуализация в кластере* содержит более подробную информацию.

3.5.2. Миграция

Миграция минимизирует простои при переносе виртуальных машин между узлами, так как их не надо будет останавливать и снова перезапускать. Эта операция уникальна для виртуальных машин.

RGManager поддерживает оба метода миграции:

- Живая миграция — перенос виртуальной машины с одного узла на другой без видимой остановки. Это возможно благодаря тому, что виртуальная машина продолжает работать на старом узле до тех пор, пока ее образ памяти не будет полностью скопирован на новый узел. Непрерывность работы достигается ценой снижения производительности виртуальной машины и замедления процесса миграции.
- Миграция с остановкой — виртуальная машина замораживается до тех пор, пока состояние памяти не будет скопировано на новый узел. Такой подход максимально сокращает время миграции.

Выбор метода зависит от требований производительности: если при остановке виртуальной машины процесс миграции занимает 8 секунд, то во время живой миграции машина будет работать, но ее производительность снизится на 29 секунд.

ВАЖНО

Виртуальную машину можно включить как компонент сервиса, но тогда операция миграции будет недоступна.

Дополнительно: миграция виртуальных машин на базе KVM требует тщательной конфигурации ssh.

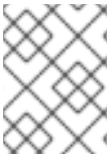
3.5.3. Особенности работы виртуальных машин под управлением RGManager

Далее обсуждаются методы оптимизации управления виртуальными машинами в RGManager.

3.5.3.1. Согласование состояния виртуальных машин

Если **clusvcadm** попытается запустить виртуальную машину, которая уже выполняется, rgmanager установит для нее статус **started**.

Если для миграции машины использовались не кластерные утилиты, а **virsh**, rgmanager установит для нее статус **started**.



ПРИМЕЧАНИЕ

RGManager не предупреждает о выполнении разных экземпляров одной виртуальной машины на разных узлах.

3.5.3.2. Временные домены libvirt

RGManager поддерживает временные домены libvirt, что позволяет создавать и удалять виртуальные машины на лету, снижая вероятность многократного запуска одной и той же машины не кластерными утилитами наподобие **virsh**.

Чтобы не синхронизировать **/etc/libvirt/qemu** вручную на каждом узле, скопируйте XML-описания доменов в файловую систему кластера.

3.5.3.2.1. Другие особенности управления

Добавление и удаление виртуальных машин из **cluster.conf** не означает их автоматический запуск и остановку.

Выбор более подходящего узла происходит во время миграции, что помогает сократить простои.

3.5.4. Неподдерживаемые события

RGManager не поддерживает:

- Изменение конфигурации и состояния виртуальных машин не кластерными утилитами, такими как **virsh** и **xm**. Операции проверки состояния (**virsh list**, **virsh dumpxml** и т.п.) разрешены.
- Миграцию виртуальной машины на узел, на котором не выполняется **rgmanager**. Так как **rgmanager** ничего не знает о новом узле, он перезапустит машину на старом узле, что приведет к тому, что в кластере будут работать два экземпляра одной и той же машины.

3.6. КОМАНДЫ УПРАВЛЕНИЯ РЕСУРСАМИ

Агенты ресурсов должны поддерживать команды:

- **start** — запуск ресурса;
- **stop** — остановка ресурса;
- **status** — проверка состояния;
- **metadata** — возвращает метаданные XML агента OCF.

3.6.1. Коды возврата

Спецификация OCF поддерживает целый набор кодов возврата, но для **rgmanager** актуальны только два:

0 (успех)

Вызов **`stop`** для остановленного ресурса вернет ноль.

Вызов **`start`** для работающего ресурса тоже вернет ноль.

ненулевое значение (неудача)

Если команда **`stop`** вернула ненулевое значение, сервис перейдет в состояние **`failed`**, из которого его надо будет вывести вручную.

ГЛАВА 4. ИЗОЛЯЦИЯ УЗЛОВ

Изоляция узла (англ. fencing) — отключение неисправного узла от кластерного хранилища с целью поддержки целостности данных. До тех пор пока узел не будет благополучно изолирован, все операции ввода-вывода в кластере будут приостановлены. Это позволяет снизить вероятность повреждения данных в хранилище. Изоляцию узла проводит специальный демон **fenced**.

Как только CMAN обнаружит сбой узла, он сообщит об этом другим кластерным подсистемам, в том числе **fenced**, который тут же изолирует узел. Другие подсистемы будут действовать в соответствии с ситуацией — так, DLM и GFS2 приостановят работу до тех пор, пока **fenced** не сообщит об успешном отключении узла, после чего DLM освободит его блокировки, GFS2 восстановит журнал — и они продолжат работу.

Отключение узла от кластера осуществляется внешними устройствами, взаимодействие с которыми осуществляют агенты. При сбое узла **fenced** свяжется с агентом, который сможет изолировать узел при помощи соответствующего устройства. Агенты и устройства изоляции настраиваются в файле конфигурации кластера.

Red Hat High Availability поддерживает разные способы изоляции:

- Отключение питания — использует контроллер питания для обесточивания проблемного узла.
- Отключение от хранилища — закрывает порт Fibre Channel между узлом и хранилищем.
- Другое оборудование для ограничения ввода-вывода и отключения питания: IBM BladeCenter, PAP, DRAC/MC, HP ILO, IPMI, IBM RSA II и пр.

