Red Hat Enterprise Linux 6

Logical Volume Manager Administration

Manual do Administrador do LVM

Edição 1

Last Updated: 2017-10-13

Red Hat Enterprise Linux 6 Logical Volume Manager Administration

Manual do Administrador do LVM
Edição 1

Landmann
rlandmann@redhat.com

Nota Legal

Resumo

Este livro descreve o gerenciador do volume lógico LVM, incluindo as informações sobre a execução do LVM em um ambiente de cluster.

Índice

INTRODUÇÃO

1. SOBRE ESTE GUIA

Este livro descreve o Logical Volume Manager (LVM), incluindo informações sobre a execução do LVM em um ambiente de cluster.

2. PÚBLICO

Este livro é destinado para para ser usado por administradores de sistemas que gerenciam sistemas com um sistema operacional Linux. Ele requer familiaridade com o Red Hat Enterprise Linux 6 e administração de sistema de arquivo GFS2.

3. VERSÕES DE SOFTWARE

Tabela 1. Versões de Software

Software	Descrição
RHEL 6	refere-se ao RHEL 6 e posteriores
GFS2	refere-se ao GFS2 para RHEL6 e posteriores

4. DOCUMENTAÇÃO RELACIONADA

Para mais informações sobre como usar o Red Hat Enterprise Linux, consulte os seguintes recursos:

- *Guia de Instalação* — Documenta informações relevantes quanto à instalação do Red Hat Enterprise Linux 6.
- *Guia de Implementação* — Documenta informações relevantes quanto à implementação, configuração e administração do Red Hat Enterprise Linux 6.
- *Guia de Administração de Armazenamento*— Fornece instruções sobre como gerenciar de uma forma eficiente dispositivos de armazenamento e sistemas de arquivo no Red Hat Enterprise Linux 6.

Para mais informações sobre a Adição com Alta Disponibilidade (High Availability Add-On) e sobre a Adição de Armazenamento Resiliente (Resilient Storage Add-On) para Red Hat Enterprise Linux 6, consulte os seguintes recursos:

- *Visão Geral do High Availability Add-On Overview* — Fornece uma visão geral de alto nível de Adição de Alta Disponibilidade.
- *Administração de Cluster*— Fornece informações sobre instalação, configuração e gerenciamento da Adição de Alta Disponibilidade.

- *Sistema de Arquivo Global 2: Configuração e Administração*— Fornece informações sobre instalação, configuração e manutenção do Red Hat GFS2 (Red Hat Global File System 2), o qual está incluso no Resilient Storage Add-On (Adição de Armazenamento Resiliente).
- *DM Multipath* — Fornece informações sobre o uso do recurso Device-Mapper Multipath do Red Hat Enterprise Linux 6.
- *Linux Virtual Server Administration* — Fornece informações sobre configuração de sistemas de alto desempenho e serviços com a Adição de Balanceador de Carga (Load Balancer Add-On), um conjunto de componentes de software integrado que fornece Linux Virtual Servers (LVS) para balanceamento de carga de IP através um conjunto de servidores reais.
- *Notas de Lançamento* — Fornece informações sobre os lançamentos atuais dos produtos da Red Hat.

A documentação da Adição de Alta Disponibilidade e outros documentos da Red Hat estão disponíveis nas versões HTML, PDF e RPM no CD da Documentação do Red Hat Enterprise Linux e online em http://www.redhat.com/docs/.

5. PRECISAMOS DE SEU FEEDBACK!

Se você encontrar um erro de digitação neste manual ou se você souber alguma forma de melhorá-lo, adoraríamos saber! Por favor, submeta seu relatório no Bugzilla: http://bugzilla.redhat.com/ no produto **Red Hat Enterprise Linux 6** e componente **doc-Logical_Volume_Manager**. Ao submeter um relatório de erro, certifique-se de mencionar o identificador do manual: **`Logical_Volume_Manager_Administration(EN)-6 (2011-05-19-15:20)`**.

Se você tiver alguma sugestão para aprimorar esta documentação, tente ser o mais específico possível ao descrevê-lo. Se você encontrou algum erro, inclua o número da seção e um pouco do texto no qual está contido para que possamos encontrá-lo com facilidade.

CAPÍTULO 1. O GERENCIADOR DO VOLUME LÓGICO

Este capítulo fornece um resumo dos recursos do gerenciador do volume lógico LVM que são novos para o lançamento inicial e subsequentes do Red Hat Enterprise Linux 6. Desta maneira, este capítulo fornecerá uma visão geral de alto nível dos componentes do Gerenciador de Volume Lógico (LVM).

1.1. RECURSOS NOVOS E MODIFICADOS.

Esta seção lista os recursos novos e modificados do gerenciador de volume lógico LVM que estão inclusos no lançamento inicial e subsequentes do Red Hat Enterprise Linux 6.

1.1.1. Novos e Modificados Recursos do Red Hat Enterprise Linux 6.0

Red Hat Enterprise Linux 6.0 inclui a seguinte documentação e apresenta atualizações e mudanças.

- Você pode definir como o volume lógico espelhado se comporta no evento de uma falha de dispostivo com os parâmetros **`mirror_image_fault_policy`** e **`mirror_log_fault_policy`** na seção **`activation`** do arquivo **`lvm.conf`**. Quando este parâmetro for definido para **`remove`**, o sistema tentará remover o dispositivo defeituoso e rodar sem ele. Quando este parâmetro for definido para **`allocate`**, o sistema tentará remover o dispositio defeituoso e tentará alocar espaço em um dispositivo que será um substituto para o dispositivo com falha. Esta política age como a política **`remove`** caso nenhum dispositivo adequado e espaço possam ser alocados para a substituição. Para informações sobre políticas de falhas de espelho LVM, veja a Seção 4.4.3.1, “Política de Falha do Volume Lógico Espelhado”.

- Para o lançamento Red Hat Enterprise Linux 6, a pilha de E/S do Linux foi aprimorada para processar informações de limite de E/S fornecidas pelo fabricante. Isto permite que as ferramentas de gerenciamento de armazenamento, incluindo o LVM, otimize alocação de dados e acesso. Este suporte pode ser desabilitado, mudando os valores padrão do **`data_alignment_detection`** e **`data_alignment_offset_detection`** no arquivo **`lvm.conf`**, embora não seja recomendado a desativação deste suporte.

  Para informações sobre alinhamento de dados no LVM assim como informações sobre como mudar valores padrão do **`data_alignment_detection`** e **`data_alignment_offset_detection`**, veja a documentação online para o arquivo **`/etc/lvm/lvm.conf`** que também está documentado no Apêndice B, *Os arquivos de Configuração do LVM*. Para informações gerais sobre suporte da Pilha de E/S e limites de E/S no Red Hat Enterprise Linux 6, veja *Storage Administration Guide*.

- No Red Hat Enterprise Linux 6, o Mapeador de Dispositivo fornece suporte direto para integração do **`udev`**. Isto sincroniza o Mapeador de Dispositivo com todos os **`udev`** em processamento, relacionandos os dispositivos do Device Mapper (Mapeador de Dispositivo), incluindo os dispositivos LVM. Para informações sobre o suporte do Device Mapper para o gerenciador do dispositivo **`udev`**, veja a Seção A.3, “Suporte do Mapeador de Dispositivo para o Gerenciador de Dispostivo udev”.

- Para o lançamento Red Hat Enterprise Linux 6, você poderá usar o comando **`lvconvert --repair`** para reparar um espelho após falha de disco. Isto traz o espelho de volta ao seu estado consistente. Para informações sobre o comando **`lvconvert --repair`**, veja a Seção 4.4.3.3, “Reparando um Dispositivo Lógico Espelhado”.

- A partir do lançamento do Red Hat Enterprise Linux 6, você pode usar a opção **--merge** do comando **lvconvert** para mesclar um snapshot em seu volume original. Para informações sobre como mesclar snapshots, veja a Seção 4.4.5, “Incorporando Volumes Snapshots”.

- A partir do lançamento do Red Hat Enterprise Linux 6, você pode usar o argumento **--splitmirrors** do comando **lvconvert** para dividir uma imagem redundante de um volume lógico espelhado para formar um novo volume lógico. Para informações sobre como usar esta opção, veja a Seção 4.4.3.2, “Dividindo uma Imagem Redundante de um Volume Lógico Espelhado”.

- Você agora pode criar um log de espelho para um dispositivo lógico espelhado o qual é espelhado por si só, usando o argumento **--mirrorlog mirrored** do comando **lvcreate** ao criar um dispositivo lógico espelhado. Para informações sobre como usar esta opção, veja a Seção 4.4.3, “Criando Volumes Espelhados”.

1.1.2. Novos e Modificados Recursos do Red Hat Enterprise Linux 6.1

O Red Hat Enterprise Linux 6.1 inclui a seguinte documentação e apresenta atualizações e mudanças.

- O lançamento do Red Hat Enterprise Linux 6.1 suporta a criação de snapshots de volumes lógicos a partir volumes lógicos espelhados. Você cria um snapshot de um volume espelhado assim como você criaria um snapshot de um volume lógico linear ou distribuído. Para informações sobre criar volumes snapshots, veja a Seção 4.4.4, “Criando Volumes Snapshots”.

- Quando extender um volume LVM, você pode agora usar a opção **--alloc cling** do comando **lvextend** para especificar a política de alocação. Esta política escolherá espaço nos mesmos volumes físicos assim como o último segmento do volume lógico existente. Se há espaço insuficiente nos volumes físicos e uma lista de rótulos é definida no arquivo **lvm.conf**, o LVM checará se quaisquer dos rótulos estão anexados aos volumes físicos e buscam coincidir esses rótulos de volume físico entre existentes extensões e novas extensões.

  Para informações sobre extender volumes LVM espelhados com a opção **--alloc cling** do comando **lvextend**, veja a Seção 4.4.12.2, “Extendendo um Volume Lógico com a Política de Alocação **cling**.”.

- Você pode agora especificar múltiplos argumentos **--addtag** e **--deltag** dentro de um único comando **pvchange**, **vgchange** ou **lvchange**. Para informações sobre adicionar e remover rótulos de objetos, veja Seção C.1, “Adicionando e Removendo Tags de Objetos”.

- A lista de caracteres permitidos em rótulos de objetos LVM foi extendida e os rótulos podem conter os caracteres "/", "=", "!", ":", "#", and "&". Para informações sobre rótulos de objetos LVM, veja o Apêndice C, *Tags de Objetos do LVM*.

- Você pode agora combinar RAID0 (distribuição) and RAID1 (espelho) em um único volume lógico. Criando um volume lógico enquanto simultaneamente especifica o número de espelhos (**--mirrors X**) e o número de resultados de distribuições (**--stripes Y**) em um dispositivo de espelho os quais dispositivos constituintes estão distribuídos. Para informação sobre criar volumes lógicos espelhados, veja a Seção 4.4.3, “Criando Volumes Espelhados”.

- Com o lançamento do Red Hat Enterprise Linux 6.1, se você precisar criar um backup consistente de dados em um volume lógico clusterizado, você pode ativar o volume exclusivamente e então criar um snapshot. Para informações sobre ativação de volumes lógicos exclusivamente em um nó, veja a Seção 4.7, “Ativando Volumes Lógicos em Nós Individuais em um Cluster”.

1.2. VOLUMES LÓGICOS.

O gerenciamento de volume cria uma camada de abstração sobre o armazenamento físico, permitindo que você crie volumes de armazenamento lógicos. Isto fornece maior flexibilidade em diversas formas ao invés de utilizar o armazenamento físico diretamente. Com um volume lógico, você não está restrito a tamanhos de discos físicos. Além disso, a configuração do armazenamento do hardware é escondida do software para que possa ter seu tamanho redefinido e movido sem parar os aplicativos ou desmontar os sistemas de arquivo. Isto pode reduzir custos operacionais.

Volumes lógicos fornecem as seguintes vantagens em relação a usar armazenamento físico diretamente:

- Capacidade Flexível

  Ao usar volumes lógicos, os sistemas de arquivo podem se estender por múltiplos discos, desde que você agregue discos e partições em um volume lógico único.

- Pools de armazenamento redimensionáveis

  Você pode estender volumes lógicos ou reduzí-los de tamanho com comandos de software simples, sem reformatar e reparticionar os dispositivos de discos subjacentes.

- Realocação de dados online

  Para implementar subsistemas de armazenamento mais rápidos, mais novos e resistentes, você pode mover dados enquanto seu sistema estiver ativo. Os dados podem ser reorganizados em discos enquanto os discos estiverem em uso. Por exemplo, você pode esvaziar um disco de troca rápida antes de removê-lo.

- Nomeação de Dispositivo conveniente

  Volumes de armazenamento lógico podem ser gerenciados em grupos definidos por usuários, os quais você poderá nomear como desejar.

- Distribuição de Disco

  Você pode criar um volume lógico que distribui os dados em dois ou mais discos. Isto pode aumentar drasticamente a taxa de transferência.

- Espelhando volumes.

  Os volumes lógicos fornecem uma forma conveniente de configurar um espelho para seus dados.

- Snapshots de Volumes

  Usando volumes lógicos, você poderá criar snapshots de dispositivos para backups consistentes ou testar o efeito de mudanças sem afetar os dados reais.

As implementações destes recursos no LVM estão descritas no restante deste documento.

1.3. VISÃO GERAL DA ARQUITETURA DO LVM

Para o lançamento do Red Hat Enterprise Linux 4 do sistema operacional do Linux, o gerenciador de volume lógico LVM1 foi substituído pelo LVM2, o qual possui uma estrutura de kernel mais genérica do que o LVM1. O LVM2 fornece as seguintes melhorias em relação ao LVM1:

- capacidade flexível
- armazenamento de metadados mais eficiente
- formato de recuperação melhor
- novo formato de metadados em ASCII
- mudanças atômicas nos metadados
- cópias redundantes de metadados

O LVM2 é compatível com o LVM1, com exceção do snapshot e suporte de cluster. Você pode converter um grupo de volume do formato LVM1 para o formato LVM2 com o comando **`vgconvert`**. Para mais informações sobre como converter o formato de metadados LVM, veja a página man (8) do **`vgconvert`**.

A unidade armazenamento físico subjacente de um volume lógico LVM é um dispositivo de bloco como uma partição ou um disco inteiro. Este dispositivo é inicializado como um LVM *physical volume* (PV).

Para criar um volume lógico LVM, os volumes lógicos são combinados em um grupo de volumes, *volume group* (VG). Isto cria um àrea de espaço no disco, no qual os volumes lógicos LVM (LVs) podem ser alocados. Este processo é análogo na forma que os discos são divididos em partições. Um volume lógico é usado pelo sistema de arquivo e aplicativos (tal como banco de dados).

A Figura 1.1, “Componentes de Volume Lógico LVM” mostra os componentes de um volume lógico LVM simples:

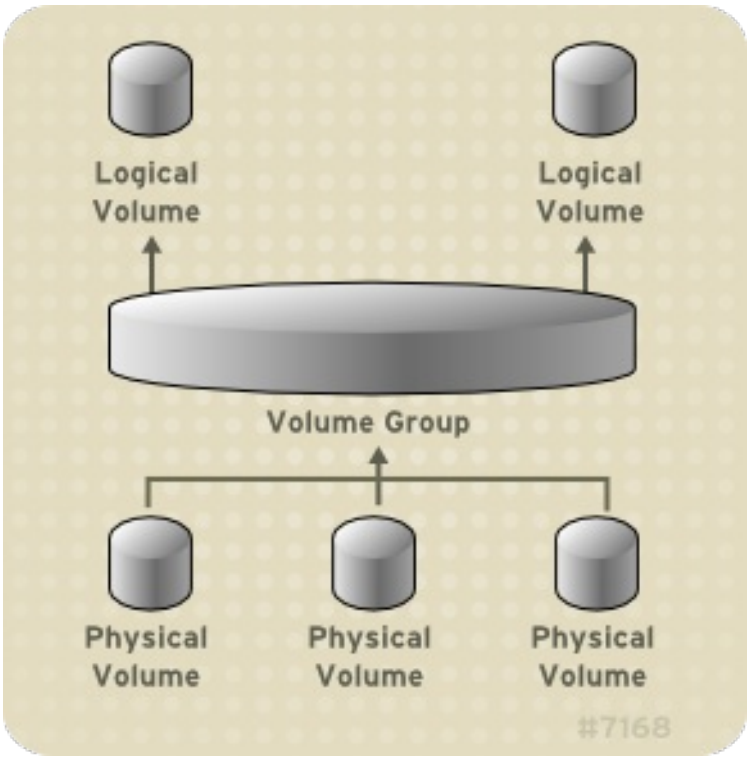


Figura 1.1. Componentes de Volume Lógico LVM

Para informações mais detalhadas sobre os componentes de um volume lógico LVM, veja o Capítulo 2, *Componentes do LVM*.

1.4. O GERENCIADOR DE VOLUME LÓGICO EM CLUSTER (CLVM)

O Clustered Logical Volume Manager (CLVM) é um conjunto de extensões de cluster para o LVM. Estas extensões permitem que um cluster de computadores gerencie o armazenamento compartilhado (por exemplo, em um SAN) usando o LVM. O CLVM é parte do Resilient Storage Add-On.

O uso do CLVM depende dos requerimentos do seu sistema:

- Se somente um nó de seu sistema requer acesso ao armazenamento que você estará configurando como volumes lógicos, você pode então utilizar o LVM sem as extensões do CLVM e os volumes lógicos criados com este nó serão todos locais ao nó.
- Se você estiver usando um sistema em cluster para failover, onde somente um único nó acessa o armazenamento e está ativo a qualquer momento, você deve usar os agentes do Gerenciamento de Volume Lógico de Disponibilidade (HA-LVM). Para informações sobre o HA-LVM, veja *Configurando e Gerenciando um Red Hat Cluster*.
- Se mais de um nó de seu cluster requer acesso ao seu armazenamento, o qual é compartilhado entre os nós ativos, você precisará usar o CLVM. O CLVM permite que um usuário configure os volumes lógicos em armazenamento compartilhado, bloqueando acesso ao armazenamento físico enquanto um volume lógico está sendo configurado, e usa os serviços de bloqueio em cluster para gerenciar o armazenamento compartilhado.

Para usar o CLVM, o software High Availability Add-On e Resilient Storage Add-On, incluindo o daemon do **clvmd**, devem estar em execução. O daemon **clvmd** é a extensão de cluster chave para o LVM. O daemon **clvmd** roda em cada computador de cluster e distribui as atualizações dos metadados do LVM em um cluster, apresentando cada computador de cluster com a mesma visão dos volumes lógicos. Para informações sobre como instalar e administrar o High Availability Add-On veja *Configurando e Gerenciando um Red Hat Cluster*.

Para se certificar que o **clvmd** foi iniciado durante a inicialização, você pode executar o comando **chkconfig ... on** no serviço **clvmd**, como em seguida:

```
# chkconfig clvmd on
```

Se o daemon **clvmd** não foi iniciado, você poderá executar um comando **service ... start** no serviço **clvmd**, como em seguida:

```
# service clvmd start
```

A criação de volumes lógicos LVM em um ambiente de cluster é idêntica à criação de volumes lógicos LVM em um nó único. Não há diferença nos comandos LVM ou na interface de usuário gráfica do LVM, como descrita no Capítulo 4, *Administração do LVM com os comandos de CLI* e no Capítulo 7, *Administração com o LVM GUI*. Para habilitar os volumes LVM que você está criando em um cluster, a infraestrutura precisa estar em execução e o cluster deve estar em quorum.

Por padrão, os volumes lógicos criados com o CLVM em armazenamento compartilhado são visíveis em todos os sistemas que possuem acesso ao armazenamento compartilhado. É possível criar grupos de volume no qual todos os dispositivos de armazenamento são visíveis à somente um nó no cluster. Também é possível modificar o status de um grupo de volume, de um grupo de volume local para um grupo de volume em cluster. Para mais informações, veja a Seção 4.3.2, “Criando Grupos de Volume em um Cluster” e a Seção 4.3.7, “Modificando os Parâmetros de um Grupo de Volume”.

ATENÇÃO

Quando você criar os grupos de volume com o CLVM em armazenamento compartilhado, você precisa se certificar que todos os nós no cluster possuem acesso aos volumes físicos que constituem o grupo de volume. As configurações de cluster assimétricas nas quais alguns nós possuem acesso ao armazenamento e outros não, não são suportados.

A Figura 1.2, “Visão Geral do CLVM” exibe uma visão geral do CLVM em um cluster.

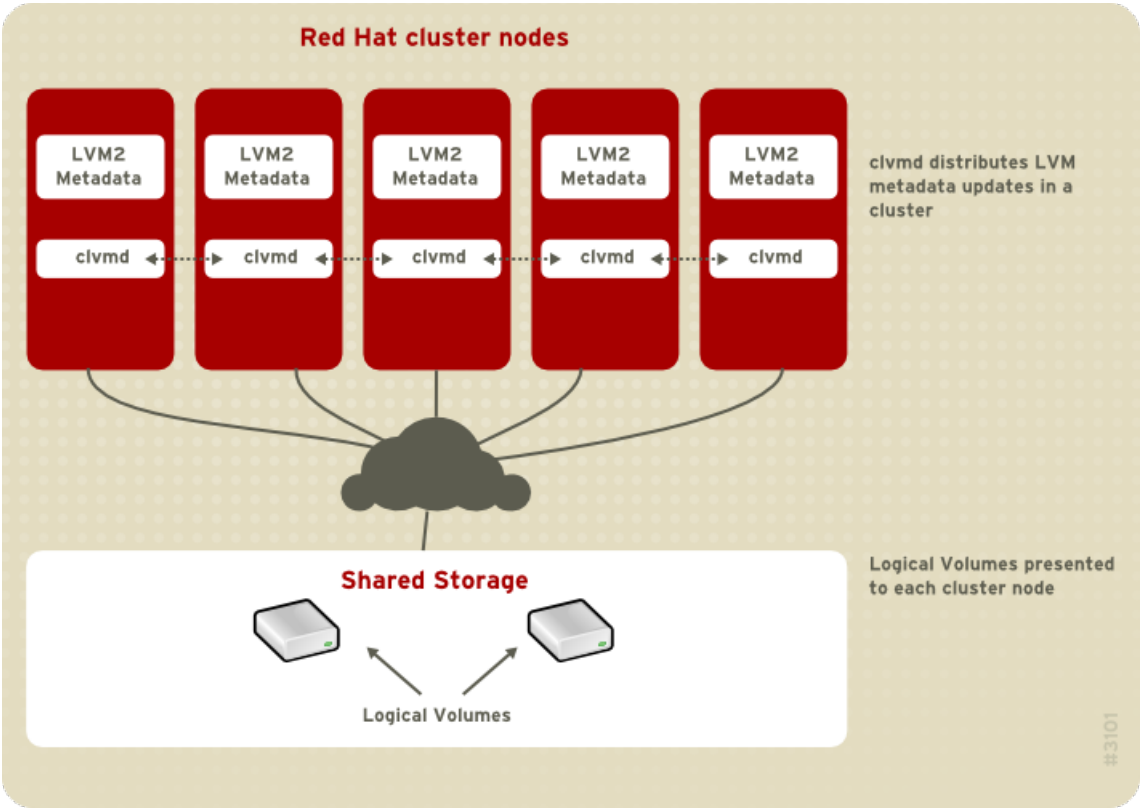
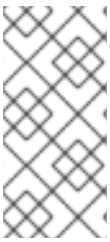


Figura 1.2. Visão Geral do CLVM



NOTA

O CLVM requer mudanças no arquivo **`lvm.conf`** para o bloqueio de todo o cluster. Informações sobre a configuração do arquivo **`lvm.conf`** para suportar um bloqueio de cluster, são fornecidas dentro do próprio arquivo **`lvm.conf`**. Para informações sobre o arquivo **`lvm.conf`**, veja o Apêndice B, *Os arquivos de Configuração do LVM*.

1.5. VISÃO GERAL DO DOCUMENTO

A continuação deste documento inclui os seguintes capítulos:

- Capítulo 2, *Componentes do LVM* descreve os componentes que criam um volume lógico LVM.

- Capítulo 3, *Visão Geral da Administração do LVM* fornece uma visão geral dos passos básicos que você realiza para configurar os volumes lógicos LVM, se você estiver usando o comando LVM Command Line Interface (CLI) ou o LVM Graphical User Interface (GUI).
- Capítulo 4, *Administração do LVM com os comandos de CLI* resume as tarefas administrativas individuais que você pode realizar com os comando LVM CLI para criar e manter os volumes lógicos.
- Capítulo 5, *Exemplos de Configuração do LVM* fornece uma variedade de exemplos de configuração do LVM.
- Capítulo 6, *Solução de Problemas LVM* fornece instruções sobre troubleshooting de diversos problemas do LVM.
- Capítulo 7, *Administração com o LVM GUI* resume o operacional do LVM GUI.
- Apêndice A, *O Mapeador de Dispositivo* descreve o Mapeador de Dispositivo que o LVM usa para mapear volumes lógicos e físicos.
- Apêndice B, *Os arquivos de Configuração do LVM* descreve os arquivos de configuração do LVM.
- Apêndice C, *Tags de Objetos do LVM* descreve as marcações de objeto do LVM e marcações de host.
- Apêndice D, *Metadados de Grupo de Volume LVM* descreve os metadados de grupo de volume do LVM, e inclui uma cópia de amostra do metadado para um grupo de volume LVM.

CAPÍTULO 2. COMPONENTES DO LVM

Este capítulo descreve os componentes de um volume Lógico LVM.

2.1. VOLUMES FÍSICOS

A unidade do armazenamento físico subjacente de um volume lógico LVM é um dispositivo de bloco tal como uma partição ou um disco inteiro. Para usar o dispositivo para um volume lógico LVM, o dispositivo deve ser inicializado como um volume físico (PV). A inicialização de um dispositivo de bloco como um volume físico coloca um rótulo perto do início do dispositivo.

Por padrão, o rótulo do LVM é colocado no segundo setor de 512 bytes. Você pode sobrescrever este padrão colocando o rótulo em qualquer um dos primeiros 4 setores. Isto permite que os volumes LVM co-existam com outros usuários destes setores, caso seja necessário.

Um rótulo do LVM fornece identificação correta e ordenação de dispositivo para um dispositivo físico, pois os dispositivos podem surgir em qualquer ordem quando o sistema é inicializado. Um rótulo do LVM fica persistente nas reinicializações e em todo o cluster.

O rótulo do LVM identifica o dispositivo como um volume físico do LVM. Ele contém um identificador único aleatório (o UUID) para o volume físico. Ele também armazena o tamanho do dispositivo de bloco em bytes e grava onde os metadados do LVM serão armazenados no dispositivo.

O metadado do LVM contém os detalhes de configuração dos grupos de volume do LVM em seu sistema. Por Padrão, uma cópia idêntica do metadado é mantida em todas as áreas de metadados em todos os volumes físicos dentro do grupo de volume. O metadado do LVM é pequeno e armazenado como ASCII.

Atualmente o LVM permite que você armazene 0, 1 ou 2 cópias idênticas de seus metadados em cada volume físico. O padrão é 1 cópia. Depois que você configurar o número de cópias de metadados no volume físico, você não pode mudar o número mais tarde. A primeira cópia é armazenada no início do dispositivo, logo depois do rótulo. Se não houver uma segunda cópia, ele é colocado no final do dispositivo. Se você sobrescrever acidentalmente a área de início do seu disco, gravando em um disco diferente do que você pretendia, uma segunda cópia dos metadados no final do dispositivo permitirá que você recupere os metadados.

Para informações detalhadas sobre os metadados do LVM e mudança de parâmetros de metadados, veja o Apêndice D, *Metadados de Grupo de Volume LVM*.

2.1.1. Layout do Volume Físico do LVM

A Figura 2.1, “Layout de Volume Físico” exibe o layout de um volume físico LVM. O rótulo do LVM está no segundo setor, seguido da area de metadados e, seguido de espaço útil no dispositivo.

NOTA

No kernel do Linux (e em toda esta documentação), os setores devem ter 512 bytes de tamanho.

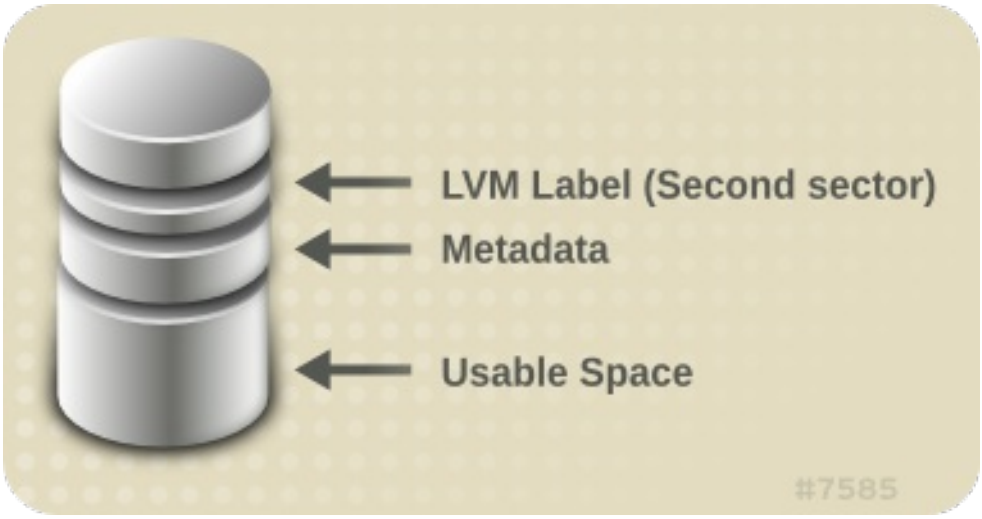


Figura 2.1. Layout de Volume Físico

2.1.2. Partições Múltiplas em um Disco

O LVM permite que você crie volumes físicos fora das partições dos discos. Recomendamos que você crie uma partição única que cubra todo o disco para rotular como um volume físico do LVM, devido as seguintes razões:

- Conveniência Administrativa

  É mais fácil saber onde está o hardware em um sistema se cada disco real somente aparecer uma vez. Principalmente se um disco falhar. Além disso, volumes físicos em um disco único podem criar um aviso de kernel sobre tipos de partição desconhecida durante a inicialização.

- Desempenho de Distribuição

  O LVM não pode informar se dois volumes físicos estão no mesmo disco físico. Se você criar um volume lógico distribuído quando dois volumes físicos estiverem no mesmo disco físico, a distribuição pode acontecer em partições diferentes no mesmo disco. Isto resultaria em uma diminuição de desempenho ao invés de aumento.

Embora não seja recomendado, podem haver algumas circunstâncias nas quais você precisará dividir um disco em volumes físicos LVM separados. Por exemplo, em um sistema com poucos discos, pode ser necessário mover os dados entre as partições quando migrar um sistema existente para volumes LVM. Além disso, se você possuir um disco grande e desejar possuir mais do que um grupo de volume para propósitos administrativos, será necessário particionar o disco. Se você possuir um disco com mais do que uma partição e ambas as partições estiverem no mesmo grupo de volume, cuidado ao especificar quais partições devem ser incluídas no volume lógico ao criar volumes distribuídos.

2.2. GRUPOS DE VOLUME

Volumes físicos são combinados em grupos de volumes (VG). Isto cria um pool de espaço de disco onde os volumes podem ser alocados.

Dentro de um grupo de volume, o espaço de disco disponível para a alocação é dividido em unidades de um tamanho fixo chamadas extensões. Uma extensão é a unidade menor do espaço que pode ser alocada. Dentro de um volume físico, as extensões são referidas como extensões físicas.

Um volume lógico é alocado em extensões lógicas do mesmo tamanho como extensões físicas. O tamanho da extensão é portanto o mesmo para todos os volumes lógicos no grupo de volume. O grupo de volume mapeia as extensões lógicas para extensões físicas.

2.3. VOLUMES LÓGICOS LVM

No LVM, um grupo de volume é dividido em volumes lógicos. Existem três tipos de volumes lógicos no LVM: volumes *lineares*, volumes *distribuídos* e volumes *espelhados*. Estando descritos nas seguintes seções:

2.3.1. Volumes Lineares

Um volume linear agrega múltiplos volumes físicos em um volume lógico. Por exemplo, se você possui dois discos de 60GB, poderá criar um volume lógico de 120GB. O armazenamento físico é concatenado.

A criação de um volume linear atribui uma variedade de extensões para uma área de um volume lógico em ordem. Por exemplo, como demonstrado na Figura 2.2, “Mapeamento de Extensão” as extensões lógicas 1 até 99 podem mapear em um volume físico e extensões lógicas de 100 à 198 podem mapear em um segundo volume físico. Do ponto de vista do aplicativo, existe um dispositivo com 198 extensões de tamanho.

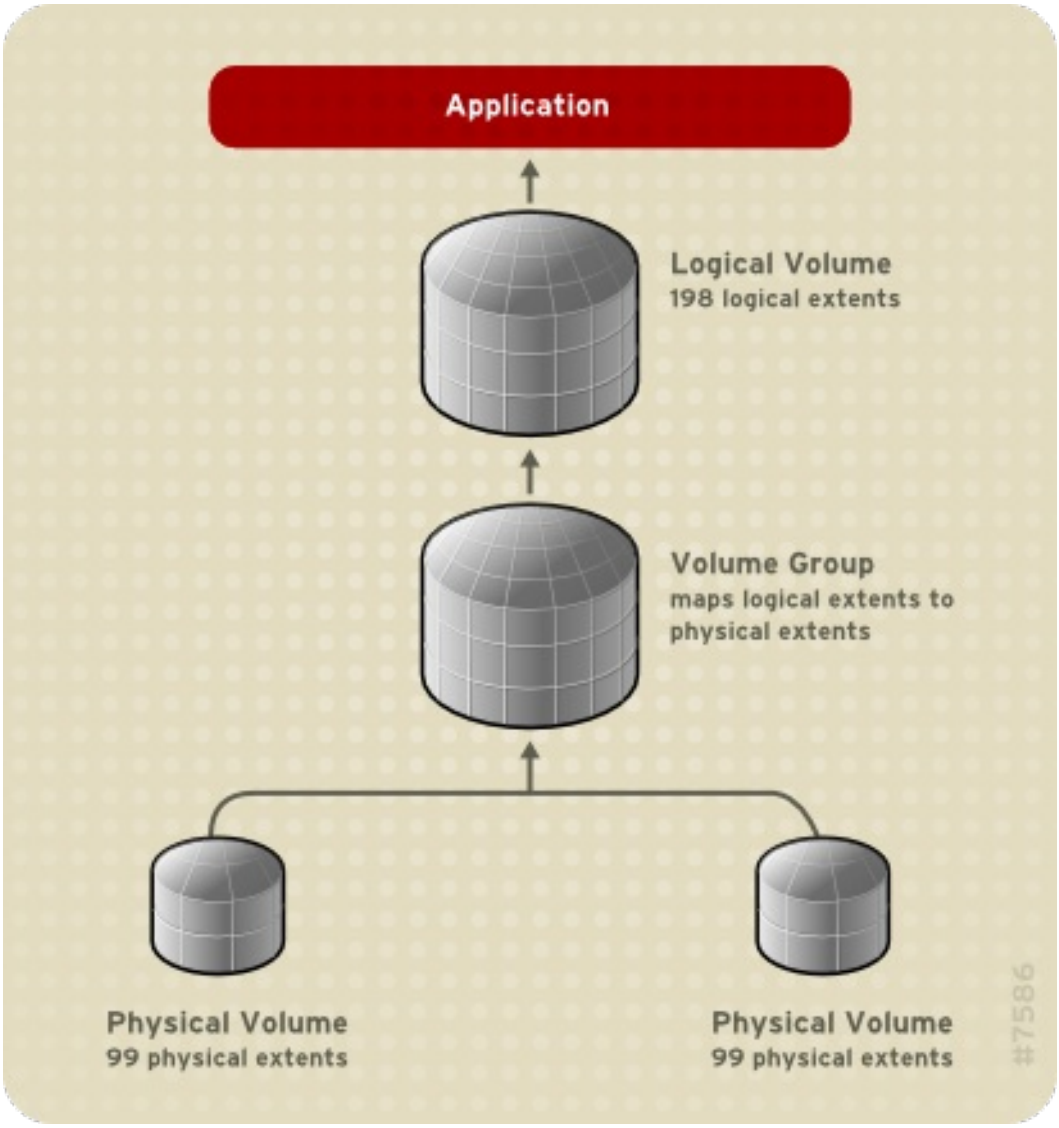


Figura 2.2. Mapeamento de Extensão

Os volumes físicos que criam um volume lógico, não precisam ter o mesmo tamanho. A Figura 2.3, “O Volume Linear com Volumes Físicos Desiguais” mostra o grupo de volume **VG1** com uma extensão física de 4MB. Este grupo de volume inclui 2 volumes físicos chamados **PV1** e **PV2**. Os volumes físicos são divididos em unidades de 4MB, pois este é o tamanho da extensão. Neste exemplo, o **PV1** tem extensão de 200 (800MB) e **PV2** possui 100 extensões em tamanho (400MB). Você pode criar um volume linear de qualquer tamanho entre 1 e 300 extensões (4MB até 1200MB). Neste exemplo, o volume linear chamado **LV1** possui 300 extensões de tamanho.

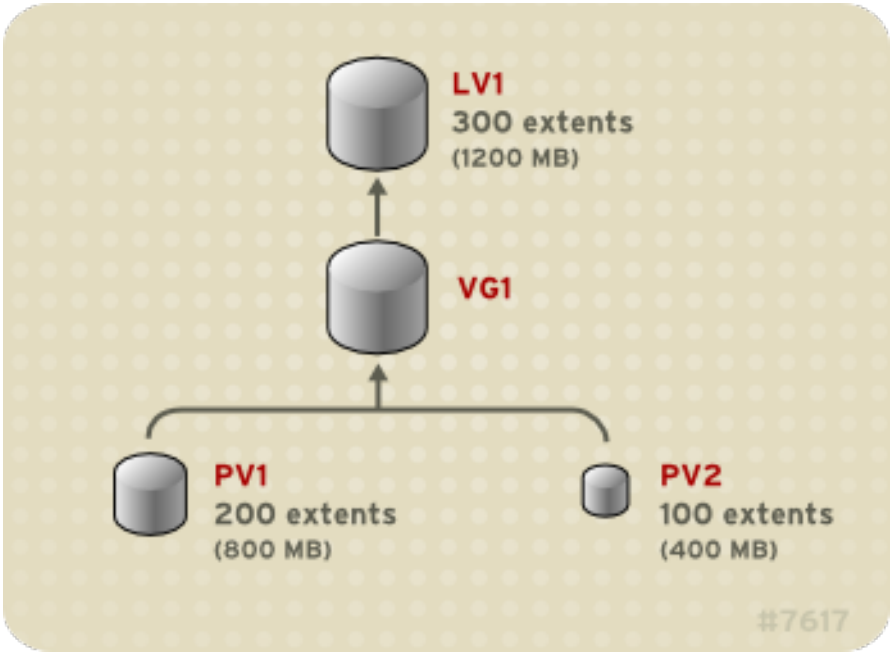


Figura 2.3. O Volume Linear com Volumes Físicos Desiguais

Você pode configurar mais do que um volume lógico linear de qualquer tamanho que desejar a partir de um grupo de extensões físicas. A Figura 2.4, “Volumes Lógicos Múltiplos” demonstra o mesmo grupo de volume da Figura 2.3, “O Volume Linear com Volumes Físicos Desiguais”, mas neste caso, dois volumes lógicos foram retirados do grupo de volume: **LV1**,o qual possui 250 extensões de tamanho (1000MB) e **LV2** com 50 extensões de tamanho (200MB).

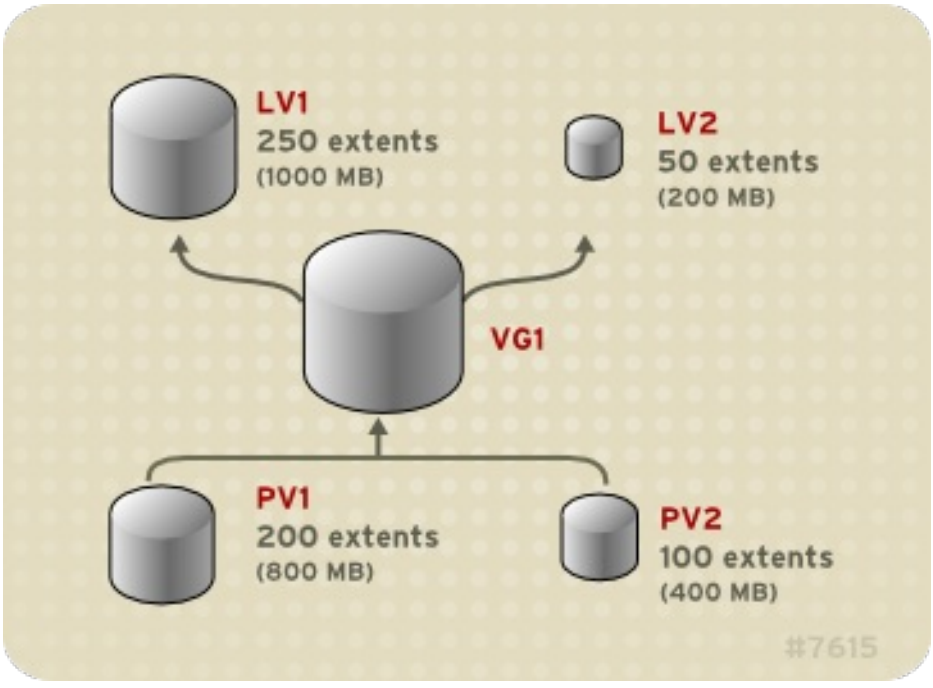


Figura 2.4. Volumes Lógicos Múltiplos

2.3.2. Volumes Lógicos Distribuídos

Quando você grava dados em um volume lógico LVM, o sistema de arquivos deixa os dados nos volumes físicos subjacentes. Você pode controlar a forma que os dados são gravados nos volumes físicos, criando um volume lógico distribuído. Para leituras e gravações sequenciais grandes, isto pode melhorar a eficiência dos dados E/S.

A distribuição melhora o desempenho gravando dados em um número de volumes físicos pré-determinados de maneira rotativa. Com a distribuição, a E/S pode ser realizada em paralelo. Em algumas situações, isto pode resultar em um ganho de desempenho quase linear para cada volume físico adicional na distribuição.

A ilustração a seguir mostra dados sendo distribuídos em três volumes físicos. Nesta figura:

- a primeira distribuição de dados é gravada em PV1.
- a segunda distribuição de dados é gravada em PV2.
- a terceira distribuição de dados é gravada em PV3.
- a quarta distribuição de dados é gravada em PV4.

Em um volume lógico distribuído, o tamanho da distribuição não pode exceder o tamanho de uma extensão.

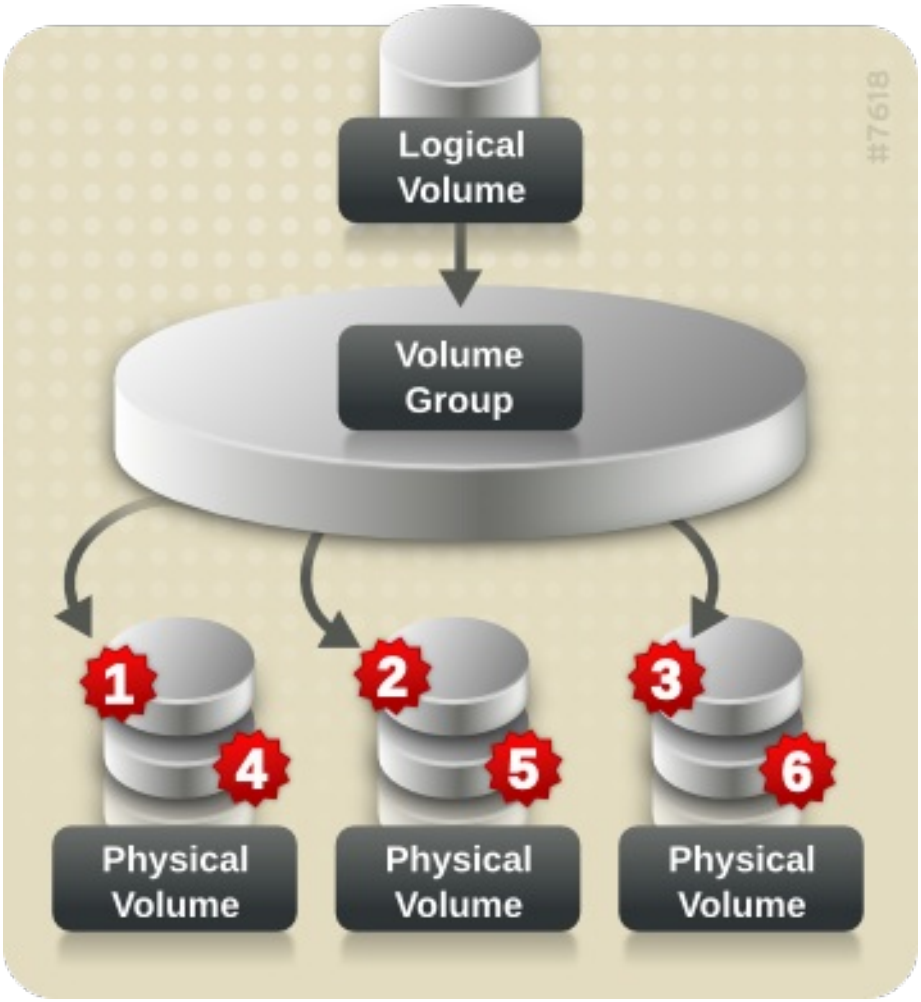


Figura 2.5. Distribuindo Dados em Três Volumes Físicos

Os volumes lógicos distribuídos podem ser estendidos concatenando outro conjunto de dispositivos no

final do primeiro conjunto. Para estender um volume lógico distribuído, no entanto, deve haver espaço livre suficiente nos volumes físicos subjacentes que criam o grupo de volume para suportar a distribuição. Por exemplo, se você possui uma distribuição de duas vias, que usa até um grupo de volume inteiro, se você adicionar um volume físico único ao grupo de volume, não conseguirá estender a distribuição. Ao invés disso, você precisa adicionar ao menos dois volumes físicos ao grupo de volume. Para mais informações sobre extensão de um volume distribuído, veja a Seção 4.4.12.1, “Extendendo um Volume Distribuído”.

2.3.3. Volumes Lógicos Espelhados

Um espelho mantém cópias idênticas de dados em dispositivos diferentes. Quando os dados são gravados no dispositivo, ele é gravado em um segundo dispositivo também, espelhando os dados. Isto fornece proteção para as falhas de dispositivos. Quando uma perna de um espelho falhar, o volume lógico se tornará um volume linear e poderá ainda ser acessado.

O LVM suporta volumes espelhados. Quando você criar um volume lógico espelhado, o LVM certificará de que os dados gravados em um volume físico subjacente seja espelhado em um volume fisico separado. Com o LVM, você pode criar volumes lógicos espelhados com espelhos múltiplos.

Um espelho LVM divide o dispositivo sendo copiado em regiões que são tipicamente 512KB em tamanho. O LVM mantém um pequeno registro que usa para manter registro de quais regiões estão em sincronia com o espelho ou espelhos. Este registro de log pode ser mantido no disco, que o manterá persistente durante reinicializações ou pode ser mantido na memória.

A Figura 2.6, “Volume lógico espelhado” exibe um volume lógico espelhado com um espelho. Nesta configuração o log é mantido no disco.

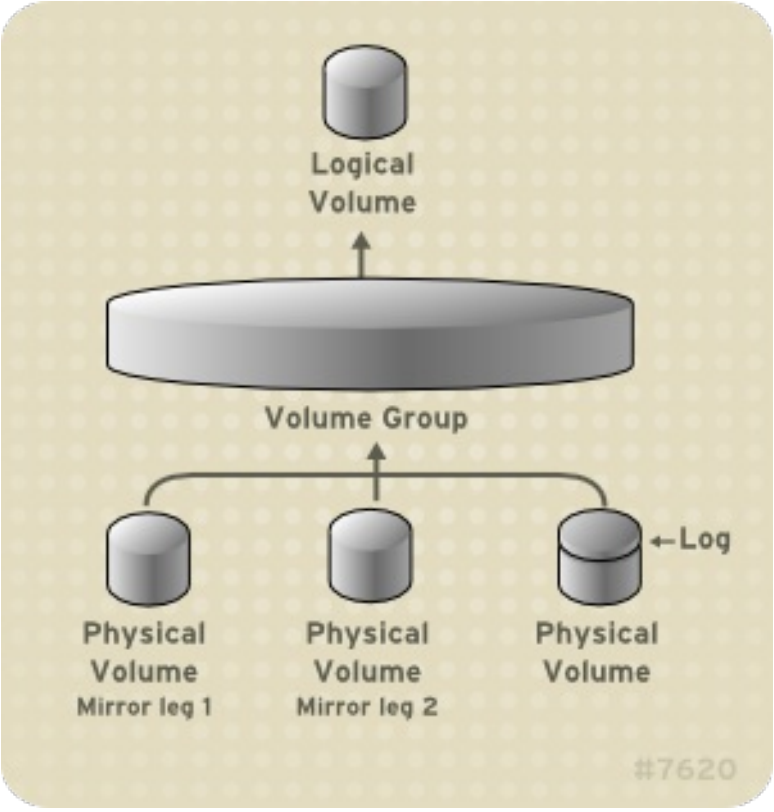
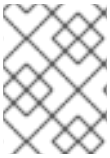


Figura 2.6. Volume lógico espelhado

Para informações sobre como criar e modificar espelhos, veja a Seção 4.4.3, “Criando Volumes Espelhados”.

2.3.4. Volumes de Snapshot

O recurso de Snapshot do LVM fornece a habilidade de criar imagens virtuais de um dispositivo em um momento específico sem causar interrupção de um serviço. Quando uma mudança é realizada ao dispositivo original (a origem) após um snapshot ter sido feito, o recurso do snapshot faz uma cópia do dado modificado como se fosse antes da mudança, assim ele pode reconstruir o estado do dispositivo.



NOTA

Os snapshots do LVM não são suportados em todos os nós em um cluster. Você não pode criar um volume de snapshot em um grupo de volume clusterizado.

NOTA

Os snapshots de LVM não são suportados para volumes lógicos espelhados de LVM.

Como as cópias do snapshot copiam somente áreas de dados que mudam após o snapshot ter sido criado, o recurso snapshot requer uma quantidade mínima de armazenamento. Por exemplo, com uma origem atualizada raramente, 3-5% da capacidade da origem é suficiente para manter o snapshot.

NOTA

As cópias de snapshot de um sistema de arquivo são cópias virtuais e não backup de mídia atual para um sistema de arquivo. Os snapshots não são um substituto para um procedimento de backup.

O tamanho dos snapshots regem a quantidade de espaço definida à parte para armazenar mudanças no volume de origem. Por exemplo, se você produziu um snapshot e sobrescreveu o original completamente, ele deveria ser do mesmo tamanho que o volume original para manter as mudanças. Você precisa dimensionar um snapshot de acordo com o nível esperado de mudança. Portanto, por exemplo, um snapshot de curta duração de um volume quase todo só de leitura, tal como **/usr**, precisaria de menor espaço do que um snapshot de longa duração de um volume que tem um número maior de gravações, tal como **/home**.

Se um snapshot se tornar cheio, ele se torna inválido pois não rastreará mais as mudanças no volume original. Você deve monitorar regularmente o tamanho do snapshot. Os snapshots podem ter seus tamanhos totalmente redefinidos, assim se tiver a capacidade de armazenamento, poderá aumentar o tamanho de um volume snapshot para evitar uma perda. Se você achar que o volume do snapshot é maior do que o que você precisa, você pode reduzir o tamanho de um volume para liberar espaço necessário para outros volumes lógicos.

Quando você cria um sistema de arquivo snapshot, total leitura e escrita à origem continua possível. Se uma parte do snapshot é modificada, esta parte é marcada e nunca será copiada do volume original.

Existem diversas utilidades para o recurso snapshot:

- A mais comum, um snapshot tirado quando você precisa realizar um backup em um volume lógico sem interromper o sistema ativo que está atualizando continuamente os dados.
- Você pode executar o comando **fsck** em um sistema de arquivo snapshot para verificar a integridade do sistema de arquivo e determinar se o sistema de arquivo original requer reparos.
- Como o snapshot é de leitura/gravação, você pode testar aplicativos em dados de produção, tirando o snapshot e executando testes no snapshot, deixando os dados reais intactos.

- Você pode criar volumes LVM para uso com a Virtualização LVM. Os snapshots LVM podem ser usados para criar snapshots de imagens de visitantes virtuais. Estes snapshots podem fornecer uma maneira conveniente para modificar visitantes existentes ou criar novos visitantes com uma armazenagem adicional mínima. Para mais informações sobre criar snapshots LVM de visitantes virtualizados, veja o *Guia de Virtualização Red Hat Enterprise Linux*.

Para informações sobre como criar volumes de snapshot, veja a Seção 4.4.4, “Criando Volumes Snapshots”.

Desde o lançamento do Red Hat Enterprise Linux 6, é possível utilizar a opção **`--merge`** do comando **`lvconvert`** para mesclar um snapshot em seu volume original. Uma das utilidades deste recurso é realizar um rollback de sistema se você perdeu dados ou arquivos, ou então precisar recuperar seu sistema em um estado anterior. Após mesclar o volume de snapshot, o volume lógico resultante terá o nome do volume de origem, número menor e UUID e o snapshot incorporado é removido. Para informações sobre como usar esta opção, veja a Seção 4.4.5, “Incorporando Volumes Snapshots”.

CAPÍTULO 3. VISÃO GERAL DA ADMINISTRAÇÃO DO LVM

Este capítulo fornece uma visão geral de procedimentos administrativos usados para configurar os volumes lógicos do LVM. Este capítulo tem o objetivo de proporcionar uma visão geral dos passos envolvidos. Para exemplos de passo-a-passo dos procedimentos de configuração do LVM comuns, veja o Capítulo 5, *Exemplos de Configuração do LVM*.

Para descrições de comandos de CLI que você pode usar para realizar a administração do LVM, veja o Capítulo 4, *Administração do LVM com os comandos de CLI*. Como forma alternativa, você pode utilizar o LVM GUI, o qual está descrito no Capítulo 7, *Administração com o LVM GUI*.

3.1. CRIANDO VOLUMES LVM EM UM CLUSTER

Para criar volumes lógicos em um ambiente de cluster, você pode utilizar o Gerenciador de Volume Lógico Clusterizado (CLVM), o qual é um conjunto de extensões de cluster para o LVM. Estas extensões permitem que um cluster de computadores gerencie o armazenamento compartilhado (por exemplo, em um SAN) usando o LVM. Para utilizar o CLVM, o software High Availability Add-On e Resilient Storage Add-On, incluindo o daemon de **`clvmd`**, devem ser rodados durante a inicialização, como descrito na Seção 1.4, “O Gerenciador de Volume Lógico em Cluster (CLVM)”.

A criação de volumes lógicos do LVM em um ambiente de cluster é idêntica à criação dos volumes lógicos em um nó único. Não há diferenças nos próprios comandos do LVM ou na interface do LVM GUI. Para habilitar os volumes do LVM que você está criando em um cluster, a infraestrutura do cluster deve estar rodando e o cluster deve ser quorum.

O CLVM requer mudanças no arquivo **`lvm.conf`** para o bloqueio de todo o cluster. Informações sobre a configuração do arquivo **`lvm.conf`** para suportar um bloqueio de cluster, são fornecidas dentro do próprio arquivo **`lvm.conf`**. Para informações sobre o arquivo **`lvm.conf`**, veja o Apêndice B, *Os arquivos de Configuração do LVM*.

Por padrão, os volumes lógicos criados com o CLVM em armazenamento compartilhado são visíveis em todos os sistemas que possuem acesso ao armazenamento compartilhado. É possível criar grupos de volumes nos quais todos os dispositivos de armazenamento são visíveis à somente um nó no cluster. É possível também mudar o estado de um grupo de volumes pertencente a um grupo de volumes local para um grupo de volumes clusterizado. Para informações, veja a Seção 4.3.2, “Criando Grupos de Volume em um Cluster” e a Seção 4.3.7, “Modificando os Parâmetros de um Grupo de Volume”.

ATENÇÃO

Quando você criar os grupos de volume com o CLVM em armazenamento compartilhado, você precisa se certificar que todos os nós no cluster possuem acesso aos volumes físicos que constituem o grupo de volume. As configurações de cluster assimétricas nas quais alguns nós possuem acesso ao armazenamento e outros não, não são suportados.

Para informações sobre como instalar o High Availability Add-On e configurar uma infraestrutura de cluster, veja *Administração do Cluster*.

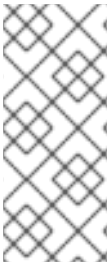
Para um exemplo de como criar um volume lógico espelhado em um cluster, veja a Seção 5.5, “Criando um Volume Lógico LVM Espelhado em um Cluster”.

3.2. VISÃO GERAL DA CRIAÇÃO DE VOLUMES LÓGICOS.

Segue um resumo dos passos a serem feitos para criar um volume lógico do LVM.

1. Inicialize as partições que você irá utilizar para o volume LVM como volumes físicos (isso os rotulará).
2. Crie um grupo de volume.
3. Crie um volume lógico.

Após criar o volume lógico, você pode criar e montar o sistema de arquivo . Os exemplos neste documento usam os sistemas de arquivo GFS2.



NOTA

Embora o sistema de arquivo GFS2 possa ser implementado em um sistema autônomo ou como parte de uma configuração de cluster, para a versão Red Hat Enterprise 6, a Red Hat não suporta o uso do GFS2 como um sistema de arquivo de nó único. A Red Hat irá continuar a suportar sistemas de arquivo GFS2 de nó único para montar snapshots de sistemas de arquivos de cluster (por exemplo, para propósito de backup).

1. Criar um sistema de arquivo GFS2 em um volume lógico com o comando **`mkfs.gfs2`**.
2. Criar um novo ponto de montagem com o comando **`mkdir`**. Em um sistema clusterizado, crie o ponto de montagem em todos os nós em um cluster.
3. Montar o sistema de arquivo. Você pode querer adicionar uma linha ao arquivo **`fstab`** para cada nó no sistema.

Como forma alternativa, você pode criar e montar o sistema de arquivo do GFS2 com o LVM GUI.

A criação de volume LVM é independente de máquina, desde que a área de armazenamento para as informações sobre a configuração do LVM estiver nos volumes físicos e não na máquina onde o volume foi criado. Os servidores que utilizam o armazenamento possuem cópias locais, mas podem recriá-las a partir do que estiver nos volumes físicos. Você pode anexar os volumes físicos em um servidor diferente se as versões do LVM forem compatíveis.

3.3. AUMENTANDO UM SISTEMA DE ARQUIVO EM UM VOLUME LÓGICO

Para aumentar um sistema de arquivo em um volume lógico, realize os seguintes passos:

1. Crie um novo volume fïsico.
2. Estenda o grupo de volume que contém o volume lógico com o sistema de arquivo que você está aumentando para incluir o novo volume físico.
3. Estenda o volume lógico para incluir o novo volume físico.
4. Aumente o sistema de arquivo.

Se você tiver suficiente espaço não alocado no grupo de volume, você pode utilizar este espaço para estender o volume lógico ao invés de realizar os passos 1 e 2.

3.4. BACKUP DE VOLUME LÓGICO

Os backups de metadados e arquivos são criados automaticamente em todos os grupos de volume e a configuração de volume lógico se altera a menos que esteja desabilitada no arquivo **lvm.conf**. Por padrão, o backup de metadados está armazenado no arquivo **/etc/lvm/backup** e os arquivos de metadados estão armazenados no arquivo **/etc/lvm/archive**. O tempo que os arquivos de metadados armazenados no arquivo **/etc/lvm/archive** são mantidos e a quantidade de arquivos mantida, são determinados pelos parâmetros que você pode estabelecer no arquivo **lvm.conf**. Um sistema diário de backup deve incluir o conteúdo do diretório **/etc/lvm** no backup.