Рисунок 4.1, «Отключение питания» демонстрирует пример, в котором программа изоляции на узле A связывается с контроллером, который выключит узел D. Рисунок 4.2, «Отключение от хранилища» рассматривает ситуацию, в которой коммутатор Fibre Channel закроет порт для узла D, тем самым отключив его от хранилища.

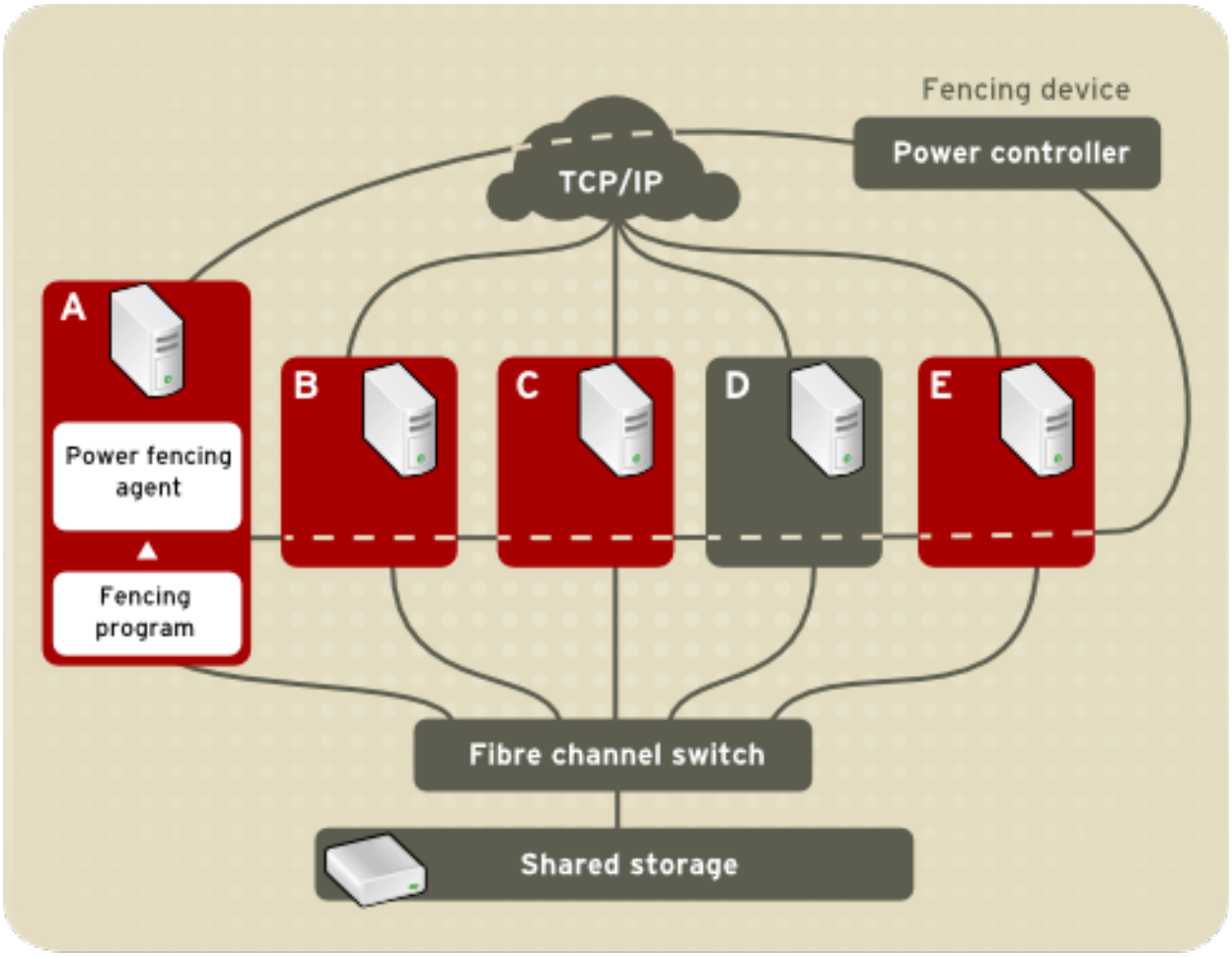


Рисунок 4.1. Отключение питания

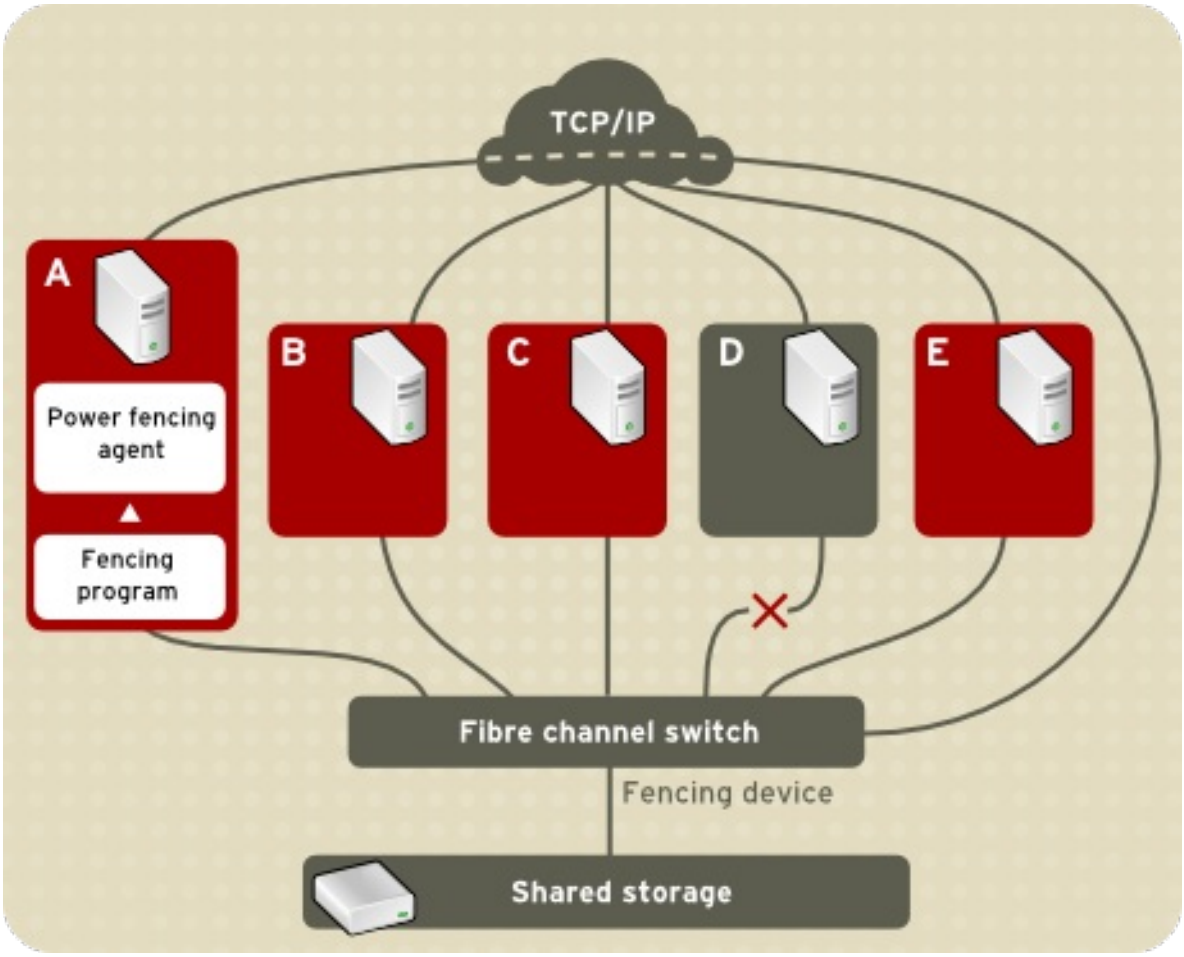


Рисунок 4.2. Отключение от хранилища

Чтобы настроить метод изоляции, для каждого узла в файле конфигурации кластера надо определить имя метода, агент и устройство изоляции.

Если компьютер оснащен двумя источниками питания, для его изоляции надо будет задействовать два устройства (см. Рисунок 4.3, «Изоляция узла с двумя источниками питания»). Аналогично, если узел использует несколько маршрутов для подключения к хранилищу, надо будет настроить по одному устройству изоляции на каждый маршрут (см. Рисунок 4.4, «Изоляция узла с двумя соединениями Fibre Channel»).