Note que um backup de metadados não faz o backup do usuário e os dados de sistema contidos nos volumes lógicos.

Você pode fazer o backup do metadado manualmente no arquivo **/etc/lvm/backup** com o comando **vgcfgbackup** . Você pode restaurar os metadados com o comando **vgcfgrestore**. Os comandos **vgcfgbackup** e **vgcfgrestore** estão descritos na Seção 4.3.12, “Fazendo Cópias de Segurança de Metadados de Grupo de Volume”.

3.5. REGISTRANDO (LOGGING)

Todos os resultados de mensagens passam pelo módulo logging com opções independentes de níveis de registro para:

- erro / resultado padrão
- syslog
- arquivo log
- função de log externa

Níveis de logging são estabelecidos no arquivo **/etc/lvm/lvm.conf** descrito no Apêndice B, *Os arquivos de Configuração do LVM*.

CAPÍTULO 4. ADMINISTRAÇÃO DO LVM COM OS COMANDOS DE CLI

Este capítulo resume as tarefas administrativas individuais que você pode realizar com a interface de linha de comando (CLI) do LVM para criar e manter volumes lógicos.

NOTA

Se você estiver criando ou modificando um volume LVM para um ambiente clusterizado, você precisa certificar-se de que está executando o daemon do **clvmd**. Para informações, veja a Seção 3.1, “Criando Volumes LVM em um Cluster”.

4.1. USANDO OS COMANDOS DO CLI

Existem vários recursos dos comandos CLI LVM.

Quando os tamanhos são necessários em um argumento de linha de comando, as unidades podem sempre serem especificadas explicitamente. Se você não quiser especificar uma unidade, então o padrão assumirá, geralmente KB ou MB. Os comandos LVM CLI não aceitam frações.

Quando especificar as unidades em argumento na linha de comando, o LVM não diferencia maiúsculo de minúsculo, especificando o M ou m é equivalente por exemplo e as potências de 2 (múltiplos de 1024) são usadas. No entanto, quando você especifica o argumento **--units** em um comando, a letra minúscula indica que as unidades estão em múltiplos de 1024 enquanto a letra maiúscula indica que as unidades estão em múltiplos de 1000.

Quando os comandos levam um nome de grupo de volume ou nomes de volume lógicos como argumentos, o nome inteiro do caminho é opcional. Um volume lógico chamado **lvol0** dentro de um grupo de volume chamado **vg0** pode ser especificado como **vg0/lvol0**. Onde uma lista de grupo de volumes é requerida mas é deixada vazia, uma lista de todos os grupos de volumes é substituida. Onde uma lista de volumes lógicos é requerida mas o grupo de volume é dado, uma lista com todos os volumes lógicos daquele grupo de volume será substituida. Por exemplo, o comando **lvdisplay vg0** mostrará todos os volumes lógicos dentro do grupo de volumes **vg0**.

Todos os comandos LVM aceitam o argumento **-v**, o qual pode ser inserido multiplas vezes para aumentar o as informações do resultado. Os exemplos seguintes mostram os resultados padrões do comando **lvcreate**.

```
# lvcreate -L 50MB new_vg
  Rounding up size to full physical extent 52.00 MB
  Logical volume "lvol0" created
```

O comando a seguir mostra o resultado do comando **lvcreate** com o argumento **-v**.

```
# lvcreate -v -L 50MB new_vg
    Finding volume group "new_vg"
  Rounding up size to full physical extent 52.00 MB
    Archiving volume group "new_vg" metadata (seqno 4).
    Creating logical volume lvol0
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
    Found volume group "new_vg"
    Creating new_vg-lvol0
    Loading new_vg-lvol0 table
```

```
    Resuming new_vg-lvol0 (253:2)
    Clearing start of logical volume "lvol0"
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
  Logical volume "lvol0" created
```

Você pode também ter usado os argumentos **-vv**, **-vvv** ou **-vvvv** para mostrar cada vez mais detalhes sobre a execução do comando. O argumento **-vvvv** fornece a máxima quantidade de informação neste momento. O exemplo seguinte mostra somente as primeiras linhas de resultados para o comando **lvcreate** com o argumento **-vvvv** especificado.

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913         Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:916         O_DIRECT will be used
#config/config.c:864       Setting global/locking_type to 1
#locking/locking.c:138       File-based locking selected.
#config/config.c:841       Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358       Getting target version for linear
#ioctl/libdm-iface.c:1569        dm version   OF   [16384]
#ioctl/libdm-iface.c:1569        dm versions   OF   [16384]
#activate/activate.c:358       Getting target version for striped
#ioctl/libdm-iface.c:1569        dm versions   OF   [16384]
#config/config.c:864       Setting activation/mirror_region_size to 512
...
```

Você pode exibir ajuda para qualquer um dos comandos LVM CLI com o argumento **--help** no comando.

```
commandname --help
```

Para exibir a página man para um comando, execute o comando **man**:

```
man commandname
```

O comando **man lvm** fornece informações gerais online sobre o LVM.

Todos os objetos LVM são referenciados internamente por um UUID, o qual é designado quando você cria o objeto. Isto pode ser útil numa situação onde você remove um volume físico chamado **/dev/sdf** o qual é parte de um grupo de volumes e, quando você o ligar de volta, você verá que ele é agora **/dev/sdk**. O LVM ainda encontrará o volume físico porque ele identifica o volume físico pelo seu UUID and não pelo nome do dispositivo. Para informação sobre como especificar o UUID de um volume físico quando estiver criando um volume físico, veja a Seção 6.4, “Recuperando Metadados de Volume Físico”.

4.2. ADMINISTRAÇÃO DE VOLUME FÍSICO

Esta seção descreve os comandos que realizam os diversos aspectos da administração de volume físico.

4.2.1. Criando Volumes Físicos

As subseções a seguir descrevem os comandos usados para criar volumes físicos.

4.2.1.1. Configurando o Tipo de Partição

Se você estiver usando um dispositivo de disco inteiro para seu volume físico, o disco não deve conter nenhuma tabela de partição. Para as partições do disco DOS, a id da partição deve ser configurada para 0x8e usando o comando **fdisk** ou **cfdisk** ou ainda um equivalente a estes. Para os dispositivos de disco inteiro somente a tabela de partição deve ser apagada, que irá destruir efetivamente todos os dados naquele disco. Você pode remover uma tabela de partição existente, zerando o primeiro setor com o seguinte comando:

```
dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

4.2.1.2. Inicializando os Volumes Físicos

Use o comando **pvcreate** para inicializar um dispositivo de bloco a ser usado como um volume físico. Inicialização é análoga para formatar um sistema de arquivo.

O comando a seguir inicializa **/dev/sdd1**, **/dev/sde1** e **/dev/sdf1** para usar como volumes físicos LVM.

```
pvcreate /dev/sdd1 /dev/sde1 /dev/sdf1
```

Para inicializar partições ao invés de discos inteiros: execute o comando **pvcreate** na partição. O exemplo a seguir inicializa a partição **/dev/hdb1** como um volume físico LVM para uso mais tarde como parte de um volume lógico LVM.

```
pvcreate /dev/hdb1
```

4.2.1.3. Escaneando por Dispositivos de Bloco

Você pode escanear por dispositivos de bloco que podem ser usados como volumes físicos com o comando **lvmdiskscan**, como demonstrado no exemplo a seguir.

```
# lvmdiskscan
  /dev/ram0                 [       16.00 MB]
  /dev/sda                  [       17.15 GB]
  /dev/root                 [       13.69 GB]
  /dev/ram                  [       16.00 MB]
  /dev/sda1                 [       17.14 GB] LVM physical volume
  /dev/VolGroup00/LogVol01  [      512.00 MB]
  /dev/ram2                 [       16.00 MB]
  /dev/new_vg/lvol0         [       52.00 MB]
  /dev/ram3                 [       16.00 MB]
  /dev/pkl_new_vg/sparkie_lv [        7.14 GB]
  /dev/ram4                 [       16.00 MB]
  /dev/ram5                 [       16.00 MB]
  /dev/ram6                 [       16.00 MB]
  /dev/ram7                 [       16.00 MB]
  /dev/ram8                 [       16.00 MB]
  /dev/ram9                 [       16.00 MB]
  /dev/ram10                [       16.00 MB]
  /dev/ram11                [       16.00 MB]
  /dev/ram12                [       16.00 MB]
  /dev/ram13                [       16.00 MB]
  /dev/ram14                [       16.00 MB]
  /dev/ram15                [       16.00 MB]
```

```
  /dev/sdb                      [       17.15 GB]
  /dev/sdb1                     [       17.14 GB] LVM physical volume
  /dev/sdc                      [       17.15 GB]
  /dev/sdc1                     [       17.14 GB] LVM physical volume
  /dev/sdd                      [       17.15 GB]
  /dev/sdd1                     [       17.14 GB] LVM physical volume
  7 disks
  17 partitions
  0 LVM physical volume whole disks
  4 LVM physical volumes
```

4.2.2. Exibindo os Volumes Físicos

Existem três comandos que você pode usar para exibir as propriedades dos volumes físicos LVM: **pvs**, **pvdisplay** e **pvscan**.

O comando **pvs** fornece informações do volume físico numa forma configurável, exibindo um volume físico por linha. O comando **pvs** fornece uma grande parcela de controle de formato e é útil para criação de scripts. Para informações sobre o uso do comando **pvs** para personalizar seu resultado, veja a Seção 4.8, “Relatório Personalizado para LVM”.

O comando **pvdisplay** fornece um resultado multi linha para cada volume físico. Ele mostra as propriedades físicas (tamanho, extensão, grupo de volume, etc) num formato fixo.

Os exemplos a seguir demonstram o resultado do comando **pvdisplay** para um único volume físico.

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               new_vg
  PV Size               17.14 GB / not usable 3.40 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              4388
  Free PE               4375
  Allocated PE          13
  PV UUID               Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
```

O comando **pvscan** escaneia todos os dispositivos de bloco LVM suportados no sistema para volumes físicos.

O comando a seguir demonstra todos os dispositivos físicos encontrados:

```
# pvscan
 PV /dev/sdb2   VG vg0   lvm2 [964.00 MB / 0   free]
 PV /dev/sdc1   VG vg0   lvm2 [964.00 MB / 428.00 MB free]
 PV /dev/sdc2            lvm2 [964.84 MB]
 Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

Você pode definir um filtro no **lvm.conf** assim esse comando evitará o escaneamento de volumes físicos específicos. Para informação sobre o uso de filtros para controlar quais dispositivos são escaneados, veja a Seção 4.5, “Controlando Escaneamentos de Dispositivos LVM com Filtros” .

4.2.3. Prevenindo a Alocação em um Volume Físico

Você pode prevenir a alocação de extenções físicas no espaço livre de um ou mais volumes físicos com o comando **pvchange**. Isto será necessário se existirem erros de disco ou se você estiver removendo um volume físico.

O comando a seguir desabilita a alocação de extensões físicas no **/dev/sdk1**.

```
pvchange -x n /dev/sdk1
```

Você também pode usar os argumentos **-xy** do comando **pvchange** para permitir a alocação onde antes teria sido desabilitada.

4.2.4. Redefinindo o tamanho de um Volume Físico

Se você precisar alterar o tamanho de um dispositivo de bloco subjacente, use o comando **pvresize** para atualizar o LVM com o novo tamanho. Você pode executar este comando enquanto o LVM estiver usando o volume físico.

4.2.5. Removendo Volumes Físicos

Se um dispositivo não é mais requerido para uso pelo LVM, você pode remover a legenda LVM com o comando **pvremove**. Executando o comando **pvremove** zerará os metadados do LVM em um volume físico vazio.

Se um volume físico que você queira remover é atualmente parte de um grupo de volumes, você deve remove-lo do grupo de volume com o comado **vgreduce**, como descrito na Seção 4.3.6, “Removendo Volumes Físicos de um Grupo de Volume”.

```
# pvremove /dev/ram15
  Labels on physical volume "/dev/ram15" successfully wiped
```

4.3. ADMINISTRAÇÃO DE GRUPO DE VOLUME

Esta seção descreve os comandos que criam os diversos detalhes da administração do grupo de volume.

4.3.1. Criando Grupos de Volume

Para criar um grupo de volume de um ou mais volumes físicos, use o comando **vgcreate** O comando **vgcreate** cria um novo grupo de volume pelo nome e adiciona pelo menos um volume físico ao mesmo.

O comando a seguir cria um grupo de volume chamado **vg1** que contém volumes físicos **/dev/sdd1** e **/dev/sde1**.

```
vgcreate vg1 /dev/sdd1 /dev/sde1
```

Quando volumes físicos são usados para criar um grupo de volume, sua extenção em disco é divida em tamanhos de 4MB por padrão. Esta extenção é a quantidade mínima pela qual o volume lógico pode ser aumentado ou diminuido. Números de grande extenção não terão impacto no desempenho do volume lógico.

Você pode especificar o tamanho da extensão com a opção **-s** no comando **vgcreate** se a extensão padrão não for adequada. Você pode colocar limites nos números de volumes físicos ou lógicos que o grupo de volume pode ter usando argumentos **-p** e **-l** do comando **vgcreate**.

Por padrão, um grupo de volume aloca extensões físicas de acordo com regras de senso comum tais como não colocar distribuições paralelas no mesmo volume físico. Esta é a política de alocação **normal**. Você pode usar o argumento **--alloc** do comando **vgcreate** para especificar a política de **contiguous**, **anywhere** ou **cling**.

A política **contiguous** requer que novas extensões sejam subjacentes às extensões existentes. Se existem extensões livres suficientes para satisfazer um pedido de alocação mas uma política de alocação **normal** não os usar, a política de alocação **anywhere** o fará, mesmo que reduza o desempenho por colocar duas distribuições no mesmo volume físico. A política **cling** coloca novas extenções no mesmo volume físico assim como extenções existentes na mesma distribuição do volume lógico. Estas políticas podem ser alteradas usando o comando **vgchange**.

Para informações sobre o uso da política **cling** em conjunto com os rótulos LVM para especificar quais volumes físicos adicionais usar quando extender um volume LVM, veja a Seção 4.4.12.2, “Extendendo um Volume Lógico com a Política de Alocação **cling**.”.

No geral, as políticas de alocação que não sejam **normal** são necessárias somente em casos especiais, onde você precisará especificar alocação de extensão incomum ou não padrão.

Grupos de volume LVM e volumes lógicos subjacentes estão incluídos na árvore de diretório de arquivo especial no diretório **/dev** com o seguinte plano:

```
/dev/vg/lv/
```

Por exemplo, se você criar dois grupos de volume **myvg1** e **myvg2**, cada um com três volumes lógicos chamados **lvo1**, **lvo2** e **lvo3**, isto irá criar seis arquivos especiais de dispositivo:

```
/dev/myvg1/lv01
/dev/myvg1/lv02
/dev/myvg1/lv03
/dev/myvg2/lv01
/dev/myvg2/lv02
/dev/myvg2/lv03
```

O tamanho máximo de dispositivo com LVM é 8 Exabytes no 64-bit CPUs.

4.3.2. Criando Grupos de Volume em um Cluster

Você pode criar grupos de volume num ambiente de cluster com o comando **vgcreate**, assim como você os cria num nó único.

Por padrão, grupos de volume criados com CLVM em armazenamento compartilhado são visíveis a todos os computadores que têm acesso ao armazenamento compartilhado. É possível no entanto criar grupos de volume que sejam locais, visíveis somente a um nó no cluster usando **-c n** do comando **vgcreate**.

O seguinte comando, quando usado num ambiente de cluster, cria um grupo de volume que é local ao nó no qual o comando foi executado. O comando cria um volume local chamado **vg1** que contém volumes físicos **/dev/sdd1** e **/dev/sde1**.

```
vgcreate -c n vg1 /dev/sdd1 /dev/sde1
```

Você pode mudar se um grupo de volume existente é local ou parte do cluster com a opção **-c** do comando **vgchange**, que é descrito na Seção 4.3.7, “Modificando os Parâmetros de um Grupo de Volume”.

Você pode checar se um grupo de volume existente é um volume clusterizado com o comando **vgs**, que exibe o argumento **c** se o volume é parte de um cluster. O seguinte comando mostra os argumentos de um grupo de volumes **VolGroup00** e **testvg1**. Neste exemplo, **VolGroup00** não faz parte de um cluster, enquanto **testvg1** faz, conforme indicado pelo argumento **c** sob o título **Attr**.

```
[root@doc-07]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  VolGroup00      1   2   0 wz--n- 19.88G    0
  testvg1         1   1   0 wz--nc 46.00G 8.00M
```

Para maiores informações sobre o comando **vgs** veja a Seção 4.3.4, “Exibindo Grupos de Volume” Seção 4.8, “Relatório Personalizado para LVM” e a página man **vgs**.

4.3.3. Adicionando Volumes Físicos à um Grupo de Volume

Para adicionar volumes físicos adicionais a um grupo de volume existente, use o comando **vgextend**. O comando **vgextend** aumenta a capacidade de um grupo de volume adicionando um ou mais volumes físicos livres.

O seguinte comando adiciona o volume físico **/dev/sdf1** ao grupo de volume **vg1**.

```
vgextend vg1 /dev/sdf1
```

4.3.4. Exibindo Grupos de Volume

Existem dois comandos que você pode usar para exibir propriedades dos grupos de Volumes LVM: **vgs** e **vgdisplay**.

O comando **vgscan**, no qual escaneia todos os discos por grupos de volume e reconstrói o arquivo de cache do LVM, também exibe os grupos de volume. Para informação do comando **vgscan**,veja Seção 4.3.5, “Escaneando Discos para Grupos de Volume para Construir o Arquivo do Cache”.

O comando **vgs** fornece informações do grupo de volume numa forma configurável, exibindo uma linha por grupo de volume. O comando **vgs** fornece um grande controle de formato e é útil para criação de scripts. Para informação sobre o uso do comando **vgs** para personalizar seu resultado, veja a Seção 4.8, “Relatório Personalizado para LVM”.

O comando **vgdisplay** exibe as propriedades de um grupo de volume (tamanho, extensão, número de volumes físicos, etc) numa forma fixa. O exemplo seguinte mostra o resultado do comando **vgdisplay** para o grupo de volume **new_vg**. Se você não especificar um grupo de volume, todos os grupos de volumes existentes serão exibidos.

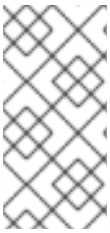
```
# vgdisplay new_vg
  --- Volume group ---
  VG Name               new_vg
  System ID
  Format                lvm2
```

```
  Metadata Areas        3
  Metadata Sequence No  11
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               51.42 GB
  PE Size               4.00 MB
  Total PE              13164
  Alloc PE / Size       13 / 52.00 MB
  Free  PE / Size       13151 / 51.37 GB
  VG UUID               jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

4.3.5. Escaneando Discos para Grupos de Volume para Construir o Arquivo do Cache

O comando **vgscan** escaneia todos os dispositivos de disco suportados no sistema procurando por volumes físicos e grupos de volume LVM. Isto constrói o cache do LVM no arquivo **/etc/lvm/.cache**, que mantém uma lista de dispositivos atuais do LVM.

O LVM roda automaticamente o comando **vgscan** ao inicializar o sistema e outras vezes durante a operação do LVM, tais como quando você executa o comando **vgcreate** ou quando o LVM detecta uma inconsistência.



NOTA

Você pode precisar rodar o comando **vgscan** manualmente quando você trocar a configuração de seu hardware e adicionar ou deletar um dispositivo de um nó, fazendo que novos dispositivos sejam visíveis ao sistema que não estavam presentes na inicialização do sistema.

Você pode definir um filtro no arquivo **lvm.conf** para restringir o escaneamento para evitar determinados dispositivos. Para informações sobre o uso de filtros para controlar quais dispositivos são escaneados, veja a Seção 4.5, “Controlando Escaneamentos de Dispositivos LVM com Filtros” .

O exemplo a seguir exibe o resultado do comando **vgscan**.

```
# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "new_vg" using metadata type lvm2
  Found volume group "officevg" using metadata type lvm2
```

4.3.6. Removendo Volumes Físicos de um Grupo de Volume

Para remover volumes físicos não utilizados de um grupo de volume, use o comando **vgreduce**. O comando **vgreduce** encolhe a capacidade de um grupo de volume removendo um ou mais volumes físicos vazios. Isso liberta os volumes físicos para serem usados em diferentes grupos de volume ou serem removidos do sistema.

Antes de remover um volume físico de um grupo de volumes, você poderá certificar-se que o volume físico não está sendo usado por qualquer volume lógico usando o comando **pvdisplay**.

```
# pvdisplay /dev/hda1

-- Physical volume ---
PV Name               /dev/hda1
VG Name               myvg
PV Size               1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#                   1
PV Status             available
Allocatable           yes (but full)
Cur LV                1
PE Size (KByte)       4096
Total PE              499
Free PE               0
Allocated PE          499
PV UUID               Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

Se o volume físico ainda está sendo usado você terá de migrar os dados para um outro volume físico usando o comando **pvmove**. Então use o comando **vgreduce** para remover o volume físico:

O seguinte comando remove o volume físico **/dev/hda1** do grupo de volume **my_volume_group**.

```
# vgreduce my_volume_group /dev/hda1
```

4.3.7. Modificando os Parâmetros de um Grupo de Volume

O comando **vgchange** é usado para desativar e ativar grupos de volumes, como descrito na Seção 4.3.8, “Ativando e Desativando os Grupos de Volume”. Você pode também usar este comando para efetuar diversas mudanças nos parâmetros para um grupo de volume existente.

O comando a seguir modifica o número máximo de volumes lógicos de um grupo de volume **vg00** para 128.

```
vgchange -l 128 /dev/vg00
```

Para uma descrição dos parâmetros do grupo de volume, você pode mudar com o comando **vgchange** veja a página man (8) **vgchange**.

4.3.8. Ativando e Desativando os Grupos de Volume

Quando você cria um grupo de volume, ele é por padrão desativado. Isto quer dizer que volumes lógicos naquele grupo estão acessíveis e sujeitos à mudança.

Existem diversas circunstâncias pelas quais você precisa tornar um grupo de volume inativo e assim desconhecidos para o kernel. Para desativar ou ativar um grupo de volume, use o argumento **-a** (**--available**) do comando **vgchange**.

O exemplo a seguir desativa o grupo de volume **my_volume_group**.

```
vgchange -a n my_volume_group
```

Se o bloqueio de cluster estiver ativado, adicione 'e' para ativar ou destivar um grupo de volume exclusivamente num nó ou 'i' para ativar ou/desativar um grupo de volume no nó local. Volumes lógicos com snapshots de host únicos são sempre ativados exclusivamente porque eles podem somente serem usados em um nó por vez.

Você pode desativar volumes lógicos indiviuais com o comando **lvchange**, como descrito na Seção 4.4.8, "Alterando os Parâmetros de um Grupo de Volume Lógico". Para informações sobre como ativar volumes lógicos em nós individuais em um cluster, veja a Seção 4.7, "Ativando Volumes Lógicos em Nós Individuais em um Cluster".

4.3.9. Removendo Grupos de Volume

Para remover um grupo de volume que não contém nenhum volume lógico, use o comando **vgremove**.

```
# vgremove officevg
  Volume group "officevg" successfully removed
```

4.3.10. Dividindo um Grupo de Volume

Para dividir os volumes físicos de um grupo de volume e criar um novo grupo de volume, use o comando **vgsplit**.

Volumes lógicos não podem ser divididos entre grupos de volume. Cada volume lógico existente deve estar inteiramente no volume físico formando o antigo ou o novo grupo de volume. Se necessário no entanto você pode usar o comando **pvmove** para forçar a divisão.

O seguinte exemplo divide o novo grupo de volumes **smallvg** do original grupo de volume **bigvg**.

```
# vgsplit bigvg smallvg /dev/ram15
  Volume group "smallvg" successfully split from "bigvg"
```

4.3.11. Combinando Grupos de Volume

Para combinar dois grupos de volumes num único grupo de volume, use o comando **vgmerge**. Você pode juntar um volume "source" com um volume "destination" ativo ou inativo se o tamanho da extensão física dos volumes são iguais e os resumos de volumes físicos e lógicos de ambos grupos de volume cabem no limite do grupo de volume destino.

O seguinte comando junta o grupo de volume inativo **my_vg** ao grupo de volume ativo ou inativo **databases** fornecendo informações de tempo de execução detalhadas.

```
vgmerge -v databases my_vg
```

4.3.12. Fazendo Cópias de Segurança de Metadados de Grupo de Volume

Backups de metadados e arquivos são automaticamente criados em todas as mudanças de configuração em grupos de volumes e volumes lógicos ao menos que sejam desativados no arquivo **lvm.conf**. Por padrão, o backup de metadados é guardado no arquivo **/etc/lvm/backup** e os arquivos de metadados são guardados no arquivo **/etc/lvm/archives**. Você pode manualmente criar um backup de metadados no arquivo **/etc/lvm/backup** com o comando **vgcfgbackup**.

O comando **vgcfrestore** restaura os metadados de um grupo de volume a partir do arquivo para todos os volumes físicos nos grupos de volume.

Para um exemplo do uso do comando **vgcfgrestore** para recuperar metadados do volume físico, veja a Seção 6.4, “Recuperando Metadados de Volume Físico”.

4.3.13. Renomeando um Grupo de Volume

Use o comando **vgrename** para renomear um grupo de volume existente.

Quaisquer dos seguintes comandos renomeia os grupos de volumes existentes **vg02** para **my_volume_group**

```
vgrename /dev/vg02 /dev/my_volume_group
```

```
vgrename vg02 my_volume_group
```

4.3.14. Movendo um Grupo de Volume para Outro Sistema

Você pode mover um grupo de volume LVM inteiro para outro sistema. É recomendado que você use os comandos **vgexport** e **vgimport** quando fizer isso.

O comando **vgexport** torna um grupo de volume inativo inacessível para o sistema, o qual permite que você retire o volume físico. O comando **vgimport** torna um grupo de volume accessível para uma máquina novamente após o comando **vgexport** te-lo desativado.

Para mover um grupo de volume de um sistema para outro, realize os seguintes passos:

1. Certifique-se de que não existem usuários acessando documentos nos volumes ativos no grupo de volume e depois desmonte os volumes lógicos.

2. Use o argumento **-a n** do comando **vgchange** para marcar o grupo de volume como inativo, o qual previne qualquer outra atividade no grupo de volume.

3. Use o comando **vgexport** para exportar o grupo de volume. Isto previne o acesso pelo sistema no qual você está removendo.

   Depois de você exportar o grupo de volume, o volume físico exibirá como estando num grupo de volume exportado quando você executar o comando **pvscan**, como no seguinte exemplo.

   ```
   [root@tng3-1]# pvscan
     PV /dev/sda1    is in exported VG myvg [17.15 GB / 7.15 GB free]
     PV /dev/sdc1    is in exported VG myvg [17.15 GB / 15.15 GB free]
     PV /dev/sdd1   is in exported VG myvg [17.15 GB / 15.15 GB free]
     ...
   ```

   Quando o sistema estiver desligado, você pode desplugar os discos que constituem o grupo de volumes e conecta-los no novo sistema.

4. Quandos os discos estiverem plugados no novo sistema, use o comando **vgimport** para importar o grupo de volume, o fazendo acessível para o novo sistema.

5. Ative o grupo de volume com o argumento **-a y** do comando **vgchange**.

6. Monte o sistema de arquivo para disponibilizá-lo para uso.

4.3.15. Recriando um Diretório de Grupo de Volume

Para recriar um diretório de grupo de volume e arquivos especiais de volumes lógicos, use o comando **vgmknodes**. Este comando checa os arquivos especiais do LVM2 dentro do diretório **/dev** que é necessário para ativar volumes lógicos. Isso cria qualquer arquivo especial que estiver faltando e remove os não usados.

Você pode incorporar o comando **vgmknodes** ao comando **vgscan** especificando o argumento **mknodes** ao comando **vgscan**.

4.4. ADMINISTRAÇÃO DE VOLUME LÓGICO

Esta seção descreve os comandos que configuram os diversos aspectos de uma administração de volume lógico.

4.4.1. Criando Volumes Lógicos Lineares

Para criar um volume lógico, use o comando **lvcreate**. Se você não especificar um nome para o volume lógico, o nome padrão **lvol#** é usado onde *#* é o número interno do volume lógico.

Quando você cria um volume lógico, este é feito do grupo de volume usando as extensões livres no volume físico que compõem o grupo de volume. Normalmente os volumes lógicos usam qualquer espaço disponível no volume físico subjacente. Modificando o volume lógico libera e realoca o espaço nos volumes físicos.

O comando a seguir cria um volume lógico de 10 gigabytes no grupo de volume **vg1**.

```
lvcreate -L 10G vg1
```

O seguinte comando cria um volume lógico e linear de 1500 megabytes chamado **testlv** no grupo de volume **testvg**, criando o bloco de dispositivo **/dev/testvg/testlv**.

```
lvcreate -L1500 -n testlv testvg
```

O seguinte comando cria um volume lógico de 50 gigabytes chamado **gfslv** a partir das extensões livres do grupo de volume **vg0**.

```
lvcreate -L 50G -n gfslv vg0
```

Você pode usar o argumento **-l** do comando **lvcreate** para especificar o tamanho do volume lógico em extensão. Você pode também usar este argumento para especificar a porcentagem que o grupo de volume usará para o volume lógico. O seguinte comando cria um volume lógico chamado **mylv** que usa 60% do espaço total no grupo de volume **testvol**.

```
lvcreate -l 60%VG -n mylv testvg
```

Você pode também usar o argumento **-l** do comando **lvcreate** para especificar a porcentagem restante do espaço livre num grupo de volume como o tamanho do volume lógico. O seguinte comando cria um volume lógico chamado **yourlv** que usa todo o espaço não alocado no grupo de volume **testvol**.

```
lvcreate -l 100%FREE -n yourlv testvg
```

Você pode usar o argumento **-l** do comando **lvcreate** para criar um volume lógico que usa o grupo de volume inteiro. Uma outra maneira de criar um volume lógico que usa todo o grupo de volume é usar o comando **vgdisplay** para encontrar o tamanho "Total PE" e usar esses resultados como entrada no comando **lvcreate**.

O seguinte comando cria um volume lógico chamado **mylv** que preenche o grupo de volume chamado **testvg**.

```
# vgdisplay testvg | grep "Total PE"
Total PE              10230
# lvcreate -l 10230 testvg -n mylv
```

Os volumes físicos subjacentes usados para criar um volume lógico podem ser importantes se o volume físico precisa ser removido, então você deve considerar esta possibilidade quando criar o volume lógico. Para informações sobre remover um volume físico de um grupo de volumes, veja a Seção 4.3.6, “Removendo Volumes Físicos de um Grupo de Volume”.

Para criar um volume lógico a ser alocado a partir de um volume físico específico no grupo de volume, especifique o ou os volumes físicos no final da linha de comando **lvcreate**. O seguinte comando cria um volume lógico chamado **testlv** no grupo de volume **testvg** alocado a partir do volume físico **/dev/sdg1**,

```
lvcreate -L 1500 -ntestlv testvg /dev/sdg1
```

Você pode especificar quais extensões de um volume físico serão usadas por um volume lógico. O exemplo seguinte cria um volume lógico linear de extensão 0 a 25 do volume físico **/dev/sda1** e de 50 a 124 do volume físico **/dev/sdb1** no grupo de volume **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-24 /dev/sdb1:50-124
```

O exemplo seguinte cria um volume lógico linear de extensão 0 a 25 do volume físico **/dev/sda1** e continua no volume lógico à extensão 100.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25:100-
```

A política padrão para o quanto a extensão de um volume lógico é alocada é **inherit**, a qual aplica a mesma política para o grupo de volume. Estas políticas pode sem alteradas usando o comando **lvchange**. Para informações sobre políticas de alocação, veja a Seção 4.3.1, “Criando Grupos de Volume”.

4.4.2. Criação de Volumes Distribuídos

Para grandes sequências de leitura e escrita, criar um volume lógico distribuído pode melhorar a eficiência dos dados de E/S. Para informações gerais sobre volumes distribuídos, veja a Seção 2.3.2, “Volumes Lógicos Distribuídos”.

Quando você criar um volume lógico distribuído, você especifica o número de distribuições com o argumento **-i** do comando **lvcreate**. Isto determina em quantos volumes físicos o volume lógico será distribuído. O número de distribuições não pode ser maior que o número de volumes físicos no grupo de volume (a menos que o argumento **--alloc anywhere** seja usado).

Se os dispositivos físicos subjacentes que compõem um volume lógico distribuído são de tamanhos diferentes, o tamanho máximo do volume distribuído é determinado pelo menor dispositivo subjacente. Por exemplo, em uma distribuição de duas pernas, o tamanho máximo é o dobro do tamanho do menor dispositivo. Em uma distribuição de três pernas, o tamanho máximo é de três vezes o tamanho do menor dispositivo.

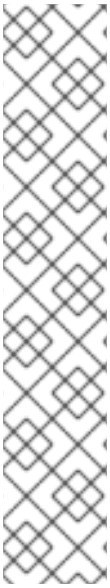
O seguinte comando cria um volume lógico distribuído sobre 2 volumes físicos com uma distribuição de 64kB. O volume lógico é 50 gigabytes em tamanho, é nomeado **gfslv**, and é feito a partir do grupo de volume **vg0**.

```
lvcreate -L 50G -i2 -I64 -n gfslv vg0
```

Como nos volumes lineares, você pode especificar as extensões do volume físico que você está usando para a distribuição. O seguinte comando cria um volume distribuído de extensão 100 em tamanho que se divide em dois volumes físicos, é chamado **stripelv** e está no grupo de volume **testvg**. A distribuição usará os setores 0-49 do **/dev/sda1** e setores 50-99 de **/dev/sdb1**.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-49 /dev/sdb1:50-99
  Using default stripesize 64.00 KB
  Logical volume "stripelv" created
```

4.4.3. Criando Volumes Espelhados



NOTA

Criando um volume lógico espelhado LVM em um cluster requer os mesmos comandos e procedimentos quando criando um volume lógico espelhado LVM em um nó único. Entretanto, para criar um volume espelhado em um cluster, o cluster e o espelho de cluster devem estar em atividade, o cluster deve estar quórum e o tipo de bloqueio no arquivo **lvm.conf** deve estar configurado corretamente para permitir bloqueio de cluster. Para um exemplo de criação de um volume espelhado em um cluster, veja a Seção 5.5, “Criando um Volume Lógico LVM Espelhado em um Cluster”.

Tentar executar a criação de múltiplos espelhos LVM e executar comandos de conversão em uma rápida sucessão de múltiplos nodos em um cluster poderá causar uma lista de pendências destes comandos. Isto poderá causar em algumas das operações requisitadas expirar o tempo limite e subsequentemente falhar. Para evitar este problema, é recomendado que os comandos de criação do espelho de cluster sejam executados a partir de um nó do cluster.

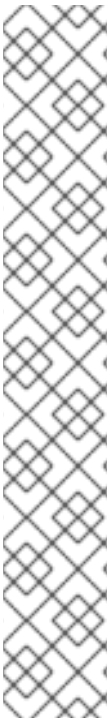
Quando você cria um volume espelhado, você especifica o número de cópias de dados a serem feitas com o argumento **-m** do commando **lvcreate**. Especificando **-m1** se cria um espelho, que produz duas cópias do sistema de arquivos: um volume lógico linear mais uma cópia. Similarmente, especificando **-m2** se cria dois espelhos, produzindo três cópias do sistema de arquivos.

O seguinte comando cria um volume lógico espelhado com um único espelho. O volume é de 50 gigabytes em tamanho, é chamado **mirrorlv** e é feito do grupo de volume **vg0**:

```
lvcreate -L 50G -m1 -n mirrorlv vg0
```

Um espelho LVM divide o dispositivo sendo copiado em regiões que, por padrão, são 512KB em tamanho. Você pode usar o argumento **-R** do comando **lvcreate** para especificar o tamanho da região em MB. Você pode também alterar o tamanho padrão da região editando a configuração

mirror_region_size no arquivo **lvm.conf**.



NOTA

Devido às limitações na infra estrutura do cluster, espelhos de cluster maiores que 1.5TB não podem ser criados com a região padrão de tamanho 512KB. Usuários que requerem espelhos maiores devem aumentar o tamanho da região a partir da padrão para algo maior. Falha no aumento de tamanho da região causará travamento na criação do LVM e pode travar outros comando do LVM também.

Como um guia geral para especificar o tamanho da região de espelhos que são maiores que 1.5TB, you pode pegar o tamanho de seu espelho em terabytes e arredondar para cima esse número para a próxima potência de 2, usando esse número como o argumento **-R** do comando **lvcreate**. Por exemplo, se o tamanho de seu espelho é 1.5TB, você pode especificar **-R 2**. Se o tamanho de seu espelho é 3TB, você pode especificar **-R 4**. Para um espelho de tamanho 5TB, você pode especificar **-R 8**.

O seguinte comando cria um volume lógico espelhado com uma região de tamanho de 2MB:

```
lvcreate -m1 -L 2T -R 2 -n mirror vol_group
```

O LVM mantém um pequeno log que é usado para manter controle de quais regiões estão sincronizadas com o espelho ou espelhos. Por padrão, este log é mantido no disco, mantendo-o persistente entre as reinicializações e garante que o espelho não tenha de ser re-sincronizado toda vez que a máquina reinicializar ou travar. Você pode especificar no entanto que este log será mantido na memória com o argumento **--mirrorlog core**; isto elimina a necessidade de um dispositivo de log extra, mas requer que o espelho inteiro seja re-sicronizado em toda reinicialização.

O comando seguinte cria um volume lógico espelhado a partir do grupo de volume **bigvg**. O volume lógico é nomeado **ondiskmirvol** e tem um espelho único. O volume é 12MB em tamanho e mantém o log do espelho na memória.

```
# lvcreate -L 12MB -m1 --mirrorlog core -n ondiskmirvol bigvg
  Logical volume "ondiskmirvol" created
```

O log do espelho é criado em um dispositivo separado dos dispositivos nas quais qualquer perna do espelho é criada. É possível no entanto criar um log do espelho no mesmo dispositivo quanto em uma das pernas do espelho usando o argumento **--alloc anywhere** do comando **vgcreate**. Isto pode degradar a performance, mas permite que você crie um espelho mesmo se tiver somente dois dispositivos subjacentes.

O comando a seguir cria um volume lógico espelhado com um espelho único para qual o log do espelho está no mesmo dispositivo das pernas do espelho. Neste exemplo, o grupo de volume **vg0** consiste em apenas dois dispositivos. Este comando cria um volume de 500 megabytes chamado **mirrorlv** no grupo de volume **vg0**.

```
lvcreate -L 500M -m1 -n mirrorlv -alloc anywhere vg0
```

NOTA

Com espelhos clusterizados, o gerenciador do log de espelho é completamente responsável pelo nodo do cluster com o menor

Para criar um log de espelho que é auto espelhado, você pode especificar o argumento **--mirrorlog mirrored**. O seguinte comando cria um volume lógico espelhado a partir do grupo de volume **bigvg**. O volume lógico é chamado **twologvol** e tem um espelho único. O volume tem 12MB em tamanho e o log é espelhado, com cada log mantido num dispositivo separado.

```
# lvcreate -L 12MB -m1 --mirrorlog mirrored -n twologvol bigvg
  Logical volume "twologvol" created
```

Assim como ocorre com um log de espelho padrão, é possível criar logs de espelhos redundantes no mesmo dispositivo quanto nas pernas do espelho usando o argumento **--alloc anywhere**do comando **vgcreate**. Isso pode degradar o desempenho, mas permite que você crie um log de espelho mesmo se você não tiver dispositivos subjacentes para cada log a serem mantidos em dispositivos separados do que pernas de espelhos.

Quando um espelho é criado, as regiões do espelho são sincronizadas. Para maiores dispositivos de espelho, o processo de sincronia pode levar um longo tempo. Quando você estiver criando um novo espelho que não precisa ser recuperado, você pode especificar o argumento **nosync** para inidicar que uma sincronização inicial a partir do primeiro dispositivo não é necessária.

Você pode especificar quais dispositivos usar para as pernas do espelho e log e quais extensões destes dispositivos usar. Para forçar o log num determinado disco, especifique exatamente uma extensão no disco no qual este será instalada. O LVM não respeita necessariamente a ordem no qual os dispositivos estão listados na linha de comando. Se quaisquer volumes físicos estão listados como o único espaço no qual a alocação será realizada. Qualquer extensão física incluída na lista que já está alocada, será ignorada.

O seguinte comando cria um volume lógico espelhado com espelho único e um log único não espelhado. O volume é 500 megabytes em tamanho, chamado **mirrorlv** e é feito do grupo de volume **vg0**. A primeira perna do espelho está no dispositivo **/dev/sda1**, a segunda perna do espelho está no dispositivo **/dev/sdb1** e o log de espelho está no **/dev/sdc1**.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1 /dev/sdb1 /dev/sdc1
```

O seguinte comando cria um volume lógico espelhado com um espelho simples. O volume tem 500 megabytes em tamanho e é chamado **mirrorlv** e é feito do grupo de volume **vg0**. A primeira perna do volume tem extensão de 0 a 499 do dispositivo **/dev/sda1**, a segunda perna do espelho tem extensão 0 a 499 do dispositivo **/dev/sdb1** e o log do espelho inicia na extensão 0 do dispositivo **/dev/sdc1**. Estes possuem extensão de 1MB. Se qualquer das extensões já foram alocadas, elas serão ignoradas.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499
/dev/sdc1:0
```

NOTA

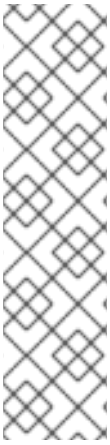
A partir do lançamento do Red Hat Enterprise Linux 6.1, você pode combinar RAID0(distribuição) e RAID1 (espelhamento) em um volume lógico único. Criando um volume lógico enquanto simultaneamente especifica o número de espelhos (**--mirrors X**) e o número de distribuições (**--stripes Y**) resulta em um dispositivo de espelho dos quais dispositivos constituintes são distribuídos.

4.4.3.1. Política de Falha do Volume Lógico Espelhado

Você pode definir como um volume lógico espelhado se comporta em um evento de falha de dispositivo com os parâmetros **mirror_image_fault_policy** e **mirror_log_fault_policy** na seção **activation** do arquivo **lvm.conf**. Quando estes parâmetros estão configurados em **remove**, o sistema tenta remover o dispositivo com defeito e rodar sem ele. Quando este parâmetro está configurado como **allocate**, o sistema tenta remover o dispositivo com defeito e tenta alocar espaço em um dispositivo novo para ser um substituto do dispositivo defeituoso; esta política é como a política **remove** se não houver nenhum dispositivo ou espaćo adequados para a substituição.

Por padrão, o parâmetro **mirror_log_fault_policy** é configurado para **allocate**. Usar esta política para o log é rápida e mantém a habilidade de lembrar o estado sync durante travamentos e reinicializações. Se você configurar esta política para **remove**, quando um log de dispositivo falhar, o espelho converte para o uso de um log de memória e o espelho não lembrará seu estado de sincronia entre travamentos e reinicializações e o espelho inteiro será re-sincronizado.

Por padrão, o parâmetro **mirror_image_fault_policy** é configurado para **remove**. Com esta política, se uma imagem de espelho falha o espelho converterá para um dispositivo não espelhado se houver somente uma boa cópia remanescente. Configurando esta política para **allocate** para um dispositivo de espelho requer que o espelho resincronize os dispositivos; isto é um processo lento, mas preserva as características de espelho do dispositivo.



NOTA

Quando um espelho LVM sofrer uma falha de dispositivo, uma recuperação de dois estágios toma lugar. O primeiro estágio envolve remover os dispositivos com falha. Isto pode resultar em um espelho sendo reduzido a um dispositivo linear. O segundo estágio, se o parâmetro **mirror_log_fault_policy** estiver configurado para **allocate**, é para tentar substituir quaisquer dos dispositivos com falha. Nota, entretanto, não há garantia que o segundo estágio escolherá dispositivos anteriormente em uso pelo espelho que não foi parte da falha se outros estiverem disponíveis.

Para informações sobre recuperar manualmente de uma falha de espelho LVM, consulte a Seção 6.3, “Recuperação de Falha do Espelho LVM.”.

4.4.3.2. Dividindo uma Imagem Redundante de um Volume Lógico Espelhado

Você pode dividir uma imagem redundante de um volume lógico espelhado para formar um novo volume lógico. Para dividir imagem, você usa o argumento **--splitmirrors** do comando **lvconvert**, especificando o número de imagens redundantes a serem divididas. Você deve usar o argumento **--name** do comando para especificar o nome para o volume recém dividido.

O seguinte comando divide um novo volume lógico chamado **copy** do volume lógico espelhado **vg/lv**. O novo volume lógico conterá duas pernas de espelho. Neste exemplo, o LVM seleciona quais dispositivos a serem divididos.

```
lvconvert --splitmirrors 2 --name copy vg/lv
```

Você pode especificar quais dispositivos serão retirados. O comando seguinte retira um novo volume lógico chamado **copy** do volume lógico espelhado **vg/lv**. O novo volume lógico contém duas pernas de espelho composto dos dispositivos **/dev/sdc1** e **/dev/sde1**.

```
lvconvert --splitmirrors 2 --name copy vg/lv /dev/sd[ce]1
```

4.4.3.3. Reparando um Dispositivo Lógico Espelhado

Você pode usar o comando **lvconvert --repair** para reparar um espelho após uma falha de disco. Isto trará o espelho de volta num estado consistente. O comando **lvconvert --repair** é um comando interativo que lhe pede para indicar se você quer que o sistema tente repor qualquer dispositivo com falha.

- Para ignorar os avisos e substituir todos os dispositivos com falha, especifique a opção **-y** na linha de comando.
- Para ignorar os avisos e não substituir nenhum dos dispositivos com falha, especifique a opção **-f** na linha de comando.
- Para ignorar os avisos e ainda indicar diferentes políticas de substituição para o espelho de imagem e o log de espelho, você pode especificar o argumento **--use-policies** para usar a política de substituição de dispositivo especificadas pela **mirror_log_fault_policy** e parâmetros **mirror_device_fault_policy** no arquivo **lvm.conf**.

4.4.3.4. Trocando a Configuração de Volume Espelhado

Você pode converter um volume lógico a partir de volume espelhado para um linear ou a partir de um volume linear para um volume espelhado com o comando **lvconvert**. Você pode também usar este comando para reconfigurar outros parâmetros do espelho de um volume lógico existente, tal como **corelog**.

Quando você converter um volume lógico para um volume espelhado, você está basicamente criando pernas de espelho para um volume existente. Isto significa que seu grupo de volume deve conter os dispositivos e espaço para as pernas do espelhos e para o log do espelho.

Se você perder a perna de um espelho, o LVM converte o volume para volume linear para que você ainda possa acessar o volume, sem redundancia de espelho. Depois que você substituir a perna, você pode usar o comando **lvconvert** para restaurar o espelho. Este procedimento é fornecido na Seção 6.3, “Recuperação de Falha do Espelho LVM.”.

O comando seguinte converte o volume lógico linear **vg00/lvol1** para um volume lógico espelhado.

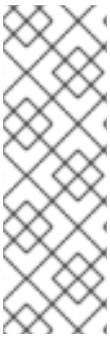
```
lvconvert -m1 vg00/lvol1
```

O seguinte comando converte um volume lógico espelhado **vg00/lvol1** para um volume lógico linear, removendo a perna do espelho.

```
lvconvert -m0 vg00/lvol1
```

4.4.4. Criando Volumes Snapshots

Use o argumento **-s** do comando **lvcreate** para criar um volume snapshot. Um volume snapshot é gravável.



NOTA

Os snapshots LVM não são suportados através de nós em um cluster. Você não pode criar um volume snapshot em um grupo de volume clusterizado. A partir do lançamento do Red Hat Enterprise Linux 6.1, entretanto, se você precisar criar um backup consistente de dados em um volume lógico clusterizado você pode ativar o volume exclusivamente e então criar o snapshot. Para informações sobre ativar volumes lógicos exclusivamente em um nó, veja a Seção 4.7, “Ativando Volumes Lógicos em Nós Individuais em um Cluster”.

NOTA

A partir do lançamento do Red Hat Enterprise Linux 6.1, snapshots LVM são suportados por volumes lógicos espelhados.

O seguinte comando cria um volume lógico snapshot que tem 100 megabytes em tamanho e chamado **/dev/vg00/snap**. Isto cria um snapshot do volume lógico original chamado **/dev/vg00/lvol1**. Se o volume lógico original contém um sistema de arquivos, você pode montar o volume lógico snapshot em um diretório arbitrário para pode acessar o conteúdo do sistema de arquivo e fazer um backup enquanto o sistema de arquivos original é atualizado.

```
lvcreate --size 100M --snapshot --name snap /dev/vg00/lvol1
```

Depois que você cria um volume lógico snapshot, especificando o volume de origem no comando **lvdisplay**, receberá um resultado que inclui uma lista de todos os volumes lógicos snapshots e seus estados (ativos ou inativos).

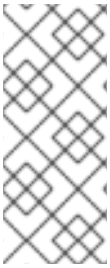
O seguinte exemplo mostra o estado do volume lógico **/dev/new_vg/lvol0**, para o qual o volume snapshot **/dev/new_vg/newvgsnap** foi criado.

```
# lvdisplay /dev/new_vg/lvol0
  --- Logical volume ---
  LV Name                /dev/new_vg/lvol0
  VG Name                new_vg
  LV UUID                LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  LV Write Access        read/write
  LV snapshot status     source of
                         /dev/new_vg/newvgsnap1 [active]
  LV Status              available
  # open                 0
  LV Size                52.00 MB
  Current LE             13
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:2
```

O comando **lvs**, por padrão, exibe o volume de origem e a percentagem atual sendo usados para cada volume snapshot. O seguinte exemplo exibe o resultado padrão para o comando **lvs** para um sistema que inclui o volume lógico **/dev/new_vg/lvol0**, para o qual um volume snapshot

/dev/new_vg/newvgsnap foi criado.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```



NOTA

Pelo motivo que o snapshot aumenta em tamanho conforme o volume de origem muda, é importante monitorar a porcentagem do volume snapshot regularmente com o comando **lvs** para certificar que este não fique cheio. Um snapshot que está 100% cheio é perdido completamente, já que uma gravação em partes não alteradas na origem não aconteceriam sem corromper o snapshot.

4.4.5. Incorporando Volumes Snapshots

Com o lançamento do Red Hat Enterprise Linux 6, você pode usar a opção **--merge** do comando **lvconvert** para incorporar um snapshot em seu volume de origem. Se ambos origem e snapshot não estão abertos, a incorporação iniciará imediatamente. Caso contrário, a incorporação iniciará tanto estejam a origem e snapshot ativados e ambos fechados. Incorporar um snapshot em uma origem que não pode ser fechada, por exemplo, o sistema de arquivos root será deferido até a próxima vez que o volume de origem seja ativado. Quando a incorporação inicia, o volume lógico resultante terá o nome de origem, número menor e UUID. Enquanto a incorporação estiver em progresso, leitura e escrita na origem aparecem como se fossem direcionadas para o snapshot sendo incorporado. Quando o processo termina, o snapshot incorporado é removido.

O seguinte comando incorpora o volume snapshot **vg00/lvol1_snap** em sua origem.

```
lvconvert --merge vg00/lvol1_snap"
```

Você pode especificar múltiplos snapshots na linha de comando ou você pode usar as tags de objeto do LVM que especificam que múltiplos snapshots sejam incorporados em suas respectivas origens. No exemplo seguinte, os volumes lógicos **vg00/lvol1**, **vg00/lvol2** e **vg00/lvol3** recebem a tag **@some_tag**. O seguinte comando incorpora os volumes lógicos snapshots para os três volumes serialmente: **vg00/lvol1**, então **vg00/lvol2**, então **vg00/lvol3**. Se a opção **--background** foi usada, todos os volumes lógicos snapshots iniciariam em paralelo.

```
lvconvert --merge @some_tag"
```

Para mais informações sobre rotular objetos LVM, veja o Apêndice C, *Tags de Objetos do LVM*. Para mais informações sobre o comando **lvconvert --merge**, veja a página man **lvconvert**(8).

4.4.6. Números de Dispositivos Persistentes

Números de dispositivos maiores e menores são alocados dinamicamente no carregamento do módulo. Algumas aplicações funcionam melhor se o dispositivo de bloco estiver sempre ativado com o mesmo número de dispositivo (maior e menor). Você pode especificar estes com os comandos **lvcreate** e **lvchange** usando os seguintes argumentos:

```
--persistent y --major major --minor minor
```

Use um número menor alto para ter certeza de que já não tenha sido atribuído a um outro dispositivo dinamicamente.

Se você estiver exportando um arquivo de sistema usando NFS, especificando o parâmetro nos arquivos de exportação pode evitar a necessidade de configurar um número de dispositivo persistente dentro do LVM.

4.4.7. Redimensionando Volumes Lógicos

Para reduzir o tamanho de um volume lógico, use o comando **lvreduce**. Se o volume lógico contém um arquivo de sistema, certifique-se de reduzir o arquivo de sistema primeiro (ou use o GUI do LVM) para que esse volume lógico esteja com o tamanho necessário que o arquivo de sistema necessita.

O seguinte comando reduz o tamanho do volume lógico **lvol1** no grupo de volume **vg00** por 3 extensões lógicas.

```
lvreduce -l -3 vg00/lvol1
```

4.4.8. Alterando os Parâmetros de um Grupo de Volume Lógico

Para alterar os parâmetros de um volume lógico, use o comando **lvchange**. Para uma lista dos parâmetros que você pode alterar veja a página man **lvchange**(8)

Você pode usar o comando **lvchange** para ativar e desativar volumes lógicos. Para ativar e desativar todos os volumes lógicos em um grupo de volume ao mesmo tempo, use o comando **vgchange**, conforme descrito na Seção 4.3.7, “Modificando os Parâmetros de um Grupo de Volume”.

O seguinte comando altera as permissões no volume **lvol1** grupo de volume **vg00** para somente leitura.

```
lvchange -pr vg00/lvol1
```

4.4.9. Renomeando Volumes Lógicos

Para renomear um volume lógico existente, use o comando **lvrename**.

Qualquer dos seguintes comandos renomeiam os volumes lógicos **lvold** no grupo de volume **vg02** para **lvnew**.

```
lvrename /dev/vg02/lvold /dev/vg02/lvnew
```

```
lvrename vg02 lvold lvnew
```

Para mais informações sobre ativação de volumes lógicos em nós individuais num cluster, veja a Seção 4.7, “Ativando Volumes Lógicos em Nós Individuais em um Cluster”.

4.4.10. Removendo Volumes Lógicos

Para remover um volume lógico inativo, use o comando **lvremove**. Se o volume lógico estiver atualmente montado, desmonste o volume antes de remove-lo. E em um ambiente de cluster você deve desativar um volume lógico antes de remove-lo.

O seguinte comando remove o volume lógico **/dev/testvg/testlv** do grupo de volume **testvg**. Note que neste caso o volume lógico não foi desativado.

```
[root@tng3-1 lvm]# lvremove /dev/testvg/testlv
Do you really want to remove active logical volume "testlv"? [y/n]: y
  Logical volume "testlv" successfully removed
```

Você poderia desativar explicitamente o volume lógico antes de remove-lo com o comando **lvchange -an**, neste caso você não seria perguntado se quer remover um volume lógico ativo.

4.4.11. Exibindo Volumes Lógicos

Existem três comandos que você pode usar para exibir as propriedades dos volumes lógicos LVM: **lvs**, **lvdisplay** e **lvscan**.

O comando **lvs** fornece informações sobre volumes lógicos em uma forma configurável, exibindo um volume lógico por linha, O comando **lvs** fornece uma grande variedade de controle de formatos, e é útil para criação de scripts. Para informações sobre o uso do comando **lvs** para personalizar seus resultados, veja a Seção 4.8, “Relatório Personalizado para LVM”.

O comando **lvdisplay** exibe as propriedades do volume lógico (como tamanho, layout e mapeamento) em um formato fixo.

O comando seguinte exibe os atributos de **lvol2** no **vg00**. Se um volume lógico snapshot foi criado para este volume lógico original, este comando mostra uma lista de todos os volumes lógicos snapshots e seus estados (ativos ou inativos) também.

```
lvdisplay -v /dev/vg00/lvol2
```

O comando **lvscan** escaneia por todos os volumes lógicos no sistema e os lista, como no exemplo a seguir.

```
# lvscan
 ACTIVE                   '/dev/vg0/gfslv' [1.46 GB] inherit
```

4.4.12. Aumentando Volumes Lógicos

Para aumentar o tamanho de um volume lógico, use o comando **lvextend**.

Quando você extender um volume lógico, você pode indicar o quanto você quer extender o volume ou o tamanho que você quer que seja depois de extender-lo.

O seguinte comando extende o volume lógico **/dev/myvg/homevol** para 12 gigabytes.

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

O seguinte comando adiciona outro gigabyte para o volume lógico **/dev/myvg/homevol**.

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
```

```
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

Como no comando **lvcreate**, você pode usar o argumento **-l** do comando **lvextend** para especificar o número de extensões pelas quais aumentar o tamanho do volume lógico. Você pode também usar este argumento para especificar a porcentagem do grupo de volume ou uma porcentagem do espaço livre restante no grupo de volume. O comando seguinte extende o volume lógico chamado **testlv** para preencher todo o espaço não alocado no grupo de volume**myvg**.

```
[root@tng3-1 ~]# lvextend -l +100%FREE /dev/myvg/testlv
  Extending logical volume testlv to 68.59 GB
  Logical volume testlv successfully resized
```

Depois de você ter extendido o volume lógico é necessário aumentar o tamanho do sistema de arquivos para equivalencia.

Por padrão, a maioria das ferramentas de redimensionamento do sistema de arquivos aumentarão o tamanho do sistema de arquivos para ter o tamanho do volume lógico subjacente para que você não precise se preocupar em especificar o mesmo tamanho para cada um dos dois comandos.

4.4.12.1. Extendendo um Volume Distribuído

Para aumentar o tamanho de um volume lógico distribuído, deverá haver espaço livre suficiente nos volumes físicos subjacentes que compõem o grupo de volume para suportar a distribuição. Por exemplo, se você tem uma distribuição de duas vias que usa um grupo de volume inteiro, adicionando um único volume físico ao grupo de volume não permitirá que você extenda a distribuição. Ao invés disso, você deve adicionar ao menos dois volumes físicos ao grupo de volume.

Por exemplo, considere um grupo de volume **vg** que consista em dois volumes físicos subjacentes, como mostrado com o seguinte comando **vgs**.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   0   0 wz--n- 271.31G 271.31G
```

Você pode criar uma distribuição usando a quantidade inteira de espaço no grupo de volume.

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
  Using default stripesize 64.00 KB
  Rounding up size to full physical extent 271.31 GB
  Logical volume "stripe1" created
# lvs -a -o +devices
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe1 vg   -wi-a- 271.31G
/dev/sda1(0),/dev/sdb1(0)
```

Note que o grupo de volume não possui mais espaço livre.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   1   0 wz--n- 271.31G    0
```

O comando seguinte adiciona um outro volume físico ao grupo de volume, no qual então tem 135G de espaço adicional.

```
# vgextend vg /dev/sdc1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     3   1   0 wz--n- 406.97G 135.66G
```

Neste momento você não pode extender o volume lógico distribuído para o volume total, porque dois dispositivos subjacentes são necessários para dividir os dados.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1:
34480
more required
```

Para extender o volume lógico distribuído, adicione um outro volume físico e então extenda o volume lógico. Neste exemplo, adicionando dois volumes físicos ao grupo de volume nós poderemos extender o volume lógico ao máximo tamanho do grupo de volume.

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 542.00 GB
  Logical volume stripe1 successfully resized
```

Se você não possui dispositivos físicos subjacentes suficientes para extender o volume lógico distribuído, é possível extender o volume de qualquer maneira se não for importante que a extensão não seja em distribuições, o que pode resultar em um desempenho desigual. Quando estiver adicionando espaço ao volume lógico, a operação padrão é usar os mesmos parâmetros de distribuição do último segmento do volume lógico existente, mas você pode mudar estes parâmetros. O exemplo seguinte extende o volume lógico distribuído existente para usar o espaço livre restante depois que o comando inicial **lvextend** ter falhado.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1:
34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

4.4.12.2. Extendendo um Volume Lógico com a Política de Alocação `cling`.

Quando extender um volume LVM, você pode usar a opção **`--alloc cling`** do comando **lvextend** para especificar a política de alocação **`cling`**. Esta política escolherá espaço nos mesmos volumes físicos como no último segmento do volume lógico existente. Se há espaço insuficiente nos volumes

físicos e uma lista de rótulos é definida no arquivo **lvm.conf**, o LVM checará se qualquer dos rótulos estão anexados aos volumes físicos e buscam coincidir aqueles rótulos de volume físico entre extensões existentes e novas extensões.