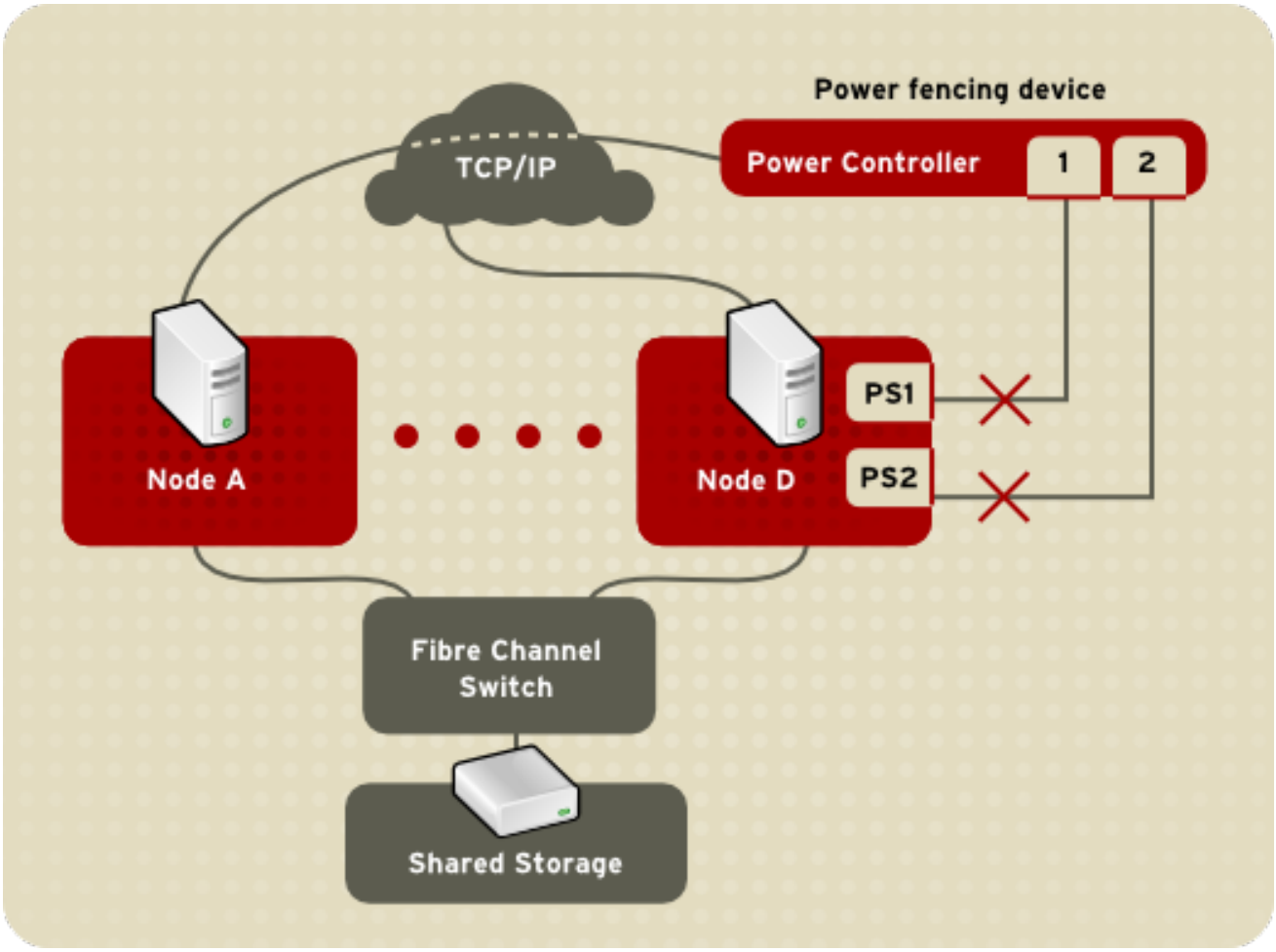


Рисунок 4.3. Изоляция узла с двумя источниками питания

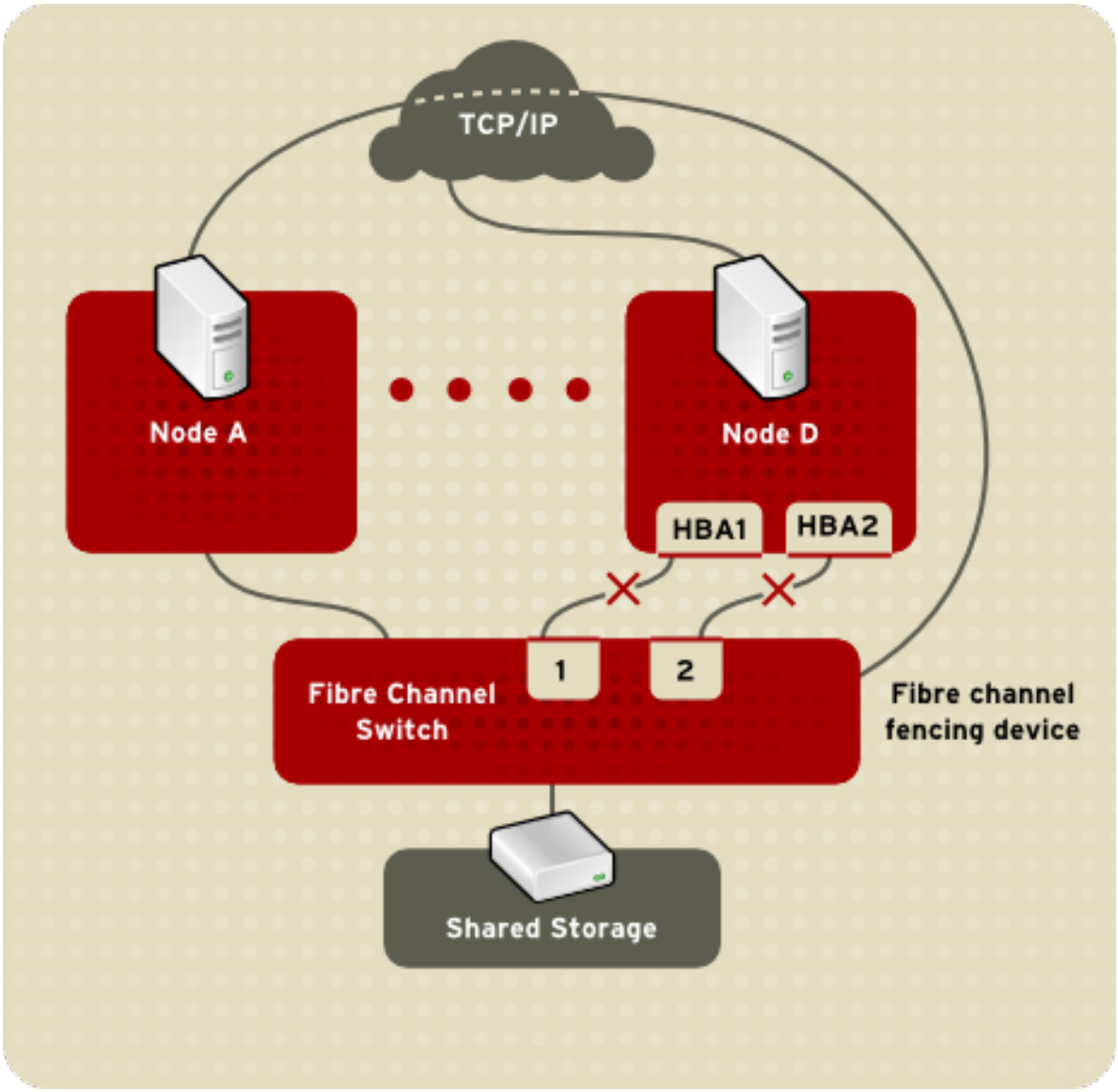


Рисунок 4.4. Изоляция узла с двумя соединениями Fibre Channel

Для одного узла можно настроить несколько методов изоляции. Методы будут выбираться *каскадно* в соответствии с их порядком в файле конфигурации: если первый метод не сработал, будет выбран следующий и т.д. Если же ни один метод не смог изолировать узел, снова будет выбран первый метод и цикл повторится. Это будет продолжаться до тех пор, пока узел не будет благополучно отключен.

Подробную информацию можно найти в руководстве *Администрирование кластера*.

ГЛАВА 5. УПРАВЛЕНИЕ БЛОКИРОВКАМИ

Синхронизация доступа компонентов кластерной инфраструктуры к общим ресурсам осуществляется с помощью блокировок. В кластере Red Hat эти функции выполняет механизм распределенного управления блокировками (DLM, Distributed Lock Manager).

Менеджер блокировок регулирует доступ кластерному хранилищу (например, GFS2), предотвращая несогласованное изменение данных разными узлами.

DLM работает на всех узлах кластера и реализует распределенный механизм управления блокировками. GFS2 использует DLM для синхронизации доступа к файловой системе, CLVM — для синхронизации обновлений томов LVM, а `rgmanager` — для согласования состояний сервисов.

5.1. МОДЕЛЬ БЛОКИРОВАНИЯ DLM

Модель блокирования DLM предусматривает целый набор режимов блокировки для синхронной и асинхронной обработки запросов. Ресурсы могут блокироваться одним процессом монопольно или несколькими процессами в совместном режиме.

DLM может блокировать файлы, структуры данных, базы данных, исполняемые процедуры и другие объекты. Размер объекта определяет степень детализации — например, блокирование отдельных записей в базе данных обеспечивает более тонкую детализацию по сравнению с блокированием всей базы данных.

DLM поддерживает:

- 6 режимов блокировки в зависимости от уровня доступа;
- изменение режимов блокировки;
- синхронную обработку запросов блокировки;
- асинхронную обработку;
- блоки состояния, откуда процессы смогут получать информацию о состоянии заблокированного ресурса.

DLM использует собственные механизмы для согласования блокировок между узлами, их восстановления при сбое узла и переноса блокировок на узел при его подключении к кластеру. Так как DLM интегрируется в существующую кластерную инфраструктуру, он предъявляет к ней собственные требования:

- Подконтрольный узел должен входить в состав кластера.
- В кластере должен быть набран необходимый кворум.
- DLM должен иметь возможность обращаться к узлу по IP-адресу. Обычно для взаимодействия между узлами DLM использует TCP/IP, что накладывает определенные ограничения и требует, чтобы узел имел только один IP-адрес. Чтобы разрешить узлам иметь несколько IP-адресов, для DLM вместо TCP/IP можно настроить SCTP.

При выполнении вышеперечисленных условий DLM сможет работать в любых кластерных окружениях с открытым и закрытым кодом. Однако надо учитывать, что в разных окружениях тестирование DLM проводится в разных объемах, и это тоже может повлиять на принятие окончательного решения.

5.2. СОСТОЯНИЕ БЛОКИРОВКИ

Состояние блокировки отражает текущую стадию обработки запроса блокировки:

- Установлена — блокировка установлена.
- Изменение режима — клиент пытается изменить режим блокировки, но новый режим не совместим со старым.
- Отказ — не удалось установить блокировку вследствие конфликта с существующей блокировкой.

Состояние блокировки выбирается в зависимости от запрашиваемого режима и текущих блокировок ресурса другими процессами.

ГЛАВА 6. КОНФИГУРАЦИЯ И АДМИНИСТРИРОВАНИЕ КЛАСТЕРА

Настройки кластера хранятся в файле **`/etc/cluster/cluster.conf`** в формате XML, который содержит несколько частей:

- Общие параметры — имя кластера, версия файла, время задержки изоляции после сбоя узла и при добавлении узлов в кластер.
- Узлы кластера — для каждого узла определяется имя, идентификатор, число голосов кворума и метод изоляции узла.
- Устройства изоляции — параметры устройств определяются их типом, например для внешнего блока питания надо будет указать имя, IP-адрес, логин и пароль.
- Ресурсы — список ресурсов для создания кластерных сервисов. Эта секция содержит определения резервных доменов и ресурсов (например, IP-адрес или файловые системы).

Проверка формата осуществляется в соответствии со схемой **`/usr/share/cluster/cluster.rng`** при запуске кластера и перезагрузке конфигурации. С помощью команды **`ccs_config_validate`** это можно сделать в любое время.

Описание схемы с комментариями можно найти в **`/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html`**, например: **`/usr/share/doc/cman-3.0.12/cluster_conf.html`**.

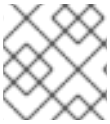
В ходе проверки будут выявлены ошибки:

- форматирования XML;
- параметров конфигурации;
- значений параметров.

6.1. СРЕДСТВА АДМИНИСТРИРОВАНИЯ КЛАСТЕРА

Red Hat High Availability предоставляет несколько инструментов для управления кластерами.