Por exemplo, se você possui volumes lógicos que estão espelhados entre dois lugares dentro de um grupo de volume único, você pode rotular os volumes físicos de acordo com onde eles estão situados rotulando os volumes físicos com tags @site1 e @site2 e especificar a seguinte linha no arquivo **lvm.conf**:

```
cling_tag_list = [ "@site1", "@site2" ]
```

Para informações sobre como rotular volumes físicos, veja o Apêndice C, *Tags de Objetos do LVM*.

No exemplo seguinte, o arquivo **lvm.conf** foi modificado para conter a seguinte linha:

```
cling_tag_list = [ "@A", "@B" ]
```

Também neste exemplo, um grupo de volume **taft** foi criado e consiste nos volumes físicos **/dev/sdb1**, **/dev/sdc1**, **/dev/sdd1**, **/dev/sde1**, **/dev/sdf1**, **/dev/sdg1**, e **/dev/sdh1**. Estes volumes físicos foram rotulados com as tags **A**, **B** e **C**. O exemplo não usa a tag **C**, mas isto mostrará que o LVM usa os rótulos para selecionar quais volumes físicos usar para as pernas do espelho.

```
[root@taft-03 ~]# pvs -a -o +pv_tags /dev/sd[bcdefgh]1
  PV         VG   Fmt  Attr PSize   PFree   PV Tags
  /dev/sdb1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdc1  taft lvm2 a-   135.66g 135.66g B
  /dev/sdd1  taft lvm2 a-   135.66g 135.66g B
  /dev/sde1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdf1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdg1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdh1  taft lvm2 a-   135.66g 135.66g A
```

O seguinte comando cria um volume espelhado de 100G a partir do grupo de volume **taft**.

```
[root@taft-03 ~]# lvcreate -m 1 -n mirror --nosync -L 100G taft
```

O comando a seguir exibe quais dispositivos são usados pelas pernas do espelho e log do espelho.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                VG        Attr   LSize   Log         Copy%  Devices
  mirror            taft      Mwi-a- 100.00g mirror_mlog 100.00
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      iwi-ao 100.00g
/dev/sdb1(0)
  [mirror_mimage_1] taft      iwi-ao 100.00g
/dev/sdc1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

O comando seguinte estende o tamanho do volume espelhado, usando a política de alocação **cling** para indicar que as pernas do espelho deveriam ser estendidas usando volumes físicos com o mesmo rótulo.

```
[root@taft-03 ~]# lvextend --alloc cling -L +100G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 200.00 GiB
  Logical volume mirror successfully resized
```

O seguinte comando de exibição mostra que as pernas do espelho foram estendidas usando volumes físicos com o mesmo rótulo da perna. Note que volumes físicos com uma tag **C** foram ignorados.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                VG        Attr   LSize   Log         Copy%  Devices
  mirror            taft      Mwi-a- 200.00g mirror_mlog  50.16
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdb1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdg1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdc1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdd1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

4.4.13. Diminuindo Volumes Lógicos

Para reduzir o tamanho de um volume lógico, primeiro desmonte o sistema de arquivo. Então você poderá usar o comando **lvreduce** para diminuir o volume. Depois de diminuir o volume, remonte o sistema de arquivo.

ATENÇÃO

É importante reduzir o tamanho do sistema de arquivo ou o que estiver dentro do volume antes de diminuir o volume, caso contrário você poderá perder dados.

Diminuir um volume lógico libera o grupo de volume para ser alocado a outros volumes lógicos dentro do grupo de volume.

O exemplo seguinte reduz o tamanho do volume lógico **lvol1** no grupo de volume **vg00** por 3 extensões lógicas.

```
lvreduce -l -3 vg00/lvol1
```

4.5. CONTROLANDO ESCANEAMENTOS DE DISPOSITIVOS LVM COM FILTROS

Ao inicializar, o comando **vgscan** é rodado para escanear os dispositivos em bloco no sistema procurando por tags LVM, para determinar quais são volumes físicos, ler os metadados e construir uma

lista de grupo de volumes. Os nomes dos volumes físicos são guardados no arquivo de cache em cada nó no sistema, **/etc/lvm/.cache**. Comandos subsequentes poderão ler esse arquivo para evitar reescaneamento.

Você pode controlar quais dispositivos o LVM escaneia configurando filtros no arquivo de configuração. Os filtros no arquivo **lvm.conf** consistem em uma série de simples expressões regulares que são aplicadas aos nomes dos dispositivos que estão no diretório **/dev** para decidir se aceitam ou rejeitam cada disposito de bloco encontrado.

Os seguintes exemplos mostram o uso de filtros para controlar quais dispositivos o LVM escaneia. Note que alguns destes exemplos não representam necessariamente as melhores práticas, assim como as expressões regulares são combinadas livremente contra o caminho completo. Por exemplo, **a/loop/** é equivalente a **a/.*loop.*/** e não encontraria **/dev/solooperation/lvol1**.

O seguinte filtro adiciona todos os dispositivos descobertos, o qual é comportamento padrão já que não há filtro configurado no arquivo de configuração:

```
filter = [ "a/.*/" ]
```

O seguinte filtro remove o dispositivo de cdrom para evitar atrasos se este não contém nenhuma mídia:

```
filter = [ "r|/dev/cdrom|" ]
```

O seguinte filtro adiciona todo o loop e remove todos os outros dispositivos em bloco:

```
filter = [ "a/loop.*/", "r/.*/" ]
```

O filtro seguinte adiciona todo o loop e remove todos os outros dispositivos em bloco:

```
filter =[ "a|loop.*|", "a|/dev/hd.*|", "r|.*|" ]
```

O filtro seguinte adiciona apenas a partição 8 no primeiro drive IDE e remove todos os outros dispositivos em bloco:

```
filter = [ "a|^/dev/hda8$|", "r/.*/" ]
```

Para mais informações sobre o arquivo **lvm.conf**, veja o Apêndice B, *Os arquivos de Configuração do LVM* e a página man **lvm.conf**(5).

4.6. RELOCAÇÃO ONLINE DE DADOS

Você pode mover dados enquanto o sistema estiver em uso com o comando **pvmove**.

O comando **pvmove** quebra os dados para serem movidos em seções e cria um espelho temporário para mover cada seção. Para maiores informações sobre a operação do comando **pvmove**, veja a página man **pvmove**(8).

O seguinte comando move todo o espaço alocado do volume físico **/dev/sdc1** para outro volume físico livre no grupo de volume:

```
pvmove /dev/sdc1
```

O seguinte comando move apenas as extensões do volume lógico **MyLV**.

```
pvmove -n MyLV /dev/sdc1
```

Já que o comando **pvmove** pode levar um longo tempo para ser executado, você pode rodar o comando em segundo plano para evitar exibições de progresso na tela. O seguinte comando move todas as extensões alocadas ao volume físico **/dev/sdc1** para **/dev/sdf1** no background.

```
pvmove -b /dev/sdc1 /dev/sdf1
```

O seguinte comando reporta o progresso da transferência em porcentagem em intervalos de 5 segundos.

```
pvmove -i5 /dev/sdd1
```

4.7. ATIVANDO VOLUMES LÓGICOS EM NÓS INDIVIDUAIS EM UM CLUSTER

Se você possui o LVM instalado num ambiente de cluster, você pode às vezes precisar ativar volumes lógicos exclusivamente em um único nó.

Para ativar volumes lógicos exclusivamente em um nó, use o comando **lvchange -aey**. Alternativamente, você pode usar o comando **lvchange -aly** para ativar volumes lógicos somente no nó local mas não exclusivamente. Você pode mais tarde ativa-los em nós adicionais simultaneamente.

Você pode também ativar volumes lógicos em nós individuais usando tags LVM, os quais são descritos no Apêndice C, *Tags de Objetos do LVM*. Você pode também especificar a ativação de nós no arquivo de configuração, que é descrito no Apêndice B, *Os arquivos de Configuração do LVM*.

4.8. RELATÓRIO PERSONALIZADO PARA LVM

Você pode produzir relatórios de objetos LVM concisos e personalizáveis com os comandos **pvs**, **lvs** e **vgs**. Os relatórios que estes comandos geram incluem uma linha de resultado para cada objeto. Cada linha contém uma lista ordenada de propriedades dos campos relacionados ao objeto. Existem cinco maneiras para selecionar os objetos a serem reportados: por volume físico, grupo de volume, segmento de volume físico e segmento de volume lógico.

A seguinte seção fornece:

- Um resumo de argumentos dos comandos que você pode usar para controlar o formato dos relatórios gerados.
- Uma lista de campos que você pode selecionar para cada objeto LVM.
- Um resumo de argumentos dos comandos que você pode usar para classificar o relatório gerado.
- Instruções para especificar as unidades para o resultado do relatório.

4.8.1. Controle de Formato

Qualquer dos comandos **pvs**, **lvs** ou **vgs** determinam o conjunto padrão de campos exibidos e a ordem de classificação. Você pode controlar o resultado destes comandos com os seguintes argumentos:

- Você pode alterar quais campos serão exibidos a mais do que somente o padrão usando o argumento **-o**. Por exemplo, o seguinte resultado é a exibição padrão para o comando **pvs** (o qual mostra informações sobre volumes físicos).

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

  O seguinte comando exibe somente o nome e tamanho do volume físico.

```
# pvs -o pv_name,pv_size
  PV         PSize
  /dev/sdb1  17.14G
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
```

- Você pode anexar um campo ao resultado com o sinal de mais (+), que é usado em combinação com o argumento -o.

  O seguinte exemplo exibe o UUID do volume físico além dos campos padrões.

```
# pvs -o +pv_uuid
  PV         VG     Fmt  Attr PSize  PFree  PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-
M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-
01S9-X08M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-UqkCS
```

- Adicionando o argumento **-v** ao comando, incluirá campos extras. Por exemplo, o comando **pvs -v** exibirá os campos **DevSize** e **PV UUID** alem dos campos padrões.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-
ughJ-D9eB-M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-
kuEn-IdwM-01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G  17.14G yvfvZK-Cf31-
j75k-dECm-0RZ3-0dGW-tUqkCS
```

- O argumento **--noheadings** suprime a linha título. Isto pode ser útil na escrita de scripts.

  O exemplo seguinte usa o argumento **--noheadings** em combinação com o argumento **pv_name**, que vai gerar uma lista de todos os volumes físicos.

```
# pvs --noheadings -o pv_name
  /dev/sdb1
  /dev/sdc1
  /dev/sdd1
```

- O argumento **--separator *separador*** usa sinais para separar cada campo.

  O seguinte exemplo separa os resultados de campo padrão do comando **pvs** com o sinal igual (=).

```
# pvs --separator =
  PV=VG=Fmt=Attr=PSize=PFree
  /dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
  /dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
  /dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

  Para manter os campos alinhados quando estiver usando o argumento **separator**, use este argumento em conjunto com o argumento **--aligned**.

```
# pvs --separator = --aligned
  PV         =VG    =Fmt =Attr=PSize =PFree
  /dev/sdb1 =new_vg=lvm2=a-  =17.14G=17.14G
  /dev/sdc1 =new_vg=lvm2=a-  =17.14G=17.09G
  /dev/sdd1 =new_vg=lvm2=a-  =17.14G=17.14G
```

Você pode usar o argumento **-P** dos comandos **lvs** ou **vgs** para exibir informações sobre um volume com falha que caso contrário não apareceria no resultado. Para informações sobre os resultados fornecidos por este argumento, veja a Seção 6.2, “A exibição de informações em Dispositivos Falhos”.

Para uma lista completa de argumentos de exibição, veja as páginas man **pvs**(8), **vgs**(8) and **lvs**(8).

Os campos do grupo de volume podem ser misturados tanto com campos de volume físico (e segmentos de volume físico) ou campos de volume lógico (e segmentos de volume lógico), mas campos de volumes físico e lógicos não podem ser misturados. Por exemplo, o comando seguinte exibirá uma linha por resultado para cada volume físico.

```
# vgs -o +pv_name
  VG     #PV #LV #SN Attr   VSize  VFree  PV
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdc1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdd1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdb1
```

4.8.2. Seleção de Objeto

Esta seção fornece uma série de tabelas que listam as informações que você pode exibir sobre os objetos LVM com os comandos **pvs**, **vgs** e **lvs**.

Por conveniência, um prefixo do nome do campo pode ser descartado se coincidir com o padrão para o comando. Por exemplo, com o comando **pvs**, **name** significa **pv_name**, mas com o comando **vgs**, **name** é interpretado como **vg_name**.

Executar o seguinte comando é o equivalente a executar **pvs -o pv_free**.

```
# pvs -o +free
  PFree
  17.14G
```

```
  17.09G
  17.14G
```

4.8.2.1. O comando pvs

A Tabela 4.1, “Campos de Exibição do pvs” lista os argumentos de exibição do comando **pvs** , junto com o nome do campo como aparece na exibição do cabeçalho e uma descrição do campo.

Tabela 4.1. Campos de Exibição do pvs

Argumento	Cabeçalho	Descrição
dev_size	DevSize	O tamanho do dispositivo subjacente ao qual o volume físico foi criado.
pe_start	1st PE	Offset para o início da primeira extensão física do dispositivo subjacente
pv_attr	Attr	Estado do volume físico: (a)locável ou e(x)portado.
pv_fmt	Fmt	O formato dos metadados do volume físico (**lvm2** ou **lvm1**)
pv_free	PFree	O espaço livre restante no volume físico
pv_name	PV	O nome do volume físico
pv_pe_alloc_count	Alloc	Número de extensões físicas usadas
pv_pe_count	PE	Número de extensões físicas
pvseg_size	SSize	O tamanho de segmento do volume físico
pvseg_start	Início	O início da extensão física do segmento de volume físico
pv_size	PSize	O tamanho do volume físico
pv_tags	PV Tags	Rótulos LVM anexados ao volume físico
pv_used	Usado	A quantidade de espaço atualmente usada no volume físico
pv_uuid	PV UUID	O UUID do volume físico

O comando **pvs** exibe os seguintes comando por padrão: **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**. A exibição é classificada por **pv_name**.

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
```

```
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G
```

Usando o argumento **-v** com o comando **pvs** se adiciona os seguintes campos à exibição padrão: **dev_size**, **pv_uuid**.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-
M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-
01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-tUqkCS
```

Você pode usar o argumento **--segments** do comando **pvs** para exibir informações sobre cada segmento de volume físico. Um segmento é um grupo de extensões. Uma visão do segmento pode ser útil se você quer ver se seu volume lógico está fragmentado.

O comando **pvs --segments** exibe os seguintes campos por padrão: **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**, **pvseg_start**, **pvseg_size**. A exibição é classificada por **pv_name** e **pvseg_size** dentro do volume físico.

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  Start SSize
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M     0  1172
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1172    16
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1188     1
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G     0    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    26    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    50    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    76    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   100    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   126    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   150    22
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   172  4217
  /dev/sdb1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdc1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdd1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sde1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdf1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdg1  vg         lvm2 a-   17.14G 17.14G     0  4389
```

Você pode usar o comando **pvs -a** para ver dispositivos detectados pelo LVM que não foram inicializados como volumes físicos LVM.

```
# pvs -a
  PV                            VG     Fmt  Attr PSize  PFree
  /dev/VolGroup00/LogVol01                   --       0      0
  /dev/new_vg/lvol0                          --       0      0
  /dev/ram                                   --       0      0
  /dev/ram0                                  --       0      0
  /dev/ram2                                  --       0      0
  /dev/ram3                                  --       0      0
```

```
  /dev/ram4                                --       0      0
  /dev/ram5                                --       0      0
  /dev/ram6                                --       0      0
  /dev/root                                --       0      0
  /dev/sda                                 --       0      0
  /dev/sdb                                 --       0      0
  /dev/sdb1                  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc                                 --       0      0
  /dev/sdc1                  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd                                 --       0      0
  /dev/sdd1                  new_vg lvm2 a-   17.14G 17.14G
```

4.8.2.2. O Comando vgs

A Tabela 4.2, “Campos de Exibição vgs” lista os argumentos de exibição do comando **vgs**, junto com o nome do campo como aparece no cabeçalho e uma descrição do campo.

Tabela 4.2. Campos de Exibição vgs

Argumento	Cabeçalho	Descrição
lv_count	#LV	O número de volumes lógicos que o grupo de volume contém.
max_lv	MaxLV	O número máximo de volumes lógicos permitidos no grupo de volume (0 se ilimitados)
max_pv	MaxPV	O número máximo de volumes físicos permitidos no grupo de volume (0 se ilimitados)
pv_count	#PV	O número de volumes físicos que definem o grupo de volume
snap_count	#SN	O número de snapshots que o grupo de volume contém
vg_attr	Attr	Estado do grupo de volume:(w)gravável, (r)somente leitura, (z)redimensionável, (x)exportado, (p)parcial e (c)em cluster.
vg_extent_count	#Ext	O número de extensões físicas no grupo de volume
vg_extent_size	Ext	O tamanho das extensões físicas no grupo de volume
vg_fmt	Fmt	O formato de metadados no grupo de volume (**lvm2** ou **lvm1**)
vg_free	VFree	Tamanho do espaço livre restante no grupo de volume
vg_free_count	Livre	Número de extensões físicas livres no grupo de volume
vg_name	VG	O nome do grupo de volume

Argumento	Cabeçalho	Descrição
vg_seqno	Seq	Número representando a revisão do grupo de volume
vg_size	VSize	O tamanho do grupo de volume
vg_sysid	SYS ID	LVM1 System ID
vg_tags	VG Tags	Tags LVM anexadas ao grupo de volume
vg_uuid	VG UUID	O UUID do grupo de volume

O comando **vgs** exibe os seguintes campos por padrão: **vg_name**, **pv_count**, **lv_count**, **snap_count**, **vg_attr**, **vg_size**, **vg_free**. A exibição é classificada por **vg_name**.

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  new_vg   3   1   1 wz--n- 51.42G 51.36G
```

Usando o argumento **-v** com o comando **vgs** se adiciona os seguintes campos à exibição padrão: **vg_extent_size**, **vg_uuid**.

```
# vgs -v
    Finding all volume groups
    Finding volume group "new_vg"
  VG     Attr   Ext   #PV #LV #SN VSize  VFree  VG UUID
  new_vg wz--n- 4.00M   3   1   1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-
nlwO-wwqd-fD5D32
```

4.8.2.3. O Comando lvs

A Tabela 4.3, “Campos de Exibição lvs” lista os argumentos de exibição do comando **lvs**, junto com o nome do campo como aparece no cabeçalho e uma descrição do campo.

Tabela 4.3. Campos de Exibição lvs

Argumento	Cabeçalho	Descrição
chunksize **chunk_size**	Chunk	Tamanho da unidade em um volume snapshot
copy_percent	Copy%	A porcentagem de sincronização de um volume lógico espelhado, também usado quando extensões físicas estao sendo movidas com o comando **pv_move**

Argumento	Cabeçalho	Descrição
devices	Devices	Os dispositivos subjacentes que compõem o volume lógico: os volumes físicos, volumes lógicos e extensões físicas e extensões lógicas de início
lv_attr	Attr	O estado do volume lógico. Os bits do volume lógico são os seguintes: Bit 1: Tipo de volume:(m)espelhado, (M)espelhado sem sync inicial, (o)rigem,(p)pvmove, (s)napshot, (S)napshot inválido, (v)irtual Bit 2: Permissões: (w)gravável, (r)leitura somente Bit 3: Política de alocação: (c)ontíguo, (n)ormal, (a)qualquer lugar, (i)herdado. As letras são em maiúsculo se o volume está atualmente bloqueado contra alterações de alocação, por exemplo enquanto executando o comando **pvmove**. Bit 4: ajustado (m)enor Bit 5: Estado (a)tivo, (s)uspenso, (l)snapshot inválido, (S)napshot suspenso, (d)ispositivo mapeado present sem tabelas, (i) dispositivo mapeado present sem tabelas inativas Bit 6: (o)dispositivo aberto
lv_kernel_major	KMaj	Número maior atual do dispositivo do volume lógico (-1 se inativo)
lv_kernel_minor	KMIN	Número menor atual do dispositivo do volume lógico (-1 se inativo)
lv_major	Maj	O número maior do dispositivo persistente do volume lógico (-1 if not specified)
lv_minor	Min	O número menor do dispositivo persistente do volume lógico (-1 if not specified)
lv_name	LV	O nome do volume lógico
lv_size	LSize	O tamanho do volume lógico
lv_tags	LV Tags	Rótulos LVM anexados ao volume lógico
lv_uuid	LV UUID	O UUID do volume lógico

Argumento	Cabeçalho	Descrição
mirror_log	Log	Dispositivo onde o log de espelho reside
modules	Modules	Correspondente alvo do mapeador de dispositivo kernel necessário para usar este volume lógico
move_pv	Move	Volume físico de origem de um volume lógico temporário criado com o comando **pvmove**
origin	Origin	O dispositivo origem de um volume snapshot
regionsize **region_size**	Region	O tamanho da unidade de um volume lógico espelhado
seg_count	#Seg	O número de segmentos no volume lógico
seg_size	SSize	O tamanho dos segmentos no volume lógico
seg_start	Início	Offset do segmento no volume lógico
seg_tags	Seg Tags	Tags LVM anexadas aos segmentos do volume lógico
segtype	Type	O tipo de segmento de um volume lógico (por exemplo: espelho, distribuído, linear)
snap_percent	Snap%	Porcentagem atual de um volume snapshot que está em uso
stripes	#Str	Número de distribuições ou espelhos em um volume lógico
stripesize **stripe_size**	Stripe	Tamanho da unidade de uma distribuição em um volume lógico distribuído

O comando **lvs** exibe os seguintes comandos por padrão: **lv_name**, **vg_name**, **lv_attr**, **lv_size**, **origin**, **snap_percent**, **move_pv**, **mirror_log**, **copy_percent**. A exibição padrão é classificada por **vg_name** e **lv_name** dentro do grupo de volume.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```

Usando o argumento **-v** com o comando **lvs** adiciona os seguintes campos à exibição padrão: **seg_count**, **lv_major**, **lv_minor**, **lv_kernel_major**, **lv_kernel_minor**, **lv_uuid**.

```
# lvs -v
    Finding all logical volumes
  LV         VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%
Move Copy%  Log LV UUID
  lvol0      new_vg    1 owi-a- 52.00M  -1  -1 253  3
LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20
1ye1OU-1cIu-o79k-20h2-ZGF0-qCJm-CfbsIx
```

Você pode usar o argumento **--segments** do comando **lvs** para exibir informações com as colunas padrões que enfatizam as informações do segmento. Quando você usa o argumento **segments**, o prefixo **seg** é opcional. O comando **lvs --segments** exibe os seguintes campos por padrão: **lv_name**, **vg_name**, **lv_attr**, **stripes**, **segtype**, **seg_size**. A exibição padrão é classificada por **vg_name**, **lv_name** dentro do grupo de volume e **seg_start** dentro do volume lógico. Se os volumes lógicos foram fragmentados, o resultado deste comando mostraria isso.

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    1 linear 512.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear  88.00M
```

Usando o argumento **-v** com o comando **lvs --segments** adiciona os seguintes campos à exibição padrão: **seg_start**, **stripesize**, **chunksize**.

```
# lvs -v --segments
    Finding all logical volumes
  LV         VG     Attr   Start SSize  #Str Type   Stripe Chunk
  lvol0      new_vg owi-a-     0  52.00M    1 linear     0     0
  newvgsnap1 new_vg swi-a-     0   8.00M    1 linear     0  8.00K
```

O seguinte comando mostra o resultado padrão do comando **lvs** em um sistema com um volume lógico configurado, seguido pelo resultado padrão do comando **lvs** com argumento **segments** especificado.

```
# lvs
  LV    VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 new_vg -wi-a- 52.00M
# lvs --segments
  LV    VG     Attr   #Str Type   SSize
  lvol0 new_vg -wi-a-    1 linear 52.00M
```

4.8.3. Ordenando Relatórios LVM

Normalmente o resultado completo do comandos **lvs**, **vgs** ou **pvs**, tem de ser gerado e guardado internamente antes que possa ser ordenado e as colunas alinhadas corretamente. Você pode especificar o argumento **--unbuffered** para exibir um resultado não ordenado tão logo ele é gerado.

Para especificar uma lista classificada de colunas alternativas para serem classificadas , use o argumento **-O** em qualquer um dos comandos de relatório. Não é necessário incluir estes campos dentro do próprio resultado.

O exemplo seguinte mostra o resultado do comando **pvs** que exibe o nome do volume físico, tamanho e espaço livre.

```
# pvs -o pv_name,pv_size,pv_free
  PV         PSize  PFree
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
```

O exemplo seguinte exibe o mesmo resultado, ordenado pelo campo espaço livre.

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
  PV         PSize  PFree
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
```

O exemplo seguinte mostra que você não precisa exibir o campo no qual está ordenando.

```
# pvs -o pv_name,pv_size -O pv_free
  PV         PSize
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
  /dev/sdb1  17.14G
```

Para exibir uma ordenação reversa, coloque depois do argumento **-O** o argumento **-** no campo desejado.

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
  PV         PSize  PFree
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
```

4.8.4. Especificando Unidades

Para especificar a unidade para o relatório do LVM exibir, use o argumento **--units** do comando de relatório. Você pode especificar (b)ytes, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (e)xabytes, (p)etabytes ou (h)uman-readable (legível). A exibição padrão é human-readable (legível). Você pode substituir o padrão pela configuração do parâmetro **units** na seção **global** do arquivo **lvm.conf**.

O exemplo seguinte especifica o resultado do comando **pvs** em megabytes ao invés do padrão gigabytes.

```
# pvs --units m
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sda1         lvm2 --   17555.40M 17555.40M
  /dev/sdb1  new_vg lvm2 a-   17552.00M 17552.00M
  /dev/sdc1  new_vg lvm2 a-   17552.00M 17500.00M
  /dev/sdd1  new_vg lvm2 a-   17552.00M 17552.00M
```

Por padrão, unidades são exibidas em potências de 2 (múltiplos de 1024). Você pode especificar que as unidades sejam exibidas em mútliplos de 1000, digitando em maiúsculo as especificações de unidades (B, K, M, G, T, H).

O seguinte comando exibe o resultado em múltiplos de 1024, o comportamento padrão.

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

O seguinte comando exibe o resultado em múltiplos de 1000.

```
#  pvs --units G
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   18.40G 18.40G
  /dev/sdc1  new_vg lvm2 a-   18.40G 18.35G
  /dev/sdd1  new_vg lvm2 a-   18.40G 18.40G
```

Você pode também especificar (s)etores (definidos em 512 bytes) ou unidades personalizadas.

O seguinte exemplo exibe o resultado do comando **pvs** em número de setores.

```
# pvs --units s
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sdb1  new_vg lvm2 a-   35946496S 35946496S
  /dev/sdc1  new_vg lvm2 a-   35946496S 35840000S
  /dev/sdd1  new_vg lvm2 a-   35946496S 35946496S
```

O exemplo seguinte exibe o resultado do comando **pvs** em unidades de 4 megabytes.

```
# pvs --units 4m
  PV         VG     Fmt  Attr PSize    PFree
  /dev/sdb1  new_vg lvm2 a-   4388.00U 4388.00U
  /dev/sdc1  new_vg lvm2 a-   4388.00U 4375.00U
  /dev/sdd1  new_vg lvm2 a-   4388.00U 4388.00U
```

CAPÍTULO 5. EXEMPLOS DE CONFIGURAÇÃO DO LVM

Este capítulo fornece alguns exemplos de configuração do LVM.

5.1. CRIANDO UM VOLUME LÓGICO LVM EM TRÊS DISCOS

Este exemplo cria um volume lógico LVM chamado **new_logical_volume** que consiste dos discos em **/dev/sda1**, **/dev/sdb1** e **/dev/sdc1**.

5.1.1. Criando Volumes Físicos

Para usar discos em um grupo de volume, você os rotula como volumes físicos LVM.

ATENÇÃO

Este comando destrói qualquer dado em **/dev/sda1**, **/dev/sdb1** e **/dev/sdc1**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

5.1.2. Criando o Grupo de Volume

O comando a seguir cria um grupo de volume **new_vol_group**.

```
[root@tng3-1 ~]# vgcreate new_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "new_vol_group" successfully created
```

Você pode usar o comando **vgs** para exibir os atributos do novo grupo de volume.

```
[root@tng3-1 ~]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  new_vol_group   3   0   0 wz--n- 51.45G 51.45G
```

5.1.3. Criando o Volume Lógico

O comando a seguir cria um volume lógico **new_logical_volume** a partir do grupo de volume **new_vol_group**. Este exemplo cria um volume lógico que usa 2GB do grupo de volume.

```
[root@tng3-1 ~]# lvcreate -L2G -n new_logical_volume new_vol_group
  Logical volume "new_logical_volume" created
```

5.1.4. Criando o Sistema de Arquivo

O comando a seguir cria um sistema de arquivo GFS2 no volume lógico

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/new_vol_group/new_logical_volume
This will destroy any data on /dev/new_vol_group/new_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/new_vol_group/new_logical_volume
Blocksize:                 4096
Filesystem Size:           491460
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Os comandos a seguir montam o volume lógico e reportam o uso do espaço de disco do sistema de arquivo.