- **Conga** — многофункциональный комплект для настройки и управления кластерами Red Hat. За подробной информацией обратитесь к *руководству по настройке и управлению Red Hat High Availability.*
  - **Luci** — программный сервер для централизованного управления кластерами.
  - **Ricci** — агент, который работает на всех подконтрольных узлах и синхронизирует кластерную конфигурацию.
- Начиная с Red Hat Enterprise Linux 6.1 управление кластерами Red Hat может осуществляться с помощью команды **`ccs`**. За подробной информацией обратитесь к документу *Администрирование кластера.*



ПРИМЕЧАНИЕ

`system-config-cluster` был исключен из RHEL 6.

ГЛАВА 7. ВИРТУАЛИЗАЦИЯ В КЛАСТЕРЕ

На базе отказоустойчивого кластера Red Hat Enterprise Linux 6 можно строить комплексные решения, объединяя его с технологией виртуализации. В этой главе обсуждаются два варианта виртуализации на основе высокодоступных решений Red Hat Enterprise Linux.

В первом случае виртуальные машины создаются на узлах отказоустойчивого кластера RHEL. Виртуальными машинами, как и другими кластерными ресурсами, будет управлять `rgmanager`. Во втором случае виртуальные машины сами объединяются в кластер.

- Виртуальные машины как кластерные ресурсы
- Кластеры виртуальных машин

7.1. ВИРТУАЛЬНЫЕ МАШИНЫ КАК КЛАСТЕРНЫЕ РЕСУРСЫ

Red Hat High Availability и RHEV предоставляют механизмы для обеспечения непрерывного доступа к виртуальным машинам, которые создаются как кластерные ресурсы. Функциональность этих решений частично дублируется, поэтому в каждом случае к выбору продукта надо подходить индивидуально. Далее обсуждаются некоторые особенности, которые помогут сделать правильный выбор.

Для физических серверов и виртуальных машин надо учесть следующее:

- Для кластера с большим числом узлов, на базе которых планируется создать большое число виртуальных машин, RHEV сможет предложить более гибкие алгоритмы для размещения виртуальных машин в зависимости от процессорных ресурсов, памяти и уровня нагрузки сервера.
- При наличии небольшого числа узлов, на базе которых создается ограниченное число виртуальных машин, лучше выбрать Red Hat High Availability в силу его облегченной инфраструктуры. Дело в том, что для создания даже самой простой схемы RHEV потребуется как минимум 4 узла: два узла для обеспечения непрерывного доступа к серверу RHEVM и две платформы виртуализации.
- На самом деле, нет четких правил, предписывающих, какое именно число узлов считается достаточным для выбора RHEV. При этом следует помнить, что HA-кластер может содержать не больше 16 узлов. В любом случае, архитектура любого кластера, содержащего более 8 узлов, должна быть официально одобрена Red Hat.

Для виртуальных машин надо отметить следующее:

- Если службы виртуальных машин обращаются к общей инфраструктуре, можно выбрать и RHEL HA, и RHEV.
- Если необходимо обеспечить непрерывный доступ к ограниченному набору критических служб в виртуальной системе, можно выбрать и RHEL HA, и RHEV.
- Для создания инфраструктуры, ориентированной на быструю установку виртуальных машин, лучше выбрать RHEV.
    - RHEV допускает динамическое изменение структуры кластера и облегчает добавление новых виртуальных машин.
    - Red Hat High Availability не обладает такой гибкостью — кластер создается с фиксированным числом виртуальных машин, которое не рекомендуется менять.

- Red Hat High Availability не подходит для создания облачных схем в силу его статической конфигурации и ограниченного числа узлов (не больше 16).

RHEL 5 поддерживает две платформы виртуализации: Xen (впервые представлен в RHEL 5.0) и KVM (добавлен в RHEL 5.4).

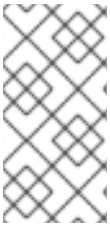
RHEL 6 поддерживает только KVM.

Кластер RHEL 5 Advanced Platform поддерживает и Xen, и KVM.

В отказоустойчивом кластере RHEL6 в качестве гипервизора может использоваться KVM.

Далее перечислены возможные варианты развертывания виртуальных машин в кластерных схемах Red Hat.

- На серверах RHEL 5.0+ в кластере RHEL Advanced Platform поддерживается гипервизор Xen;
- На серверах RHEL 5.4 в кластере RHEL Advanced Platform была добавлена экспериментальная поддержка виртуальных машин KVM;
- Версии RHEL 5.5+ полностью поддерживают виртуальные машины KVM.
- RHEL 6.0+ позволяет настроить виртуальные машины KVM как кластерные ресурсы в RHEL 6 High Availability;
- RHEL 6.0+ больше не поддерживает Xen, поэтому RHEL 6 High Availability тоже не будет работать с виртуальными машинами Xen.



ПРИМЕЧАНИЕ

Подробное описание возможных комбинаций программного обеспечения в кластерных схемах можно найти в базе знаний Red Hat:

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

Прежде чем настроить гостевую систему как кластерный сервис, убедитесь, что она поддерживается гипервизором узла. Xen и KVM поддерживают RHEL (RHEL3, RHEL4, RHEL5) и некоторые редакции Microsoft Windows. Полный список поддерживаемых операционных систем можно найти в документации RHEL.