```
[root@tng3-1 ~]# mount /dev/new_vol_group/new_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/new_vol_group/new_logical_volume
                       1965840        20   1965820   1% /mnt
```

5.2. CRIANDO UM VOLUME LÓGICO DISTRIBUÍDO

Este exemplo cria um volume lógico distribuído LVM chamado **`striped_logical_volume`** que distribui os dados entre os discos **`/dev/sda1`**, **`/dev/sdb1`** e **`/dev/sdc1`**.

5.2.1. Criando Volumes Físicos

Rotule os discos que você irá usar nos grupos de volume como volumes físicos LVM.

ATENÇÃO

Este comando destrói qualquer dado em **`/dev/sda1`**, **`/dev/sdb1`** e **`/dev/sdc1`**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

5.2.2. Criando o Grupo de Volume

Os seguintes comandos criam o grupo de volume **volgroup01**.

```
[root@tng3-1 ~]# vgcreate volgroup01 /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "volgroup01" successfully created
```

Você pode usar o comando **vgs** para exibir os atributos do novo grupo de volume.

```
[root@tng3-1 ~]# vgs
  VG                #PV #LV #SN Attr   VSize  VFree
  volgroup01          3   0   0 wz--n- 51.45G 51.45G
```

5.2.3. Criando o Volume Lógico

O comando a seguir cria um volume lógico distribuído **striped_logical_volume** a partir do grupo de volume **volgroup01**. Este exemplo cria um volume lógico que possui 2 gigabytes em tamanho, com três distribuições e o tamanho da distribuição de 4 kilobytes.

```
[root@tng3-1 ~]# lvcreate -i3 -I4 -L2G -nstriped_logical_volume volgroup01
  Rounding size (512 extents) up to stripe boundary size (513 extents)
  Logical volume "striped_logical_volume" created
```

5.2.4. Criando o Sistema de Arquivo

O comando a seguir cria um sistema de arquivo GFS2 no volume lógico

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/volgroup01/striped_logical_volume
This will destroy any data on /dev/volgroup01/striped_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/volgroup01/striped_logical_volume
Blocksize:                 4096
Filesystem Size:           492484
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Os comandos a seguir montam o volume lógico e reportam o uso do espaço de disco do sistema de arquivo.

```
[root@tng3-1 ~]# mount /dev/volgroup01/striped_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      13902624   1656776  11528232  13% /
/dev/hda1               101086     10787     85080  12% /boot
```

```
tmpfs                    127880         0    127880   0% /dev/shm
/dev/volgroup01/striped_logical_volume
                        1969936        20   1969916   1% /mnt
```

5.3. DIVIDINDO UM GRUPO DE VOLUME

Neste exemplo, um grupo de volume existente consiste de três volumes físicos. Caso haja espaço livre suficiente nos volumes físicos, um novo grupo de volume pode ser criado sem adicionar novos discos.

Na configuração inicial, o volume lógico **mylv** é retirado do grupo de volume **myvol**, que por sua vez, consiste de três volumes físicos **/dev/sda1**, **/dev/sdb1**,e **/dev/sdc1**.

Após o término deste procedimento, o grupo de volume **myvg** consistirá de **/dev/sda1** e **/dev/sdb1**. Um segundo grupo de volume, **yourvg**, conterá o **/dev/sdc1**.

5.3.1. Determinando o Espaço Livre

Você pode usar o comando **pvscan** para determinar o quanto de espaço livre existe no momento disponível no grupo de volume.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1  VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1  VG myvg   lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1  VG myvg   lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

5.3.2. Movendo os Dados

Você pode mover todas as extensões físicas do **/dev/sdc1** para **/dev/sdb1** com o comando **pvmove**. O comando **pvmove** pode levar um longo tempo para ser executado.

```
[root@tng3-1 ~]# pvmove /dev/sdc1 /dev/sdb1
  /dev/sdc1: Moved: 14.7%
  /dev/sdc1: Moved: 30.3%
  /dev/sdc1: Moved: 45.7%
  /dev/sdc1: Moved: 61.0%
  /dev/sdc1: Moved: 76.6%
  /dev/sdc1: Moved: 92.2%
  /dev/sdc1: Moved: 100.0%
```

Após mover os dados, você verá que todo o espaço no **/dev/sdc1** está livre.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 10.80 GB free]
  PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 17.15 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

5.3.3. Dividindo o Grupo de Volume

Para criar o novo grupo de volume **yourvg**, use o **vgsplit** para dividir o grupo de volume **myvg**.

Antes que você possa dividir o grupo de volume, o volume lógico deve estar inativo. Se o sistema for montado, você precisa desmontar o sistema de arquivo antes de desativar os volumes lógicos.

Você pode desativar os volumes lógicos com o comando **lvchange** ou o comando **vgchange**. O comando a seguir desativa o volume lógico **mylv** e depois divide o grupo de volume **yourvg** a partir do grupo de volume **myvg**, movendo o volume físico **/dev/sdc1** para o novo grupo de volume **yourvg**.

```
[root@tng3-1 ~]# lvchange -a n /dev/myvg/mylv
[root@tng3-1 ~]# vgsplit myvg yourvg /dev/sdc1
  Volume group "yourvg" successfully split from "myvg"
```

Você pode usar o comando **vgs** para ver os atributos dos dois grupos de volume.

```
[root@tng3-1 ~]# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  myvg     2   1   0 wz--n- 34.30G 10.80G
  yourvg   1   0   0 wz--n- 17.15G 17.15G
```

5.3.4. Criando o Novo Volume Lógico

Após a criação do novo grupo de volume, você pode criar um novo volume lógico **yourlv**.

```
[root@tng3-1 ~]# lvcreate -L5G -n yourlv yourvg
  Logical volume "yourlv" created
```

5.3.5. Criando um Sistema de Arquivo e Montando o Novo Volume Lógico

Você pode criar um sistema de arquivo no volume lógico novo e montá-lo.

```
[root@tng3-1 ~]#  mkfs.gfs2 -plock_nolock -j 1 /dev/yourvg/yourlv
This will destroy any data on /dev/yourvg/yourlv.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/yourvg/yourlv
Blocksize:                 4096
Filesystem Size:           1277816
Journals:                  1
Resource Groups:           20
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done

[root@tng3-1 ~]# mount /dev/yourvg/yourlv /mnt
```

5.3.6. Ativando e Montando o Volume Lógico Original

Como você já havia desativado o volume lógico **mylv**, você precisa ativá-lo novamente antes que possa montá-lo.

```
root@tng3-1 ~]# lvchange -a y mylv

[root@tng3-1 ~]# mount /dev/myvg/mylv /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/yourvg/yourlv    24507776        32  24507744   1% /mnt
/dev/myvg/mylv        24507776        32  24507744   1% /mnt
```

5.4. REMOVENDO UM DISCO DO VOLUME LÓGICO

Este exemplo demonstra como você pode remover um disco de um volume lógico existente tanto para substituir o disco ou para usar o disco como parte de um volume diferente. Para remover um disco, você precisa mover primeiro as extensões no volume físico LVM para um disco diferente ou um conjunto de discos.

5.4.1. Movendo as Extensões para os Volumes Físicos Existentes

Neste exemplo, o volume lógico é distribuído entre os quatro volumes físicos no grupo de volume **myvg**.

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Queremos mover as extensões de todos os **/dev/sdb1**, para que possamos removê-lo do grupo de volume.

Caso existam extensões livres suficientes em outros volumes físicos no grupo de volume, você pode executar o comando **pvmove** no dispositivo que você deseja remover sem nenhuma outra opção e as extensões serão distribuídas à outros dispositivos.

```
[root@tng3-1 ~]# pvmove /dev/sdb1
  /dev/sdb1: Moved: 2.0%
 ...
  /dev/sdb1: Moved: 79.2%
 ...
  /dev/sdb1: Moved: 100.0%
```

Após o comando **pvmove** concluir sua execução, a distribuição da extensão seria como esta:

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Use o comando **vgreduce** para remover o volume físico **/dev/sdb1** do grupo de volume.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
```

```
[root@tng3-1 ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G
  /dev/sdb1       lvm2 --   17.15G 17.15G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G
```

O disco pode agora ser fisicamente removido ou alocado à outros usuários.

5.4.2. Movendo Extensões para um Novo Disco

Neste exemplo, o volume lógico é distribuído entre os três volumes físicos no grupo de volume **myvg** como se segue:

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

Nós queremos mover as extensões do **/dev/sdb1** para um novo dispositivo, **/dev/sdd1**.

5.4.2.1. Criando o Novo Volume Físico

Crie um novo volume físico a partir do **/dev/sdd1**.

```
[root@tng3-1 ~]# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

5.4.2.2. Adicionando o Novo Volume Físico ao Grupo de Volume.

Adicione o **/dev/sdd1** ao grupo de volume existente **myvg**.

```
[root@tng3-1 ~]# vgextend myvg /dev/sdd1
  Volume group "myvg" successfully extended
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 17.15G     0
```

5.4.2.3. Movendo os Dados

Use o comando **pvmove** para mover dados de **/dev/sdb1** para **/dev/sdd1**.

```
[root@tng3-1 ~]# pvmove /dev/sdb1 /dev/sdd1
  /dev/sdb1: Moved: 10.0%
...
  /dev/sdb1: Moved: 79.7%
...
  /dev/sdb1: Moved: 100.0%
```

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

5.4.2.4. Removendo o Volume Fisico Antigo do Grupo de Volume

Após ter movido os dados do **/dev/sdb1**, você poderá removê-lo do grupo de volume.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
```

Você pode agora realocar o disco para outro grupo de volume ou remover o disco do sistema.

5.5. CRIANDO UM VOLUME LÓGICO LVM ESPELHADO EM UM CLUSTER

A criação de um volume lógico LVM espelhado em um cluster requer os mesmos comandos e procedimentos usados ao criar um volume lógico LVM espelhado em um nó único. No entanto, para criar um volume LVM espelhado em um cluster, este e sua infraestrutura de espelho devem estar em execução, o cluster deve estar em quorum e o tipo de bloqueio no arquivo **lvm.conf** deve estar definido corretamente para permitir o bloqueio de cluster, tanto diretamente quanto por meio do comando **lvmconf** como descrito na Seção 3.1, “Criando Volumes LVM em um Cluster”.

O procedimento a seguir cria um volume LVM espelhado em um cluster. Primeiro o procedimento verifica se os serviços em cluster estão instalados e em execução, então o procedimento cria um volume espelhado.

1. Para criar um volume lógico espelhado compartilhado por todos os nós em um cluster, o tipo de bloqueio deve ser definido corretamente no arquivo **lvm.conf** em todos os nós do cluster. Por padrão, o tipo de bloqueio é definido para local. Para mudar isto, execute o seguinte comando em cada nó do cluster para habilitar o bloqueio de cluster:

   ```
   # /sbin/lvmconf --enable-cluster
   ```

2. Para criar um volume lógico, a infraestrutura deve estar ativa e em execução em todos os nós do cluster. O exemplo a seguir verifica se o daemon **clvmd** está em execução no nó do qual ele foi emitido:

   ```
   [root@doc-07 ~]# ps auxw | grep clvmd
   root     17642  0.0  0.1 32164 1072 ?        Ssl  Apr06   0:00
   clvmd -T20 -t 90
   ```

   O comando a seguir demonstra a visão local do estado do cluster:

   ```
   [root@example-01 ~]# cman_tool services
   fence domain
   member count  3
   victim count  0
   victim now    0
   master nodeid 2
   ```

```
wait state    none
members       1 2 3

dlm lockspaces
name          clvmd
id            0x4104eefa
flags         0x00000000
change        member 3 joined 1 remove 0 failed 0 seq 1,1
members       1 2 3
```

3. Certifique-se de que o pacote **cmirror** está instalado.

4. Inicie o serviço **cmirrord**.

```
[root@hexample-01 ~]# service cmirrord start
Starting cmirrord:                                         [  OK
]
```

5. Crie o espelho. O primeiro passo é criar os volumes físicos. Os comandos a seguir criam três volumes físicos. Dois destes volumes físicos serão usados para pernas de espelho e o terceiro conterá um log do espelho.

```
[root@doc-07 ~]# pvcreate /dev/xvdb1
  Physical volume "/dev/xvdb1" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdb2
  Physical volume "/dev/xvdb2" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdc1
  Physical volume "/dev/xvdc1" successfully created
```

6. Crie um grupo de volume. Este exemplo cria um grupo de volume **vg001** que consiste em três volumes físicos que foram criados nos passos anteriores.

```
[root@doc-07 ~]# vgcreate vg001 /dev/xvdb1 /dev/xvdb2 /dev/xvdc1
  Clustered volume group "vg001" successfully created
```

   Note que o resultado do comando **vgcreate** indica que o grupo de volume está em cluster. Você pode verificar se o grupo de volume está em cluster com o comando **vgs**, o qual irá exibir os recursos do grupo de volume. Se um grupo de volume estiver em cluster, este será exibido com um atributo c.

```
[root@doc-07 ~]# vgs vg001
  VG       #PV #LV #SN Attr   VSize  VFree
  vg001      3   0   0 wz--nc 68.97G 68.97G
```

7. Crie um volume lógico espelhado. Este exemplo cria o volume lógico **mirrorlv** de um grupo de volume **vg001**. Este volume possui uma perna de espelho. Este exemplo especifica quais extensões de volume físico serão usadas para o volume lógico.

```
[root@doc-07 ~]# lvcreate -l 1000 -m1 vg001 -n mirrorlv
/dev/xvdb1:1-1000 /dev/xvdb2:1-1000 /dev/xvdc1:0
  Logical volume "mirrorlv" created
```

Você pode usar o comando **lvs** para exibir o progresso da criação do espelho. O exemplo a seguir exibe se o espelho está sincronizado em 47% e depois sincronizado em 91% e depois 100% quando o espelho for concluído.

```
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
47.00
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
91.00
[root@doc-07 ~]#  lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
100.00
```

A conclusão do espelho é anotada no log do sistema:

```
May 10 14:52:52 doc-07 [19402]: Monitoring mirror device vg001-
mirrorlv for events
May 10 14:55:00 doc-07 lvm[19402]: vg001-mirrorlv is now in-sync
```

8. Você pode usar **lvs** com as opções **-o +devices** para exibir a configuração do espelho, incluindo os dispositivos que criam as pernas do espelho. Você pode ver que o volume lógico neste exemplo é composto por duas imagens lineares e um log.

```
[root@doc-07 ~]# lvs -a -o +devices
  LV                 VG         Attr   LSize  Origin Snap%  Move
Log           Copy%  Convert Devices
  mirrorlv           vg001      mwi-a-  3.91G
mirrorlv_mlog 100.00
mirrorlv_mimage_0(0),mirrorlv_mimage_1(0)
  [mirrorlv_mimage_0] vg001      iwi-ao  3.91G
/dev/xvdb1(1)
  [mirrorlv_mimage_1] vg001      iwi-ao  3.91G
/dev/xvdb2(1)
  [mirrorlv_mlog]     vg001      lwi-ao  4.00M
/dev/xvdc1(0)
```

Você pode usar a opção **seg_pe_ranges** do **lvs** para exibir o layout dos dados. Você pode usar esta opção para verificar se seu layout está redundante adequadamente. O resultado deste comando exibe a classe PE no mesmo formato que os comandos **lvcreate** e **lvresize** recebem como entrada.

```
[root@doc-07 ~]# lvs -a -o +seg_pe_ranges --segments
  PE Ranges
  mirrorlv_mimage_0:0-999 mirrorlv_mimage_1:0-999
  /dev/xvdb1:1-1000
  /dev/xvdb2:1-1000
  /dev/xvdc1:0-0
```

NOTA

Para informações sobre recuperação da falha de uma das pernas de um volume espelhado, veja a Seção 6.3, “Recuperação de Falha do Espelho LVM.”.

CAPÍTULO 6. SOLUÇÃO DE PROBLEMAS LVM

Este capítulo fornece instruções para solução de problemas para uma variedade de casos no LVM.

6.1. DIAGNÓSTICOS PARA SOLUÇÃO DE PROBLEMAS

Se um comando não estiver funcionando como esperado, você pode reunir os diagnósticos das seguintes formas:

- Use os argumentos **-v**, **-vv**, **-vvv** ou **-vvvv** de qualquer comando para aumentar os níveis de informação dos resultados.
- Se o problema estiver relacionado com a ativação do volume lógico, defina 'ativação= 1' na seção 'log' do arquivo de configuração e rode o comando com o argumento **-vvvv**. Depois que acabar de examinar este resultado, certifique-se de resetar este parâmetro para 0, para evitar problemas possíveis com o bloqueio da máquina durante situações de baixa memória.
- Execute o comando **lvmdump** , que fornece despejo de informações para propósitos de diagnósticos. Para informações, veja a página man **lvmdump**(8).
- Execute o comando **lvs -v**, **pvs -a** our **dmsetu info -c** para informações de sistema adicionais.
- Examine os últimos backups de metadados no arquivo **/etc/lvm/backup** e versões arquivadas no arquivo **/etc/lvm/archive**.
- Verifique informações sobre configurações atuais executando o comando **lvm dumpconfig**.
- Verifique o arquivo **.cache** no diretório **/etc/lvm** para ver um registro de quais dispositivos possuem volumes físicos.

6.2. A EXIBIÇÃO DE INFORMAÇÕES EM DISPOSITIVOS FALHOS

Você pode usar o argumento **-P** do comando **lvs** ou **vgs** para exibir informações sobre um volume defeituoso que de outra maneira não apareceria no resultado. Este argumento permite algumas operações, embora os metadados não sejam completamente consistentes internamente. Por exemplo, se um dos dispositivos que criam o grupo de volume **vg** falhar, o comando **vgs** pode exibir o seguinte resultado:

```
[root@link-07 tmp]# vgs -o +devices
  Volume group "vg" not found
```

Se não for especificado o argumento **-P** do comando **vgs**, o grupo de volume estará ainda inutilizável mas você pode ver mais informações sobre o dispositivo com defeito.

```
[root@link-07 tmp]# vgs -P -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  VG   #PV #LV #SN Attr   VSize VFree Devices
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(0)
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(5120),/dev/sda1(0)
```

Neste exemplo, o dispositivo com falha fazia com que tanto o volume lógico linear e o volume lógico distribuido dentro do grupo de volume falhassem. O comando **lvs** sem o argumento **-P** demonstra o seguinte resultado.

```
[root@link-07 tmp]# lvs -a -o +devices
  Volume group "vg" not found
```

Ao usar o argumento **-P**, exibe-se os volumes lógicos que falharam.

```
[root@link-07 tmp]# lvs -P -a -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  LV     VG   Attr   LSize  Origin Snap%  Move Log Copy%  Devices
  linear vg   -wi-a- 20.00G                                unknown
device(0)
  stripe vg   -wi-a- 20.00G                                unknown
device(5120),/dev/sda1(0)
```

Os seguintes exemplos mostram os resultados dos comandos **pvs** e **lvs** com o argumento **-P** especificado quando uma perna de um volume lógico espelhado falhar.

```
root@link-08 ~]#  vgs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  VG    #PV #LV #SN Attr   VSize VFree Devices
  corey   4   4   0 rz-pnc 1.58T 1.34T
my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T unknown device(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdb1(0)
```

```
[root@link-08 ~]# lvs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  LV                  VG    Attr   LSize   Origin Snap%  Move Log
Copy%  Devices
  my_mirror           corey mwi-a- 120.00G
my_mirror_mlog   1.95 my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  [my_mirror_mimage_0] corey iwi-ao 120.00G
unknown device(0)
  [my_mirror_mimage_1] corey iwi-ao 120.00G
/dev/sdb1(0)
  [my_mirror_mlog]     corey lwi-ao   4.00M
/dev/sdd1(0)
```

6.3. RECUPERAÇÃO DE FALHA DO ESPELHO LVM.

Esta seção fornece um exemplo de recuperação de uma situação onde uma perna de um volume LVM espelhado falha por causa que o dispositivo subjacente para volume físico falhou e o parâmetro **mirror_log_fault_policy** está configurado para **remove**, requerendo que você manualmente reconstrua o espelho. Para informações sobre configurar o parâmetro **mirror_log_fault_policy**, consulte a Seção 6.3, “Recuperação de Falha do Espelho LVM.”.

Quando uma perna do espelho falha, o LVM converte o volume espelhado em um volume linear, o qual continua a operar como antes mas sem a redundância espelhada. Neste momento, você pode adicionar um novo disco no sistema para usar como um dispositivo físico substituto e reconstruir o espelho.

O comando a seguir cria volumes físicos que serão usados para o espelho.

```
[root@link-08 ~]# pvcreate /dev/sd[abcdefgh][12]
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sda2" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdb2" successfully created
  Physical volume "/dev/sdc1" successfully created
  Physical volume "/dev/sdc2" successfully created
  Physical volume "/dev/sdd1" successfully created
  Physical volume "/dev/sdd2" successfully created
  Physical volume "/dev/sde1" successfully created
  Physical volume "/dev/sde2" successfully created
  Physical volume "/dev/sdf1" successfully created
  Physical volume "/dev/sdf2" successfully created
  Physical volume "/dev/sdg1" successfully created
  Physical volume "/dev/sdg2" successfully created
  Physical volume "/dev/sdh1" successfully created
  Physical volume "/dev/sdh2" successfully created
```

Os comandos a seguir criam o grupo de volume **vg** e o volume espelhado **groupfs**.

```
[root@link-08 ~]# vgcreate vg /dev/sd[abcdefgh][12]
  Volume group "vg" successfully created
[root@link-08 ~]# lvcreate -L 750M -n groupfs -m 1 vg /dev/sda1 /dev/sdb1
/dev/sdc1
  Rounding up size to full physical extent 752.00 MB
  Logical volume "groupfs" created
```

Você pode usar o comando **lvs** para verificar o layout do volume espelhado e os dispositivos subjacentes para a perna do espelho e o log do espelho. Observe que no primeiro exemplo o espelho ainda não está completamente sincronizado, portanto você precisa esperar até que o campo **Copy%** seja exibido 100.00 antes de continuar.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy% Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
21.28 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)

[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy%  Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
100.00 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
```

```
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M     i
/dev/sdc1(0)
```

Neste exemplo, a perna principal do espelho **/dev/sda1** falha. Qualquer atividade de gravação no volume espelhado faz com que o LVM detecte o espelho defeituoso. Quando isto ocorre, o LVM converte o espelho em um volume único linear. Neste caso, para iniciar a conversão, executamos um comando **dd**.

```
[root@link-08 ~]# dd if=/dev/zero of=/dev/vg/groupfs count=10
10+0 records in
10+0 records out
```

Você pode usar o comando **lvs** para verificar se o dispositivo é linear agora. Podem ocorrer erros de entrada e saída devido aos discos falhos.

```
[root@link-08 ~]# lvs -a -o +devices
  /dev/sda1: read failed after 0 of 2048 at 0: Input/output error
  /dev/sda2: read failed after 0 of 2048 at 0: Input/output error
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  groupfs vg   -wi-a- 752.00M                               /dev/sdb1(0)
```

Neste momento você ainda deve conseguir usar o volume lógico, mas não haverá redundancia de espelho.

Para reconstruir o volume espelhado, você substitui o drive quebrado e recria o volume físico. Se você usar o mesmo disco ao invés de substituí-lo por um novo, você verá avisos de "inconsistencia" quando executar o comando **pvcreate**. Você pode previnir que este aviso apareça executando o comando **vgreduce --removemissing**.

```
[root@link-08 ~]# pvcreate /dev/sdi[12]
  Physical volume "/dev/sdi1" successfully created
  Physical volume "/dev/sdi2" successfully created

[root@link-08 ~]# pvscan
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdi1           lvm2 [603.94 GB]
  PV /dev/sdi2           lvm2 [603.94 GB]
  Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

Depois que você estender o grupo de volume original pelo volume físico novo.

```
[root@link-08 ~]# vgextend vg /dev/sdi[12]
  Volume group "vg" successfully extended

[root@link-08 ~]# pvscan
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdi1   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  PV /dev/sdi2   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  Total: 16 [2.11 TB] / in use: 16 [2.11 TB] / in no VG: 0 [0   ]
```

Converta o volume linear de volta ao seu estado espelhado original.

```
[root@link-08 ~]# lvconvert -m 1 /dev/vg/groupfs /dev/sdi1 /dev/sdb1
/dev/sdc1
  Logical volume mirror converted.
```

Você poderá usar o comando **lvs** para verificar se o espelho foi restaurado.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                VG   Attr   LSize   Origin Snap%  Move Log
Copy% Devices
  groupfs           vg   mwi-a- 752.00M                    groupfs_mlog
68.62 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdi1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)
```

6.4. RECUPERANDO METADADOS DE VOLUME FÍSICO

Se a área de metadados de grupo de volume de um volume físico for sobrescrita acidentalmente ou destruída, você obterá uma mensagem de erro indicando que a área de metadados está incorreta ou que o sistema não pôde encontrar o volume físico com o UUID específico. Você pode conseguir recuperar os dados do volume físico gravando uma nova área de metadados no volume físico, especificando o mesmo UUID como sendo os metadados perdidos.

ATENÇÃO

Você não deve tentar este procedimento com um volume lógico LVM funcionando. Você perderá seus dados se especificar o UUID incorreto.

Os exemplos a seguir demonstram o tipo de resultado que você obterá se a área de metadados estiver faltando ou corrompida.

```
[root@link-07 backup]# lvs -a -o +devices
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  ...
```

Você pode conseguir encontrar o UUID para o volume físico que foi sobrescrito, olhando no diretório **/etc/lvm/archive**. Procure no arquivo ***VolumeGroupName_xxxx*.vg** pelos últimos metadados LVM conhecidos válidos para aquele grupo de volume.

Alternativamente, você verá que ao desativar o volume e configurar o argumento **partial** (**-P**), você conseguirá encontrar o UUID do volume físico corrompido.

```
[root@link-07 backup]# vgchange -an --partial
  Partial mode. Incomplete volume groups will be activated read-only.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  ...
```

Use os argumentos **--uuid** e **--restorefile**do comando **pvcreate** para recuperar o volume físico. O exemplo a seguir rotula o dispositivo **/dev/sdh1** como volume físico com o UUID indicado acima, **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**. Este comando restaura o rótulo do volume físico com os metadados contidos em **VG_00050.vg**, os metadados corretos arquivados mais recentes para o grupo de volume. O argumento **restorefile** instrui o comando **pvcreate** a fazer o novo volume físico compatível com o antigo no grupo de volume, certificando que o novo metadado não seja colocado onde o volume físico antigo continha dados (o que poderia acontecer, por exemplo, se o comando original **pvcreate** tivesse usado os argumentos da linha de comando que controlam alocação de metadados ou se o volume físico fosse criado originalmente usando uma versão diferente de software que usava padrões diferentes). O comando **pvcreate** sobrescreve somente as áreas de metadados LVM e não afeta as áreas de dados existentes.

```
[root@link-07 backup]# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-
NYM5Sk" --restorefile /etc/lvm/archive/VG_00050.vg /dev/sdh1
  Physical volume "/dev/sdh1" successfully created
```

Você pode então usar o comando **vgcfgrestore** para recuperar os metadados do grupo de volume.

```
[root@link-07 backup]# vgcfgrestore VG
  Restored volume group VG
```

Você pode agora exibir os volumes lógicos.

```
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi--- 300.00G                                /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi--- 300.00G                                /dev/sdh1
(34728),/dev/sdb1(0)
```

Os seguintes comandos ativam os volumes e exibem os volumes ativos.

```
[root@link-07 backup]# lvchange -ay /dev/VG/stripe
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(34728),/dev/sdb1(0)
```

Se os metadados do LVM em disco tomam ao menos o mesmo espaço que o espaço sobrescrito, este comando pode recuperar o volume físico. Se o que sobrescreveu o metadado passou da área de metadados, os dados no volume podem ter sido afetados. Você pode conseguir usar o comando **fsck** para recuperar aquele dado.

6.5. SUBSTITUINDO UM VOLUME FÍSICO AUSENTE

Se um volume físico falhar ou se precisar ser substituído, você pode rotular um novo volume físico para substituir aquele que foi perdido no grupo de volume existente utilizando o mesmo procedimento da recuperação de metadados de volume físico, descrito na Seção 6.4, “Recuperando Metadados de Volume Físico”. Você pode usar os argumentos **--partial** e **--verbose** do comando **vgdisplay** para exibir os UUIDs e tamanhos de quaisquer volumes físicos que não estejam mais presentes. Se você quiser substituir outro volume físico do mesmo tamanho, você pode usar o comando **pvcreate** pelos argumentos **--restorefile** e **--uuid** para inicializar um novo dispositivo com o mesmo UUID que o volume físico ausente. Você pode então usar o comando **vgcfgrestore** para restaurar o metadado do grupo de volume.

6.6. REMOVENDO OS VOLUMES FÍSICOS AUSENTES DE UM GRUPO DE VOLUME.

Se você perder um volume físico, você poderá ativar os volumes físicos restantes no grupo de volume com o argumento **--partial** do comando **vgchange**. Você pode então remover todos os volumes lógicos que usaram aquele volume físico de um grupo de volumes com o argumento **--removemissing** do comando **vgreduce**.

Recomenda-se que você execute o comando **vgreduce** com o argumento **--test** para verificar o que você estará destruindo.

Como a maior parte das operações do LVM, o comando **vgreduce** é reversível se você usar o comando **vgcfgrestore** imediatamente para restaurar os metadados do grupo de volume para seu estado anterior. Por exemplo, se você usou o argumento **--removemissing** do comando **vgreduce** sem o argumento **--test** e ver que removeu os volumes lógicos que você queria manter, você ainda poderá substituir o volume físico e usar outro comando **vgcfgrestore** para retornar o grupo de volume ao seu estado anterior.

6.7. EXTENSÕES LIVRES INSUFICIENTES PARA UM VOLUME LÓGICO

Você poderá obter mensagem de erro "Insufficient free extents" ao criar um volume lógico quando você acha que possui extensões suficientes baseados nos resultados dos comandos **vgdisplay** ou **vgs**. Isto acontece porque estes comandos arredondam números em 2 decimos para fornecer um resultado legível. Para especificar um tamanho exato, use a contagem de extensão física livre ao invés de alguns múltiplos de bytes para determinar o tamanho do volume lógico.

O comando **vgdisplay**, por padrão, inclui esta linha de resultado que indica as extensões físicas livres.

```
# vgdisplay
  --- Volume group ---
  ...
  Free  PE / Size       8780 / 34.30 GB
```

Como forma alternativa, você pode usar os argumentos **vg_free_count** e **vg_extent_count** do comando **vgs** para exibir as extensões livres e o número total de extensões.

```
[root@tng3-1 ~]# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree  Free #Ext
  testvg   2   0   0 wz--n- 34.30G 34.30G 8780 8780
```

Com 8780 extensões físicas livres, você pode executar o seguinte comando, usando o argumento l em minúsculo para usar as extensões ao invés de bytes:

```
# lvcreate -l8780 -n testlv testvg
```

Isto usa todas as extensões livres no grupo de volume.

```
# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree Free #Ext
  testvg   2   1   0 wz--n- 34.30G    0     0 8780
```

Como forma alternativa, você pode estender o volume lógico para usar uma porcentagem do espaço livre restante no grupo de volume usando o argumento **-l** do comando **lvcreate**. Para mais informações, veja a Seção 4.4.1, “Criando Volumes Lógicos Lineares”.

CAPÍTULO 7. ADMINISTRAÇÃO COM O LVM GUI

Além da Interface de Linha de Comando (CLI), o LVM fornece uma Interface de Usuário Gráfica (GUI) que você pode usar para configurar os volumes lógicos LVM. Você pode chamar esta ferramenta digitando **`system-config-lvm`**. O capítulo do LVM do *Guia de Administração de Armazenamento* fornece instruções passo-a-passo para configurar um volume lógico LVM usando esta ferramenta.

APÊNDICE A. O MAPEADOR DE DISPOSITIVO

O Mapeador de Dispositivo é um driver do kernel que fornece uma estrutura para gerenciamento de volume. Ele fornece uma forma genérica de criar dispositivos mapeados, que podem ser usados como volumes lógicos. Isto não trata especificamente sobre grupos de volumes ou formatos de metadados.

O Mapeador de Dispositivos fornece a base para uma série de tecnologias de alto nível. Além do LVM, o Mapeador de Dispositivos multipath e o comando **dmraid** usam o Mapeador de Dispositivos. A interface de aplicação para o Mapeador de Dispositivos é a **ioctl** system call. A interface de usuário é o comando **dmsetup**.

Volumes lógicos LVM são ativados usando o Mapeador de Dispositivos. Cada volume lógico é convertido em um dipositivo mapeado. Cada segmento converte em uma linha na tabela de mapeamento que descreve o dispositivo. O Mapeador de Dispositivo suporta uma variedade de alvos de mapeamento, incluindo mapeamento linear, mapeamento distribuído e mapeamento de erro. Então, por exemplo, dois discos podem estar concatenados em um volume lógico com um par de mapeamentos lineares, uma para cada disco. Quando o LVM cria um volume, cria um dispositivo mapeado que pode ser consultado com o comando **dmsetup**. Para informações sobre o formato dos dispositivos em uma tabela mapeada, veja a Seção A.1, “Mapeamentos da Tabela de Dispositivo”. Para informações sobre o uso do comando **dmsetup** para consultar um dispositivo, veja a Seção A.2, “O Comando dmsetup”.

A.1. MAPEAMENTOS DA TABELA DE DISPOSITIVO

Um dispositivo mapeado é definido por uma tabela que especifica como mapear cada classe de setores lógicos do dispositivo usando um mapeamento de Tabela de Dispositivo suportado. A tabela para um dispositivo mapeado é construída a partir de uma lista de linhas na forma:

```
start length mapping [mapping_parameters...]
```

Na primeira linha da tabela de um Dispositivo Mapeado, o parâmetro ***start*** deve ser igual a 0. Os parâmetros ***start*** + ***length*** em uma linha devem ser iguais ao ***start*** na próxima linha. Os parâmetros que são especificados em uma linha da tabela de mapeamento dependem de qual tipo ***mapping*** é especificado na linha.

Tamanhos no Mapeador de Dispositivo são sempre especificados em setores (512 bytes).

Quando um dispositivo é especificado como um parâmetro de mapeamento no Mapeador de Dispositivos, ele pode ser referenciado pelo nome de dispositivo no sistema de arquivos (por exemplo, **/dev/hda**) ou pelos números maiores e menores no formato ***major***:***minor***. O formato maior:menor é preferido por que evita pesquisas no caminho de nomes.

Segue um exemplo de tabela de mapeamento para um dispositivo. Nesta tabela existem quatro alvos lineares:

```
0 35258368 linear 8:48 65920
35258368 35258368 linear 8:32 65920
70516736 17694720 linear 8:16 17694976
88211456 17694720 linear 8:16 256
```

Os 2 primeiros parâmetros de cada linha são o início do bloco de segmento e a extensão do segmento. A próxima palavra é o alvo de mapeamento, no qual em todos os casos deste exemplo são **linear**. O resto de cada linha consiste de parâmetros para um alvo **linear**.

As subseções a seguir descrevem o formato dos seguintes mapeamentos:

- linear
- distribuído
- espelho
- snapshot e origem de snapshot
- erro
- zero
- multipath
- crypt

A.1.1. O Alvo do Mapeamento Linear

O alvo do mapeamento linear mapeia uma variação contínua de blocos em outro dispositivo de bloco. O formato de um alvo linear é o seguinte:

```
start length linear device offset
```

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

device

dispositivo de bloco, referenciado pelo nome do dispositivo no sistema de arquivo ou pelos números maiores ou menores no formato ***major*:*minor***

offset

offset inicial do mapeamento no dispositivo

O seguinte exemplo exibe um alvo linear com um início de bloco no dispositivo virtual de 0, um comprimento de segmento de 1638400, um par de números maior:menor de 8:2 e um offset inicial de dispositivo de 41146992.

```
0 16384000 linear 8:2 41156992
```

O exemplo a seguir exibe um alvo linear com o parâmetro de dispositivo especificado como o dispositivo **/dev/hda**.

```
0 20971520 linear /dev/hda 384
```

A.1.2. O Alvo de Mapeamento distribuído

O alvo de mapeamento distribuído suporta distribuições em dispositivos físicos. Ele toma como argumentos o número de distribuições e os tamanhos das partes distribuídas seguido de uma lista de pares de nomes de dispositivos e setores. O formato do alvo distribuído é o seguinte:

```
start length striped #stripes chunk_size device1 offset1 ... deviceN
offsetN
```

Existe um conjunto de parâmetros de ***device*** e ***offset*** para cada distribuição.

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

#stripes

número de distribuições para o dispositivo virtual

chunk_size

número de setores gravados em cada distribuição antes de mudar para a próxima, deve ser potência de 2 pelo menos e tão grande quanto o tamanho de página do kernel.

device

dispositivo de bloco, referenciado pelo nome do dispositivo no sistema de arquivos ou pelos números maiores e menores no formato ***major***:***minor***.

offset

offset inicial do mapeamento no dispositivo

O exemplo a seguir demonstra um alvo distribuído com três distribuições e uma parte de tamanho 128:

```
0 73728 striped 3 128 8:9 384 8:8 384 8:7 9789824
```

0

bloco inicial em um dispositivo virtual

73728

comprimento deste segmento

striped 3 128

distribuição em três dispositivos com tamanho em partes de 128 blocos

8:9

números major:minor do primeiro dispositivo

384

offset inicial de mapeamento no primeiro dispositivo

8:8

números major:minor do segundo dispositivo

384

offset inicial de mapeamento no segundo dispositivo

8:7

números major: minor do terceiro dispositivo

9789824

offset inicial do mapeamento no terceiro dispositivo

O exemplo seguinte mostra um alvo distribuído para 2 distribuições com partes de 256 KiB, com os parâmetros de dispositivo especificados pelos nomes de dispositivos no sistema de arquivo ao invés dos números maiores e menores.

```
0 65536 striped 2 512 /dev/hda 0 /dev/hdb 0
```

A.1.3. O Alvo de Mapeamento do espelho

O alvo de mapeamento de espelho suporta o mapeamento de um dispositivo lógico espelhado. O formato de um alvo espelhado é como se segue:

```
start length mirror log_type #logargs logarg1 ... logargN #devs device1
offset1 ... deviceN offsetN
```

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

log_type

Os tipos de log possíveis e seus argumentos são como se segue:

core

O espelho é local e o log do espelho é mantido na memória do núcleo. Este tipo de log leva 1 - 3 argumentos:

regionsize [[**no**]**sync**] [**block_on_error**]

disk

O espelho é local e o log do espelho é mantido no disco. Este tipo de log leva de 2 - 4 argumentos:

logdevice regionsize [[**no**]**sync**] [**block_on_error**]

clustered_core

O espelho é clusterizado e o log do espelho é mantido na memória do núcleo. Este tipo de log leva de 2 - 4 argumentos:

regionsize UUID [[**no**]**sync**] [**block_on_error**]

clustered_disk

O espelho é clusterizado e o log do espelho é mantido no disco. Este tipo de log leva de 3 - 5 argumentos:

logdevice regionsize UUID [[**no**]**sync**] [**block_on_error**]

O LVM mantém um log pequeno que é usado para manter o controle de quais regiões estão em sync com o espelho ou espelhos. O argumento *regionsize* especifica o tamanho dessas regiões.

Em um ambiente clusterizado, o argumento *UUID* é um único indentificador associado com o log do dispositivo espelho para que o estado do log possa ser mantido através de todo o cluster.

O argumento opcional **[no]sync** pode ser usado para especificar o espelho como "in-sync" ou "out-of-sync". O argumento **block_on_error** é usado para dizer ao espelho para responder aos erros e não ignora-los.

#log_args

número de argumentos de log que serão especificados no mapeamento

logargs

os argumentos de logs para o espelho; o número de argumentos de logs fornecido é especificado pelo parâmetro ***#log-args*** e os argumentos de logs válidos são determinados pelo parâmetro ***log_type***.

#devs

o número de pernas no espelho; um dispositivo e um offset são especificados para cada perna.

device

dispositivo de bloco para cada perna do espelho, referenciados pelo nome do dispositivo no sistema de arquivos ou pelos números maiores e menores no formato ***major***:***minor***. Um dispositivo de bloco e offset são especificados para cada perna do espelho, como indicado pelo parâmetro ***#devs***.

offset

offset inicial do mapeador no dispositivo. Um dispositivo de bloco e offset são específicados para cada perna do espelho, como indicado pelo parâmetro ***#devs***.

O exemplo a seguir demonstra um alvo de mapeamento de espelho para um espelho clusterizado com um log de espelho mantido no disco.

```
0 52428800 mirror clustered_disk 4 253:2 1024 UUID block_on_error 3 253:3
0 253:4 0 253:5 0
```

0

bloco inicial em um dispositivo virtual

52428800

comprimento deste segmento

mirror clustered_disk

o alvo do espelho com o tipo de log especificando que aquele espelho está clusterizado e o log do espelho é mantido no disco.

4

4 argumentos de log de espelho seguirão

253:2

números de dispositivo de log major: minor

1024

tamanho da região que o log do espelho usa para manter registro do que está em sincronia

UUID

UUID do dispositivo do log de espelho para manter a informação do log em todo o cluster

`block_on_error`

espelho deve responder a erros

3

número de pernas no espelho

253:3 0 253:4 0 253:5 0

números major:minor e offset para dispositivos que constituem cada perna de um espelho

A.1.4. O Alvo de Mapeamento Snapshot e snapshot-origem

Quando você criar o primeiro snapshot LVM de um volume, quatro dispositivos do Mapeador de Dispositivo serão usados:

1. Um dispositivo com um mapeamento **`linear`** contendo a tabela de mapeamento original do volume fonte.
2. Um dispositivo com um mapeamento **`linear`** usado como dispositivo copy-on-write (COW) para o volume de origem; para cada escrita, os dados originais são salvos no dispositivo COW de cada snapshot para manter seu conteúdo visível inalterado (até o dispositivo COW encher).
3. Um dispositivo com um mapeamento **`snapshot`** combinando o #1 e #2, os quais são volumes snapshot visíveis.
4. O volume "original" (que usa o número de dispositivo usado pelo volume fonte original), o qual a tabela é substituida por um mapeador "snapshot-origin" do dispositivo #1.

Um esquema de nomeação fixo é usado para criar dispositivos. Por exemplo, você poderá usar os seguintes comandos para criar um volume LVM chamado **`base`** e um volume snapshot chamado **`snap`** baseado nesse volume.

```
# lvcreate -L 1G -n base volumeGroup
# lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

Isso produz quatro dispositivos, que você pode ver com os seguintes comandos:

```
# dmsetup table|grep volumeGroup
volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-snap-cow: 0 204800 linear 8:19 2097536
volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-*
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-
real
brw-------  1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-
cow
brw-------  1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
brw-------  1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

O formato para o alvo **snapshot-origin** se segue:

```
start length snapshot-origin origin
```

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

origin

volume base do snapshot

O **snapshot-origin** terá normalmente um ou mais snapshots baseados nele. Leituras serão mapeadas diretamente para o dispositivo de suporte. Para cada gravação, os dados originais serão salvos no dispositivo COW de cada snapshot para manter inalterados o seu conteúdo visível até que o dispositivo COW encha.

O formato para o alvo **snapshot** é como se segue:

```
start length snapshot origin COW-device P|N chunksize
```

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

origin

volume base do snapshot

COW-device

Dispositivo no qual as partes modificadas de dados são armazenadas

P|N

P (Persistente) ou N (Não persistente); indica se o snapshot sobreviverá após o reboot. Para snapshots transientes (N), menos metadados devem ser salvos no disco; eles podem ser mantidos na memória pelo kernel.

chunksize

Tamanho em setores de partes modificadas de dados que serão armazenados no dispositivo COW.

O exemplo a seguir mostra um alvo **snapshot-origin** com um dispositivo original de 254:11.

```
0 2097152 snapshot-origin 254:11
```

O seguinte exemplo exibe um alvo **snapshot** com um dispositivo de origem de 254:11 e um dispositivo COW de 254:12. Este dispositivo snapshot é persistente em reboots e os tamanhos das partes para os dados armazenados no dispositivo COW é de 16 setores.

```
0 2097152 snapshot 254:11 254:12 P 16
```

A.1.5. O Alvo de Mapeamento de erro

Com um alvo de mapeamento de erro, qualquer operação de E/S no setor mapeado irá falhar.

Um alvo de mapeamento de erro pode ser usado para testes. Para testar como o dispositivo se comportará durante uma falha, você pode criar um mapeador de dispositivo com um setor defeituoso no meio de um dispositivo ou você pode trocar a perna de um espelho e substituir a perna com um alvo errado.

Um alvo de erro pode ser usado no lugar de um dispositivo com falha, como uma maneira de evitar tempos de espera e tentativas no dispositivo em questão. Ele pode servir como um alvo intermediário enquanto você re-organiza os metadados LVM durantes falhas.

O alvo de mapeamento **error** não leva parâmetros adicionais além dos parâmetros *start* e *length*.

O exemplo a seguir mostra um alvo **error**.

```
0 65536 error
```

A.1.6. O Alvo de Mapeamento zero

O alvo de mapeamento zero é um dispositivo de bloco equivalente a **/dev/zero**. A operação de leitura para este mapeamento retorna blocos de zeros. Dados escritos para este mapeamento são descartados, mas a escrita é bem sucedida. O alvo de mapeamento **zero** não leva parâmetros adicionais além do parâmetros *start* e *length*.

O seguinte exemplo mostra um alvo **zero** para o Dispositivo 16Tb.

```
0 65536 zero
```

A.1.7. O Alvo de Mapeamento multipath

O alvo de mapeamento multipath suporta o mapeamento de um dispositivo multipath. O formato para o alvo **multipath** é como segue:

```
start length  multipath  #features [feature1 ... featureN] #handlerargs
[handlerarg1 ... handlerargN] #pathgroups pathgroup pathgroupargs1 ...
pathgroupargsN
```

Existe um conjunto de parâmetros de ***pathgroupargs*** para cada grupo de path.

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

#features

A quantidade de características de multipath, seguidos pelas características. Se este parâmetro for zero, então não há parâmetro ***feature*** e o próximo parâmetro do dispositivo de mapeamento é ***#handlerargs***. Atualmente há uma característica multipath suportada, **queue_if_no_path**. Esta indica que o dispositivo multipath está atualmente configurado para enfileirar operações de E/S se não houver um path disponível.

Por exemplo, se a opção **no_path_retry** no arquivo **multipath.conf** foi configurado para enfileirar operações de E/S somente até que todos os paths tenham sido marcados como reprovados após que um certo número de tentativas foram feitas para usar os paths, o mapeamento apareceria como a seguir até que todos os controladores de paths falhem o número de vezes especificadas.

```
0 71014400 multipath 1 queue_if_no_path 0 2 1 round-robin 0 2 1 66:128 \
1000 65:64 1000 round-robin 0 2 1 8:0 1000 67:192 1000
```

Após todos os controladores de paths falharem o número de vezes especificadas, o mapeamento aparece como a seguir:

```
0 71014400 multipath 0 0 2 1 round-robin 0 2 1 66:128 1000 65:64 1000 \
round-robin 0 2 1 8:0 1000 67:192 1000
```

#handlerargs

A quantidade de argumentos do manipulador de hardware, seguido por estes argumentos. Um manipulador de hardware especifica um módulo que será usado para executar ações específicas de hardware quando trocando grupos de paths ou manipulando erros de E/S. Se este é configurado como 0, então o próximo parâmetro é ***#pathgroups***.

#pathgroups

O número de grupos paths. Um grupo paths é o conjunto de caminhos em que um dispositivo multipath irá carregar equilíbrio. Há uma configuração de parâmetro ***pathgroupargs*** para cada grupo de path.

pathgroup

O próximo grupo de caminho a ser tentado

pathgroupsargs

Cada grupo de caminho consiste dos seguintes argumentos:

```
pathselector #selectorargs #paths #pathargs device1 ioreqs1 ... deviceN
ioreqsN
```

Existe um conjunto de argumentos de path para cada caminho no grupo.

pathselector

Especifica o algorítmo em uso para determinar qual path neste grupo de caminho usar para a próxima operação de E/S.

#selectorargs

O número de argumentos seletores de path que seguem este argumento no mapeador multipath. Atualmente, o valor deste argumento é sempre 0.

#paths

O número de paths neste grupo de caminho

#pathargs

O número de argumentos path especificados para cada caminho neste grupo. Atualmente este número é sempre 1, o argumento ***ioreqs***.

device

O número do dispositivo de bloco de path, referenciado pelos números maiores e menores no formato ***major*:*minor***

ioreqs

O número de requisições de E/S para rotear neste caminho antes de mudar para o caminho seguinte no grupo atual.

A Figura A.1, “Alvo de Mapeamento Multipath” mostra o formato de um alvo multipath com dois grupos de caminho.

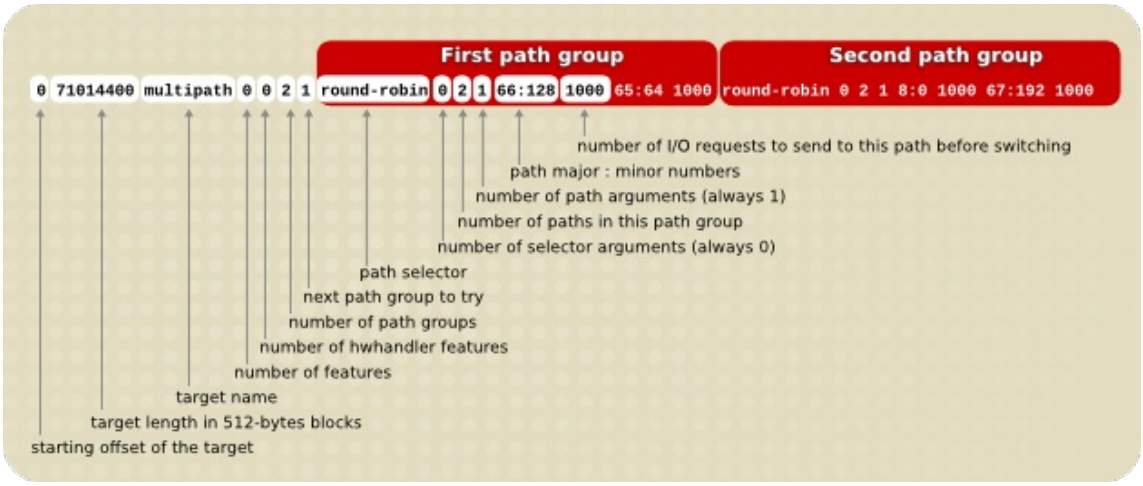


Figura A.1. Alvo de Mapeamento Multipath

O exemplo seguinte mostra uma pura definição de alvo failover no mesmo dispositivo multipath. Neste alvo existem 4 grupos de caminho, com somente um caminho aberto por grupo de caminho, então o dispositivo multpath usará somente um caminho por vez.

```
0 71014400 multipath 0 0 4 1 round-robin 0 1 1 66:112 1000 \
round-robin 0 1 1 67:176 1000 round-robin 0 1 1 68:240 1000 \
round-robin 0 1 1 65:48 1000
```

O exemplo seguinte mostra uma definição de propagação (multibus) de alvo para o mesmo dispositivo multipathed. Neste alvo há somente um grupo de caminho, que inclui todos os paths. Nesta configuração, multipath distribui a carga uniformemente a todos os caminhos.

```
0 71014400 multipath 0 0 1 1 round-robin 0 4 1 66:112 1000 \
 67:176 1000 68:240 1000 65:48 1000
```

Para mais informações sobre multipathing, veja o documento *Utilizando o Mapeador de Dispositivo Multipath*

A.1.8. O Alvo de Mapeamento crypt.

O alvo **crypt** criptografa os dados passando pelo dispositivo especificado. Ele utiliza o kernelCrypto API.

O formato para o alvo **crypt** é como a seguir:

```
start length crypt cipher key IV-offset device offset
```

start

bloco inicial em um dispositivo virtual

length

comprimento deste segmento

cipher

O cipher consiste de ***cipher[-chainmode]-ivmode[:iv options]***.

cipher

Ciphers disponíveis estão listados em **/proc/crypto** (por exemplo, **aes**).

chainmode

Sempre use o **cbc**. Não use o **ebc**; pois este não utiliza um vetor inicial (IV).

ivmode[:iv options]

IV é um setor inicial usado para variar a criptografia. O modo IV é **plain** ou **essiv:hash**. Um ***ivmode*** do **-plain** usa o número de setor (mais offset IV) como o IV. Um ***ivmode*** do **-essiv** é um aprimoramento para evitar a fraqueza de marca d'agua.

key

A chave de criptografia, fornecida em hex

IV-offset

Vetor inicial (IV) offset

device

dispositivo de bloco, referenciado pelo nome do dispositivo no sistema de arquivo ou pelos números maiores ou menores no formato ***major*:*minor***

offset

offset inicial do mapeamento no dispositivo

Segue um exemplo de um alvo **crypt**

```
0 2097152 crypt aes-plain 0123456789abcdef0123456789abcdef 0 /dev/hda 0
```

A.2. O COMANDO DMSETUP

O comando **dmsetup** é um wrapper de linha de comando para comunicação com o Mapeador de Dispositivo. Para informações gerais do sistema sobre dispositivos LVM, você pode achar úteis as opções **info**, **ls**, **status** e **deps** do comando **dmsetup**, como descrito nas seguintes subseções.

Para informações sobre opções adicionais e capacidades do comando **dmsetup**, veja a página man **dmsetup**(8).

A.2.1. O Comando dmsetup info

O comando **dmsetup info *device*** fornece um resumo de informações sobre os Dispositivos de Mapeamento. Se você não especificar um nome de dispositivo, o resultado terá informações sobre todos os Dispositivos Mapeadores atualmente configurados. Se você especificar um dispositivo, então este comando fornece informações para tal dispositivo somente.

O comando **dmsetup info** fornece informações nas seguintes categorias:

Name

O nome do dispositivo. Um dispositivo LVM é expresso como o nome do grupo de volume e o nome do volume lógico separados por um hífen. Um hífen presente nonome original é traduzido com dois hífens.

State

Possíveis estados dos dispositivos são **SUSPENDED**, **ACTIVE** e **READ-ONLY**. O comando **dmsetup suspend** configura um estado de dispositivo para **SUSPENDED**. Quando um dispositivo é suspenso, todas operações E/S desse dispositivo param. O comando **dmsetup resume** restaura em estado de dispositivo para **ACTIVE**.

Read Ahead

O número de bloco de dados que o sistema lê adiante para qualquer arquivo aberto no qual operações de leitura estão em andamento. Por padrão, o kernel escolhe um valor adequado automaticamente. Você pode alterar este valor com a opção **--readahead** do comando **dmsetup**.

Tables present

Possíveis estados para esta categoria são **LIVE** e **INACTIVE**. Um estado **INACTIVE** indica que uma tabela foi carregada na qual será trocada quando um comando **dmsetup resume** restaura o estado do dispositivo para **ACTIVE**, para o ponto que o estado da tabela se torna **LIVE**. Para informações, veja a página man **dmsetup**.

Open count

A contagem de referência aberta indica quantas vezes o dispositivo é aberto. Um comando **mount** abre um dispositivo.

Event number

O número atual de eventos recebidos. Emitindo um comando **dmsetup wait *n*** permite que o usuário espere que o evento mude, bloqueando a chamada até que esta seja recebida.

Major, minor

Número de dispositivo major e minor

Number of targets

O número de fragmentos que compõem um dispositivo. Por exemplo, um dispositivo linear ao longo de 3 discos teria três alvos. Um dispositivo linear composto do início e término de um disco mas não o meio, teria 2 alvos.

UUID

O UUID do dispositivo

O exemplo a seguir mostra um resultado parcial para o comando **dmsetup info**.

```
[root@ask-07 ~]# dmsetup info
Name:              testgfsvg-testgfslv1
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 2
Number of targets: 2
UUID: LVM-K528WUGQgPadNXYcFrrf9LnPlUMswgkCkpgPIgYzSvigM7SfeWCypddNSWtNzc2N
...
Name:              VolGroup00-LogVol00
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        1
Event number:      0
Major, minor:      253, 0
Number of targets: 1
UUID: LVM-tOcS1kqFV9drb0X1Vr8sxeYP0tqcrpdegyqj5lZxe45JMGlmvtqLmbLpBcenh2L3
```

A.2.2. O Comando dmsetup ls

Você pode listar os nomes de dispositivos dos dispositivos mapeados com o comando **dmsetup ls**.

Você pode listar dispositivos que têm pelo menos um alvo de um tipo especificado com o comando **dmsetup ls --target *target_type***. Para outras opções do **dmsetup ls**, veja a página man **dmsetup**.

O exemplo a seguir mostra o comando para listar os nomes de dispositivos dos dispositivos mapeados configurados.

```
[root@ask-07 ~]# dmsetup ls
testgfsvg-testgfslv3    (253, 4)
testgfsvg-testgfslv2    (253, 3)
testgfsvg-testgfslv1    (253, 2)
VolGroup00-LogVol01     (253, 1)
VolGroup00-LogVol00     (253, 0)
```

O exemplo a seguir mostra o comando para listar os nomes de dispositivos dos mapeamentos de espelho configurados.

```
[root@grant-01 ~]# dmsetup ls --target mirror
lock_stress-grant--02.1722      (253, 34)
lock_stress-grant--01.1720      (253, 18)
lock_stress-grant--03.1718      (253, 52)
lock_stress-grant--02.1716      (253, 40)
lock_stress-grant--03.1713      (253, 47)
lock_stress-grant--02.1709      (253, 23)
lock_stress-grant--01.1707      (253, 8)
lock_stress-grant--01.1724      (253, 14)
lock_stress-grant--03.1711      (253, 27)
```

Configurações LVM que estão empilhadas em multipath ou outros dispositivos do mapeador de dispositivos podem ser complexos para classificar. O comando **dmsetup ls** fornece uma opção **--tree** que exibe dependências entre dispositivos com uma árvore, como no exemplo seguinte.

```
# dmsetup ls --tree
vgtest-lvmir (253:13)
 ├─vgtest-lvmir_mimage_1 (253:12)
 │  └─mpathep1 (253:8)
 │     └─mpathe (253:5)
 │        ├─ (8:112)
 │        └─ (8:64)
 ├─vgtest-lvmir_mimage_0 (253:11)
 │  └─mpathcp1 (253:3)
 │     └─mpathc (253:2)
 │        ├─ (8:32)
 │        └─ (8:16)
 └─vgtest-lvmir_mlog (253:4)
    └─mpathfp1 (253:10)
       └─mpathf (253:6)
          ├─ (8:128)
          └─ (8:80)
```

A.2.3. O Comando dmsetup status

O comando **dmsetup status *device*** fornece informações de estado para cada alvo num dispositivo especificado. Se você não especificar um nome de dispositivo, o resultado será informações

sobre todos os Dispositivos Mapeadores atualmente configurados. Você pode listar somente o estado de dispositivos que têm pelo menos um alvo de um tipo especificado com o comando **dmsetup status --target *target_type***.

O exemplo a seguir mostra o comando para listar o estado dos alvos em todos os dispositivos mapeados configurados atualmente.

```
[root@ask-07 ~]# dmsetup status
testgfsvg-testgfslv3: 0 312352768 linear
testgfsvg-testgfslv2: 0 312352768 linear
testgfsvg-testgfslv1: 0 312352768 linear
testgfsvg-testgfslv1: 312352768 50331648 linear
VolGroup00-LogVol01: 0 4063232 linear
VolGroup00-LogVol00: 0 151912448 linear
```

A.2.4. O Comando dmsetup deps

O comando **dmsetup deps *device*** fornece uma lista de pares (maior, menor) para dispositivos referenciados pela tabela de mapeamento para o dispositivo especificado. Se você não especificar um nome de dispositivo, o resultado será informações sobre todos os Dispositivos de Mapeamento atualmente configurados.

O exemplo a seguir mostra o comando para listar as dependências de todos os dispositivos mapedos configurados atualmente.