7.1.1. Общие рекомендации

- До RHEL 5.3 для управления гостевыми системами Xen (domU) `rgmanager` использовал внутренние механизмы Xen. В RHEL 5.4 эти функции стал выполнять libvrit, предоставляя единый интерфейс для управления и Xen, и KVM. Последующие версии RHEL включают множество обновлений, поэтому прежде чем приступить к конфигурации сервисов под управлением Xen, рекомендуется обновить программное обеспечение серверов хотя бы до RHEL 5.5.
- Полнофункциональная поддержка виртуальных машин на базе KVM была впервые добавлена в RHEL 5.5, поэтому программное обеспечение серверов надо будет обновить до RHEL 5.5.
- Прежде чем развернуть кластер, следует установить последние обновления и исправления RHEL.
- Виртуальные машины должны находиться под управлением одного и того же гипервизора: Xen или KVM.

- Оборудование узла должно выдерживать повышение нагрузки, которое произойдет при переносе гостевых систем с других узлов. В этой ситуации важно не перерасходовать ресурсы виртуальных процессоров и памяти, в противном случае производительность кластера может пострадать.

- Не следует запускать и останавливать виртуальные машины, подконтрольные **rgmanager**, при помощи команд **xm** и **libvirt** (**virsh**, **virsh-manager**), так как это делается в обход стека управления кластером.

- Имена виртуальных машин должны быть уникальными в пределах кластера. **Libvirtd** проверяет имена отдельно на каждом узле, но не во всем кластере. Если вы собираетесь продублировать виртуальную машину вручную, не забудьте изменить имя в ее файле конфигурации.

7.2. КЛАСТЕРЫ ВИРТУАЛЬНЫХ МАШИН

В этой схеме виртуальные машины сами объединяются в отказоустойчивый кластер. Преимущество такой реализации состоит в том, что она обеспечивает непрерывный доступ к службам и приложениям в гостевой системе, так как в случае сбоя виртуальной машины они смогут возобновить работу на другой машине. Поведение узлов в виртуальном кластере практически не отличается от обычного кластера.

Далее обсуждаются особенности поддержки виртуальных кластеров на платформах RHEL. В приведенных примерах гостевые системы RHEL 6 объединяют функции High Availability с дополнением Resilient Storage, предоставляющим инструменты для повышения отказоустойчивости подсистемы хранения данных (GFS2, clvmd, cmirror).

- На платформах RHEL 5.3+ Xen можно построить виртуальные кластеры с операционными системами RHEL 5.3+.
    - Изоляция неисправных узлов в таком кластере осуществляется при помощи **fence_xvm** или **fence_scsi**.
    - Для нормальной работы **fence_xvm** необходимо, чтобы кластер физических серверов поддерживал **fence_xvmd**, а узлы виртуального кластера использовали агент изоляции **fence_xvm**.
    - Общее хранилище может быть построено на блочных устройствах iSCSI или Xen.
- Платформы RHEL 5.5+ KVM не поддерживают виртуальные кластеры.
- Платформы RHEL 6.1+ KVM поддерживают виртуальные кластеры с операционными системами RHEL 6.1+ и RHEL 5.6+.
    - В состав кластера могут входить и физические, и виртуальные узлы.
    - В кластерах RHEL 5.6+ изоляция неисправных узлов осуществляется с помощью **fence_xvm** или **fence_scsi**.
    - В кластерах RHEL 6.1+ изоляция узлов осуществляется с помощью **fence_xvm** (из пакета **fence-virt**) или **fence_scsi**.
    - Если для изоляции узлов используется агент **fence_virt** или **fence_xvm**, то на серверах RHEL 6.1+ KVM должен работать **fence_virtd**.
    - **fence_virtd** может работать в трех режимах:

        - В автономном режиме связи гостевых систем с сервером жестко определены, поэтому живая миграция невозможна.
        - Отслеживание живой миграции средствами Openais Checkpoint. Для этого необходимо, чтобы физические узлы были объединены в кластер.
        - Отслеживание живой миграции средствами QMF (Qpid Management Framework) из пакета `libvirt-qpid`. Это не требует наличия кластера физических узлов.
    - Общее хранилище кластера может быть построено на блочных устройствах iSCSI или KVM.
- На платформах RHEV-M (Red Hat Enterprise Virtualization Management) 2.2+ и 3.0 могут создаваться кластеры виртуальных машин RHEL 5.6+ и RHEL 6.1+.
    - Все гостевые операционные системы в кластере должны быть одного типа: RHEL 5.6+ или RHEL 6.1+.
    - В состав кластера могут входить и физические, и виртуальные узлы.
    - За изоляцию узлов на платформе RHEV-M 2.2+ отвечает агент `fence_scsi`, на RHEV-M 3.0 – `fence_scsi` и `fence_rhevm`.
        - Для нормальной работы `fence_scsi` необходимо, чтобы серверы iSCSI поддерживали алгоритмы постоянного резервирования SCSI-3. Эту информацию следует заранее уточнить у производителя. Так, сервер iSCSI, входящий в стандартную поставку Red Hat Enterprise Linux, не поддерживает постоянное резервирование SCSI-3 и, как следствие, не сможет использовать `fence_scsi`.
- VMware vSphere 4.1, VMware vCenter 4.1, VMware ESX и ESXi 4.1 поддерживают виртуальные кластеры с гостевыми операционными системами RHEL 5.7+ и RHEL 6.2+. VMware vSphere 5.0, vCenter 5.0, ESX и ESXi 5.0 тоже можно использовать, но исходная версия VMware vSphere 5.0 включает неполную схему WDSL, поэтому `fence_vmware_soap` не будет работать в исходной реализации. О том, как снять это ограничение, можно узнать из базы знаний Red Hat: https://access.redhat.com/knowledge/.
    - Все гостевые операционные системы в кластере должны быть одного типа: RHEL 5.7+ или RHEL 6.1+.
    - В состав кластера могут входить и физические, и виртуальные узлы.
    - Для нормальной работы `fence_vmware_soap` потребуется загрузить пакет с дополнительными Pearl API с сайта VMware и установить его в гостевых системах RHEL.
    - В противном случае можно настроить изоляцию `fence_scsi`.
    - Общее хранилище может быть построено на RAW-дисках VMware и блочных устройствах iSCSI.
    - В виртуальных кластерах на базе VMware ESX изоляция узлов должна осуществляться средствами `fence_vmware_so_ap` или `fence_scsi`.
- Виртуальные кластеры Hyper-V не поддерживаются.

7.2.1. Fence_scsi и общее хранилище iSCSI

- Во всех перечисленных выше схемах на базе исходных решений хранения данных можно создать общее хранилище iSCSI и настроить агент изоляции **fence_scsi**.

- **fence_scsi** изолирует вышедшие из строя узлы, отключая их доступ к хранилищу путем удаления их ключей регистрации с дисков. Для этого цели iSCSI должны поддерживать постоянное резервирование iSCSI-3 и команду "PREEMPT AND ABORT". Эту информацию следует заранее уточнить у производителя.

- Программное обеспечение сервера iSCSI в стандартной сборке RHEL не поддерживает постоянное резервирование SCSI-3 и не сможет использовать **fence_scsi**. Возможно, вы решите остановиться на других методах изоляции – **fence__vmware** или **fence_rhevm**.

- Если на всех узлах виртуального кластера настроен **fence_scsi**, то объединять физические серверы (RHEL 5 Xen/KVM и RHEL 6 KVM) в кластер не обязательно.

- Совместное использование общего хранилища iSCSI с подключенными напрямую устройствами не допускается.

7.2.2. Общие рекомендации

- Как говорилось выше, прежде чем развернуть виртуализацию в кластере, следует обновить программное обеспечение серверов и гостевых систем, установив последние версии пакетов RHEL.

- Виртуальными машинами в составе кластера должен управлять один гипервизор.

- Не следует создавать виртуальный кластер на базе единственного физического сервера, так как в случае его отказа весь кластер выйдет из строя. Такая схема вполне подойдет для создания экспериментального прототипа кластера, но для обеспечения отказоустойчивости разместите виртуальные машины на нескольких узлах.

- Другие рекомендации:

    - Если на каждом сервере размещается по два узла виртуального кластера, то в случае его сбоя оба узла выйдут из строя. Очевидно, что максимальной надежности можно достичь, разместив каждый узел кластера на отдельном сервере, но это не является обязательным требованием.

    - Создание на одних и тех же физических узлах нескольких кластеров, использующих агенты изоляции **fence_xvm/fence_xvmd** и **fence_virt/fence_virtd**, не поддерживается.

    - Одновременная работа независимых виртуальных кластеров в одной группе физических серверов допускается, если они используют общее хранилище iSCSI с агентом **fence_scsi** или VMware (ESX/ESXi и vCenter) с **fence_vmware**.

    - Создание независимых виртуальных машин наравне с виртуальным кластером на тех же физических серверах не воспрещается, но если серверы сами объединены в кластер и один из них вышел из строя, он будет изолирован, а независимая виртуальная машина будет отключена.

    - Оборудование узла должно выдерживать повышение нагрузки без перерасхода памяти и виртуальных процессоров, в противном случае производительность кластера пострадает.

ПРИЛОЖЕНИЕ A. ИСТОРИЯ ПЕРЕИЗДАНИЯ

Издание 1-15.2 **Wed Apr 29 2015** **Poyarkova Yuliya**

Перевод на русский язык.

Издание 1-15.1 **Wed Apr 29 2015** **Poyarkova Yuliya**

Синхронизация с XML 1-15.

Издание 1-15 **Tue Dec 16 2014** **Steven Levine**

Интеграция функции сортировки на странице приветствия RHEL 6.

Издание 1-13 **Wed Oct 8 2014** **Steven Levine**

Редакция для Red Hat Enterprise Linux 6.6.

Издание 1-12 **Thu Aug 7 2014** **Steven Levine**

Бета-выпуск для Red Hat Enterprise Linux 6.6.

Издание 1-11 **Fri Aug 1 2014** **Steven Levine**

Решает: #852720

Редакционные правки.

Издание 1-10 **Fri Jun 6 2014** **Steven Levine**

Черновой вариант для Red Hat Enterprise Linux 6.6.

Издание 1-7 **Wed Nov 20 2013** **John Ha**

Подготовлено к выпуску Red Hat Enterprise Linux 6.5.

Издание 1-4 **Mon Feb 18 2013** **John Ha**

Подготовлено к выпуску Red Hat Enterprise Linux 6.4.

Издание 1-3 **Mon Jun 18 2012** **John Ha**

Подготовлено к выпуску Red Hat Enterprise Linux 6.3.

Издание 1-2 **Fri Aug 26 2011** **John Ha**

Обновление для 6.2.

Издание 1-1 **Wed Nov 10 2010** **Paul Kennedy**

Исходная версия