```
[root@ask-07 ~]# dmsetup deps
testgfsvg-testgfslv3: 1 dependencies    : (8, 16)
testgfsvg-testgfslv2: 1 dependencies    : (8, 16)
testgfsvg-testgfslv1: 1 dependencies    : (8, 16)
VolGroup00-LogVol01: 1 dependencies     : (8, 2)
VolGroup00-LogVol00: 1 dependencies     : (8, 2)
```

O exemplo a seguir mostra o comando para listar as dependências somente do dispositivo **lock_stress-grant--02.1722**:

```
[root@grant-01 ~]# dmsetup deps lock_stress-grant--02.1722
3 dependencies  : (253, 33) (253, 32) (253, 31)
```

A.3. SUPORTE DO MAPEADOR DE DISPOSITIVO PARA O GERENCIADOR DE DISPOSTIVO UDEV

A função primária do gerenciador de dispositivo **udev** é fornecer uma maneira dinâmica de configurar nós no diretório **/dev**. A criação destes nós é direcionada pela aplicação de regras **udev** no espaço do usuário. Estas regras são processadas em eventos **udev** enviados do kernel diretamente como um resultado de adicionar, remover ou alterar particulares dispositivos. Isto fornece um mecanismo conveniente e central para suporte hotplugging.

Além de criar os nós, o gerenciador de dispositivos **udev** é também capaz de criar qualquer link simbólico com seus próprios nomes, fornecendo aos usuários a liberdade de escolher seu um nome personalizado e estrutura de diretório no diretório **/dev**, se necessário.

Cada evento **udev** contém informações básicas sobre o dispositivo sendo processado, tais como seu nome, o sub sistema que pertence, o tipo de dispositivo, seus números maiores e menores usados e o

tipo de evento. Dado isso e tendo a possibilidade de acessar toda a informação encontrada no diretório **/sys** que é também acessível dentro das regras **udev**, os usuários são capazes de utilizar filtros simples baseados nestas informações e rodar as regras condicionalmente baseadas nas mesmas.

O gerenciador de dispositivo **udev** também fornece uma maneira centralizada de configurar permissões nos nodos. Um usuário pode fácilmente adicionar um conjunto de regras personalizadas para definir as permissões para qualquer dispositivo especificado com qualquer parte de informação disponível enquanto o evento está em processamento.

Também é possível adicionar ganchos de programas diretamente nas regras **udev**. O gerenciador de dispositivos **udev** pode chamar estes programas para fornecer maior processamento que seja necessário para manusear o evento. Além disso, o programa pode exportar variáveis de ambiente, como resultado deste processamento. Qualquer resultado dado pode ser usado em outras regras como uma fonte suplementária de informação.

Qualquer software usando a bilbioteca **udev** é capaz de receber e processar eventos **udev** com todas as informações disponíveis, então o processamento não é vinculado somente ao daemon **udev**.

A.3.1. Integração do udev com o Mapeador de Dispositivo

No RHEL 6, o Mapeador de Dispositivo fornece suporte direto para integração **udev**. Isto sincroniza do Mapeador de Dispositivos com todos processamentos **udev** relacionados aos mapeadores de dispositivos, incluindo dispositivos LVM. A sincronização é necessária uma vez que a aplicação de regra no **udev** daemon é uma forma de processamento paralela com o programa que é a fonte das mudanças dos dispositivos (como **dmsetup** e LVM). Sem este suporte, era um problema para um usuário tentar remover um dispositivo que estava ainda aberto e processado pelas regras **udev** como um resultado de mudanças de eventos anteriores; isso era particularmente comum quando havia um curto tempo entre mudanças deste dispositivo.

A versão RHEL 6 fornece suporte oficial para regras **udev** em Dispositivos de Mapeamento em geral e também para LVM. A Tabela A.1, “Regras do udev para os Dispositivos de Mapeador de Dispositivo” resume estas regras, que estão instaladas em **/lib/udev/rules.d**.

Tabela A.1. Regras do udev para os Dispositivos de Mapeador de Dispositivo

Nome de Arquivo	Descrição
10-dm.rules	Contém regras gerais / básicas do Mapeador de Dispositivos e cria sumlinks em **/dev/mapper** com um alvo **/dev/dm-*N*** onde N é um número atribuído dinamicamente ao dispositivo pelo kernel (**/dev/dm-*N*** é um nó) NOTA: nódos **/dev/dm-*N*** *nunca* devem ser usados em scripts para acessar o dispositivo já que o número N é atribuído dinamicamente e muda com a sequência de como dispositivos são ativados. Portanto, nomes verdadeiros no diretório **/dev/mapper** devem ser usados. Este layout é para suportar requerimentos **udev** de como nódos/symlinks devem ser criados.

Nome de Arquivo	Descrição
11-dm-lvm.rules	Contém regras aplicadas para dispositivos LVM e cria symlinks para os volumes lógicos do grupo de volume. Os sumlinks são criados no diretório **/dev/*vgname*** com o alvo **/dev/dm-*N***. NOTA: Para ser consistente com o padrão para nomear todas as futuras regras para sub sistemas de Mapeadores de Dispositivos, regras udev devem seguir o formato **11-dm-*subsystem_name*.rules**. Qualquer usuário **libdevmapper** que proporciona regras **udev** também devem seguir este padrão.
13-dm-disk.rules	Contém regras para serem aplicadas para todos Mapeadores de Dispositivos em geral e cria symlinks no **/dev/disk/by-id**, **/dev/disk/by-uuid** e diretórios **/dev/disk/by-uuid**.
95-dm-notify.rules	Contém a regra para notificar o processo de espera usando **libdevmapper** (assim como o LVM e **dmsetup**). A notificação é feita após que todas as regras anteriores são aplicadas, para assegurar que qualquer processamento **udev** seja completo. Processo notificado é então retomado.

Você pode adicionar regras de permissões personalizadas por meio do arquivo **12-dm-permissions.rules**. Este arquivo *não* é instalado no diretório **/lib/udev/rules**. Ele é encontrado no diretório **/usr/share/doc/device-mapper-*version***. O arquivo **12-dm-permissions.rules** é um molde contendo dicas de como configurar as permissões, baseadas por exemplo em algumas regras correspondentes; o arquivo contém exemplos de algumas situações comuns. Você pode editar este arquivo e colocar manualmente no diretório **/etc/udev/rules.d** onde ele vai sobreviver à atualizações, então as configurações permanecerão.

Estas regras definem todas as variantes que poderiam ser usadas por outras regras enquanto processava os eventos.

As seguintes variáveis são definidas em 10-dm.rules:

- **DM_NAME**: Nome do dispositivo do Mapeador de Dispositivo
- **DM_UUID**: UUID do dispositivo do Mapeador de Dispositivo
- **DM_SUSPENDED**: o estado suspenso do dispostivo do Mapeador de Dispositivo
- **DM_UDEV_RULES_VSN**: Versão de regras **udev** (este é principalmente para todas as outras regras para verificar que as variáveis citadas anteriormente são definidas diretamente pelas regras oficiais do Mapeador de Dispositivo)

As seguintes variáveis estão definidas em **11-dm-lvm.rules**:

- **DM_LV_NAME**: nome do volume lógico

- **DM_VG_NAME**: nome do grupo de volume
- **DM_LV_LAYER**: nome da camada LVM

Todas essas variáveis podem ser usadas no arquivo **12-dm-permissions.rules** para definir a permissão de específicos dispositivos de Mapeadores de Dispositivos, conforme documentado no arquivo **12-dm-permissions.rules**.

A.3.2. Os Comandos e Interfaces que Suportam o udev

A Tabela A.2, "Os comandos dmsetup para Suporte do udev" resume os comandos **dmsetup** que suportam a integração **udev**

Tabela A.2. Os comandos dmsetup para Suporte do udev

Comando	Descrição
dmsetup udevcomplete	Usado para notificar que o udev completou o processo de regras e desbloqueios esperando processo (chamados a partir das regras **udev** no **95-dm-notify.rules**)
dmsetup udevcomplete_all	Usado para propósitos de depuração para desbloquear manualmente todos os processos em espera
dmsetup udevcookies	Usado para propósitos de depuração, para mostrar todos os cookies existentes (semáforos de todo o sistema)
dmsetup udevcreatecookie	Usado para criar um cookie (semaphore) manualmente. Isto é útil para executar mais processos sob um recurso de sincronização.
dmsetup udevreleasecookie	Usado para aguardar por todos os processamentos do **udev** relacionados à todos os processos colocados sob este cookie de sincronização.

As opções do **dmsetup** que suportam a integração do **udev** são estas a seguir.

--udevcookie

Precisa ser definido por todos processos dmsetup que gostariamos de adicionar em uma transação udev. É usado em conjunto com **udevcreatecookie** e **udevreleasecookie**:

```
COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command --udevcookie $COOKIE ....
  dmsetup command --udevcookie $COOKIE ....
  ....
  dmsetup command --udevcookie $COOKIE ....
dmsetup udevreleasecookie --udevcookie $COOKIE
```

Além de usar a opção **--udevcookie**, você pode simplesmente exportar a variável para um ambiente em processo:

```
export DM_UDEV_COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command ...
```

```
dmsetup command ...
...
dmsetup command ...
```

--noudevrules

Desabilita regras udev. Nódos/symlinks serão criados pelo próprio **libdevmapper** (maneira antiga). Esta opção é para propósitos de depuração, se o **udev** não funcionar corretamente.

--noudevsync

Desabilita a sincronização do **udev**. Isto também serve para propósitos de depuração.

Para mais informações sobre o **dmsetup** e suas opções, veja a página man **dmsetup**(8).

Os comandos do LVM suportam as seguintes opções que suportam a integração do **udev**:

- **--noudevrules**: como para o comando **dmsetup**, desabilita as regras **udev**.
- **--noudevsync**: como para o comando **dmsetup** desabilita a sincronização do **udev**.

O arquivo **lvm.conf** inclui as seguintes opções que suportam a integração do **udev**:

- **udev_rules**: habilita/desabilita **udev_rules** para todos os comandos LVM2 globalmente.
- **udev_sync**: habilita/desabilita a sincronização do **udev** para todos os comandos do LVM globalmente.

Para mais informações sobre as opções de arquivo do **lvm.conf**, veja os comentários no arquivo **lvm.conf**.

APÊNDICE B. OS ARQUIVOS DE CONFIGURAÇÃO DO LVM

O LVM suporta múltiplos arquivos de configuração. Durante a inicialização do sistema, o arquivo de configuração do **lvm.conf** é carregado a partir do diretório especificado pela variável de ambiente **LVM_SYSTEM_DIR**, que é definida para **/etc/lvm** por padrão.

O arquivo **lvm.conf** pode especificar arquivos de configuração adicionais para carregar. Configurações em arquivos recentes sobrescrevem configurações de arquivos anteriores. Para exibir as configurações em uso após carregar todos os arquivos de configuração, execute o comando **lvm dumpconfig**.

Para informações sobre carregar arquivos de configurações adicionais, veja a Seção C.2, “Tags de Host”.

B.1. OS ARQUIVOS DE CONFIGURAÇÃO DO LVM

Os seguintes arquivos são usados para a configuração do LVM:

/etc/lvm/lvm.conf

O arquivo de configuração central lido pelas ferramentas.

etc/lvm/lvm_*hosttag*.conf

Para cada tag de host, um arquivo extra de configuração é lido, caso exista um: **lvm_*hosttag*.conf**. Se este arquivo define novas tags, então outros arquivos de configuração serão anexados à lista de títulos a serem lidos. Para informações sobre as tags de host, veja a Seção C.2, “Tags de Host”.

Além dos arquivos de configuração do LVM, um sistema executando o LVM inclui os seguintes arquivos que afetam o sistema de configuração do LVM:

/etc/lvm/.cache

Arquivo do cache do filtro de nome do dispositivo (configurável).

/etc/lvm/backup/

O diretório para backups de metadados de grupo de volume automático (configurável).

/etc/lvm/archive/

O diretório para arquivos de metadados de grupo de volumes automáticos (configurável a respeito do caminho do diretório e profundidade de histórico de arquivo).

/var/lock/lvm/

Em uma configuração única de host, bloqueie arquivos para prevenir que ferramentas paralelas danifiquem os metadados. Em um cluster, o DLM em cluster é usado.

B.2. EXEMPLO DE ARQUIVO LVM.CONF

Segue um exemplo do arquivo de configuração **lvm.conf**. Seu arquivo de configuração pode ser diferente deste.

```
# This is an example configuration file for the LVM2 system.
# It contains the default settings that would be used if there was no
# /etc/lvm/lvm.conf file.
#
# Refer to 'man lvm.conf' for further information including the file
layout.
#
# To put this file in a different directory and override /etc/lvm set
# the environment variable LVM_SYSTEM_DIR before running the tools.


# This section allows you to configure which block devices should
# be used by the LVM system.
devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # If several entries in the scanned directories correspond to the
    # same block device and the tools need to display a name for device,
    # all the pathnames are matched against each item in the following
    # list of regular expressions in turn and the first match is used.
    # preferred_names = [ ]

    # Try to avoid using undescriptive /dev/dm-N names, if present.
    preferred_names = [ "^/dev/mpath/", "^/dev/mapper/mpath",
"^/dev/[hs]d" ]

    # A filter that tells LVM2 to only use a restricted set of devices.
    # The filter consists of an array of regular expressions.  These
    # expressions can be delimited by a character of your choice, and
    # prefixed with either an 'a' (for accept) or 'r' (for reject).
    # The first expression found to match a device name determines if
    # the device will be accepted or rejected (ignored).  Devices that
    # don't match any patterns are accepted.

    # Be careful if there there are symbolic links or multiple filesystem
    # entries for the same device as each name is checked separately
against
    # the list of patterns.  The effect is that if any name matches any
'a'
    # pattern, the device is accepted; otherwise if any name matches any
'r'
    # pattern it is rejected; otherwise it is accepted.

    # Don't have more than one filter line active at once: only one gets
used.

    # Run vgscan after you change this parameter to ensure that
    # the cache file gets regenerated (see below).
    # If it doesn't do what you expect, check the output of 'vgscan -
vvvv'.
```

```
    # By default we accept every block device:
    filter = [ "a/.*/" ]

    # Exclude the cdrom drive
    # filter = [ "r|/dev/cdrom|" ]

    # When testing I like to work with just loopback devices:
    # filter = [ "a/loop/", "r/.*/" ]

    # Or maybe all loops and ide drives except hdc:
    # filter =[ "a|loop|", "r|/dev/hdc|", "a|/dev/ide|", "r|.*|" ]

    # Use anchors if you want to be really specific
    # filter = [ "a|^/dev/hda8$|", "r/.*/" ]

    # The results of the filtering are cached on disk to avoid
    # rescanning dud devices (which can take a very long time).
    # By default this cache is stored in the /etc/lvm/cache directory
    # in a file called '.cache'.
    # It is safe to delete the contents: the tools regenerate it.
    # (The old setting 'cache' is still respected if neither of
    # these new ones is present.)
    cache_dir = "/etc/lvm/cache"
    cache_file_prefix = ""

    # You can turn off writing this cache file by setting this to 0.
    write_cache_state = 1

    # Advanced settings.

    # List of pairs of additional acceptable block device types found
    # in /proc/devices with maximum (non-zero) number of partitions.
    # types = [ "fd", 16 ]

    # If sysfs is mounted (2.6 kernels) restrict device scanning to
    # the block devices it believes are valid.
    # 1 enables; 0 disables.
    sysfs_scan = 1

    # By default, LVM2 will ignore devices used as components of
    # software RAID (md) devices by looking for md superblocks.
    # 1 enables; 0 disables.
    md_component_detection = 1

    # By default, if a PV is placed directly upon an md device, LVM2
    # will align its data blocks with the md device's stripe-width.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1

    # Default alignment of the start of a data area in MB.  If set to 0,
    # a value of 64KB will be used.  Set to 1 for 1MiB, 2 for 2MiB, etc.
    # default_data_alignment = 1

    # By default, the start of a PV's data area will be a multiple of
```

```
    # the 'minimum_io_size' or 'optimal_io_size' exposed in sysfs.
    # - minimum_io_size - the smallest request the device can perform
    #   w/o incurring a read-modify-write penalty (e.g. MD's chunk size)
    # - optimal_io_size - the device's preferred unit of receiving I/O
    #   (e.g. MD's stripe width)
    # minimum_io_size is used if optimal_io_size is undefined (0).
    # If md_chunk_alignment is enabled, that detects the optimal_io_size.
    # This setting takes precedence over md_chunk_alignment.
    # 1 enables; 0 disables.
    data_alignment_detection = 1

    # Alignment (in KB) of start of data area when creating a new PV.
    # md_chunk_alignment and data_alignment_detection are disabled if set.
    # Set to 0 for the default alignment (see: data_alignment_default)
    # or page size, if larger.
    data_alignment = 0

    # By default, the start of the PV's aligned data area will be shifted
by
    # the 'alignment_offset' exposed in sysfs.  This offset is often 0 but
    # may be non-zero; e.g.: certain 4KB sector drives that compensate for
    # windows partitioning will have an alignment_offset of 3584 bytes
    # (sector 7 is the lowest aligned logical block, the 4KB sectors start
    # at LBA -1, and consequently sector 63 is aligned on a 4KB boundary).
    # But note that pvcreate --dataalignmentoffset will skip this
detection.
    # 1 enables; 0 disables.
    data_alignment_offset_detection = 1

    # If, while scanning the system for PVs, LVM2 encounters a device-
mapper
    # device that has its I/O suspended, it waits for it to become
accessible.
    # Set this to 1 to skip such devices.  This should only be needed
    # in recovery situations.
    ignore_suspended_devices = 0

    # During each LVM operation errors received from each device are
counted.
    # If the counter of a particular device exceeds the limit set here, no
    # further I/O is sent to that device for the remainder of the
respective
    # operation. Setting the parameter to 0 disables the counters
altogether.
    disable_after_error_count = 0

    # Allow use of pvcreate --uuid without requiring --restorefile.
    require_restorefile_with_uuid = 1
}

# This section allows you to configure the way in which LVM selects
# free space for its Logical Volumes.
#allocation {
#    When searching for free space to extend an LV, the "cling"
#    allocation policy will choose space on the same PVs as the last
#    segment of the existing LV.  If there is insufficient space and a
```

```
#    list of tags is defined here, it will check whether any of them are
#    attached to the PVs concerned and then seek to match those PV tags
#    between existing extents and new extents.
#    Use the special tag "@*" as a wildcard to match any PV tag.
#
#    Example: LVs are mirrored between two sites within a single VG.
#    PVs are tagged with either @site1 or @site2 to indicate where
#    they are situated.
#
#    cling_tag_list = [ "@site1", "@site2" ]
#    cling_tag_list = [ "@*" ]
#}

# This section that allows you to configure the nature of the
# information that LVM2 reports.
log {

    # Controls the messages sent to stdout or stderr.
    # There are three levels of verbosity, 3 being the most verbose.
    verbose = 0

    # Should we send log messages through syslog?
    # 1 is yes; 0 is no.
    syslog = 1

    # Should we log error and debug messages to a file?
    # By default there is no log file.
    #file = "/var/log/lvm2.log"

    # Should we overwrite the log file each time the program is run?
    # By default we append.
    overwrite = 0

    # What level of log messages should we send to the log file and/or
syslog?
    # There are 6 syslog-like log levels currently in use - 2 to 7
inclusive.
    # 7 is the most verbose (LOG_DEBUG).
    level = 0

    # Format of output messages
    # Whether or not (1 or 0) to indent messages according to their
severity
    indent = 1

    # Whether or not (1 or 0) to display the command name on each line
output
    command_names = 0

    # A prefix to use before the message text (but after the command name,
    # if selected).  Default is two spaces, so you can see/grep the
severity
    # of each message.
    prefix = "  "

    # To make the messages look similar to the original LVM tools use:
```

```
    #   indent = 0
    #   command_names = 1
    #   prefix = " -- "

    # Set this if you want log messages during activation.
    # Don't use this in low memory situations (can deadlock).
    # activation = 0
}

# Configuration of metadata backups and archiving.  In LVM2 when we
# talk about a 'backup' we mean making a copy of the metadata for the
# *current* system.  The 'archive' contains old metadata configurations.
# Backups are stored in a human readeable text format.
backup {

    # Should we maintain a backup of the current metadata configuration ?
    # Use 1 for Yes; 0 for No.
    # Think very hard before turning this off!
    backup = 1

    # Where shall we keep it ?
    # Remember to back up this directory regularly!
    backup_dir = "/etc/lvm/backup"

    # Should we maintain an archive of old metadata configurations.
    # Use 1 for Yes; 0 for No.
    # On by default.  Think very hard before turning this off.
    archive = 1

    # Where should archived files go ?
    # Remember to back up this directory regularly!
    archive_dir = "/etc/lvm/archive"

    # What is the minimum number of archive files you wish to keep ?
    retain_min = 10

    # What is the minimum time you wish to keep an archive file for ?
    retain_days = 30
}

# Settings for the running LVM2 in shell (readline) mode.
shell {

    # Number of lines of history to store in ~/.lvm_history
    history_size = 100
}


# Miscellaneous global LVM2 settings
global {

    # The file creation mask for any files and directories created.
    # Interpreted as octal if the first digit is zero.
    umask = 077

    # Allow other users to read the files
```

```
    #umask = 022

    # Enabling test mode means that no changes to the on disk metadata
    # will be made.  Equivalent to having the -t option on every
    # command.  Defaults to off.
    test = 0

    # Default value for --units argument
    units = "h"

    # Since version 2.02.54, the tools distinguish between powers of
    # 1024 bytes (e.g. KiB, MiB, GiB) and powers of 1000 bytes (e.g.
    # KB, MB, GB).
    # If you have scripts that depend on the old behaviour, set this to 0
    # temporarily until you update them.
    si_unit_consistency = 1

    # Whether or not to communicate with the kernel device-mapper.
    # Set to 0 if you want to use the tools to manipulate LVM metadata
    # without activating any logical volumes.
    # If the device-mapper kernel driver is not present in your kernel
    # setting this to 0 should suppress the error messages.
    activation = 1

    # If we can't communicate with device-mapper, should we try running
    # the LVM1 tools?
    # This option only applies to 2.4 kernels and is provided to help you
    # switch between device-mapper kernels and LVM1 kernels.
    # The LVM1 tools need to be installed with .lvm1 suffices
    # e.g. vgscan.lvm1 and they will stop working after you start using
    # the new lvm2 on-disk metadata format.
    # The default value is set when the tools are built.
    # fallback_to_lvm1 = 0

    # The default metadata format that commands should use - "lvm1" or
"lvm2".
    # The command line override is -M1 or -M2.
    # Defaults to "lvm2".
    # format = "lvm2"

    # Location of proc filesystem
    proc = "/proc"

    # Type of locking to use. Defaults to local file-based locking (1).
    # Turn locking off by setting to 0 (dangerous: risks metadata
corruption
    # if LVM2 commands get run concurrently).
    # Type 2 uses the external shared library locking_library.
    # Type 3 uses built-in clustered locking.
    # Type 4 uses read-only locking which forbids any operations that
might
    # change metadata.
    locking_type = 1

    # Set to 0 to fail when a lock request cannot be satisfied
immediately.
```

```
    wait_for_locks = 1

    # If using external locking (type 2) and initialisation fails,
    # with this set to 1 an attempt will be made to use the built-in
    # clustered locking.
    # If you are using a customised locking_library you should set this to
0.
    fallback_to_clustered_locking = 1

    # If an attempt to initialise type 2 or type 3 locking failed, perhaps
    # because cluster components such as clvmd are not running, with this
set
    # to 1 an attempt will be made to use local file-based locking (type
1).
    # If this succeeds, only commands against local volume groups will
proceed.
    # Volume Groups marked as clustered will be ignored.
    fallback_to_local_locking = 1

    # Local non-LV directory that holds file-based locks while commands
are
    # in progress.  A directory like /tmp that may get wiped on reboot is
OK.
    locking_dir = "/var/lock/lvm"

    # Whenever there are competing read-only and read-write access
requests for
    # a volume group's metadata, instead of always granting the read-only
    # requests immediately, delay them to allow the read-write requests to
be
    # serviced.  Without this setting, write access may be stalled by a
high
    # volume of read-only requests.
    # NB. This option only affects locking_type = 1 viz. local file-based
    # locking.
    prioritise_write_locks = 1

    # Other entries can go here to allow you to load shared libraries
    # e.g. if support for LVM1 metadata was compiled as a shared library
use
    #   format_libraries = "liblvm2format1.so"
    # Full pathnames can be given.

    # Search this directory first for shared libraries.
    #   library_dir = "/lib"

    # The external locking library to load if locking_type is set to 2.
    #   locking_library = "liblvm2clusterlock.so"

    # Treat any internal errors as fatal errors, aborting the process that
    # encountered the internal error. Please only enable for debugging.
    abort_on_internal_errors = 0

    # If set to 1, no operations that change on-disk metadata will be
permitted.
    # Additionally, read-only commands that encounter metadata in need of
```

```
repair
    # will still be allowed to proceed exactly as if the repair had been
    # performed (except for the unchanged vg_seqno).
    # Inappropriate use could mess up your system, so seek advice first!
    metadata_read_only = 0
}

activation {
    # Set to 0 to disable udev synchronisation (if compiled into the
binaries).
    # Processes will not wait for notification from udev.
    # They will continue irrespective of any possible udev processing
    # in the background.  You should only use this if udev is not running
    # or has rules that ignore the devices LVM2 creates.
    # The command line argument --nodevsync takes precedence over this
setting.
    # If set to 1 when udev is not running, and there are LVM2 processes
    # waiting for udev, run 'dmsetup udevcomplete_all' manually to wake
them up.
    udev_sync = 1

    # Set to 0 to disable the udev rules installed by LVM2 (if built with
    # --enable-udev_rules). LVM2 will then manage the /dev nodes and
symlinks
    # for active logical volumes directly itself.
    # N.B. Manual intervention may be required if this setting is changed
    # while any logical volumes are active.
    udev_rules = 1

    # How to fill in missing stripes if activating an incomplete volume.
    # Using "error" will make inaccessible parts of the device return
    # I/O errors on access.  You can instead use a device path, in which
    # case, that device will be used to in place of missing stripes.
    # But note that using anything other than "error" with mirrored
    # or snapshotted volumes is likely to result in data corruption.
    missing_stripe_filler = "error"

    # How much stack (in KB) to reserve for use while devices suspended
    reserved_stack = 256

    # How much memory (in KB) to reserve for use while devices suspended
    reserved_memory = 8192

    # Nice value used while devices suspended
    process_priority = -18

    # If volume_list is defined, each LV is only activated if there is a
    # match against the list.
    #   "vgname" and "vgname/lvname" are matched exactly.
    #   "@tag" matches any tag set in the LV or VG.
    #   "@*" matches if any tag defined on the host is also set in the LV
or VG
    #
    # volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]

    # Size (in KB) of each copy operation when mirroring
```

```
    mirror_region_size = 512

    # Setting to use when there is no readahead value stored in the
metadata.
    #
    # "none" - Disable readahead.
    # "auto" - Use default value chosen by kernel.
    readahead = "auto"

    # 'mirror_image_fault_policy' and 'mirror_log_fault_policy' define
    # how a device failure affecting a mirror is handled.
    # A mirror is composed of mirror images (copies) and a log.
    # A disk log ensures that a mirror does not need to be re-synced
    # (all copies made the same) every time a machine reboots or crashes.
    #
    # In the event of a failure, the specified policy will be used to
determine
    # what happens. This applies to automatic repairs (when the mirror is
being
    # monitored by dmeventd) and to manual lvconvert --repair when
    # --use-policies is given.
    #
    # "remove" - Simply remove the faulty device and run without it.  If
    #            the log device fails, the mirror would convert to using
    #            an in-memory log.  This means the mirror will not
    #            remember its sync status across crashes/reboots and
    #            the entire mirror will be re-synced.  If a
    #            mirror image fails, the mirror will convert to a
    #            non-mirrored device if there is only one remaining good
    #            copy.
    #
    # "allocate" - Remove the faulty device and try to allocate space on
    #            a new device to be a replacement for the failed device.
    #            Using this policy for the log is fast and maintains the
    #            ability to remember sync state through crashes/reboots.
    #            Using this policy for a mirror device is slow, as it
    #            requires the mirror to resynchronize the devices, but it
    #            will preserve the mirror characteristic of the device.
    #            This policy acts like "remove" if no suitable device and
    #            space can be allocated for the replacement.
    #
    # "allocate_anywhere" - Not yet implemented. Useful to place the log
device
    #            temporarily on same physical volume as one of the mirror
    #            images. This policy is not recommended for mirror devices
    #            since it would break the redundant nature of the mirror.
This
    #            policy acts like "remove" if no suitable device and space
can
    #            be allocated for the replacement.

    mirror_log_fault_policy = "allocate"
    mirror_image_fault_policy = "remove"

    # 'snapshot_autoextend_threshold' and 'snapshot_autoextend_percent'
define
```

```
    # how to handle automatic snapshot extension. The former defines when
the
    # snapshot should be extended: when its space usage exceeds this many
    # percent. The latter defines how much extra space should be allocated
for
    # the snapshot, in percent of its current size.
    #
    # For example, if you set snapshot_autoextend_threshold to 70 and
    # snapshot_autoextend_percent to 20, whenever a snapshot exceeds 70%
usage,
    # it will be extended by another 20%. For a 1G snapshot, using up 700M
will
    # trigger a resize to 1.2G. When the usage exceeds 840M, the snapshot
will
    # be extended to 1.44G, and so on.
    #
    # Setting snapshot_autoextend_threshold to 100 disables automatic
    # extensions. The minimum value is 50 (A setting below 50 will be
treated
    # as 50).

    snapshot_autoextend_threshold = 100
    snapshot_autoextend_percent = 20

    # While activating devices, I/O to devices being (re)configured is
    # suspended, and as a precaution against deadlocks, LVM2 needs to pin
    # any memory it is using so it is not paged out.  Groups of pages that
    # are known not to be accessed during activation need not be pinned
    # into memory.  Each string listed in this setting is compared against
    # each line in /proc/self/maps, and the pages corresponding to any
    # lines that match are not pinned.  On some systems locale-archive was
    # found to make up over 80% of the memory used by the process.
    # mlock_filter = [ "locale/locale-archive", "gconv/gconv-
modules.cache" ]

    # Set to 1 to revert to the default behaviour prior to version 2.02.62
    # which used mlockall() to pin the whole process's memory while
activating
    # devices.
    use_mlockall = 0

    # Monitoring is enabled by default when activating logical volumes.
    # Set to 0 to disable monitoring or use the --ignoremonitoring option.
    monitoring = 1

    # When pvmove or lvconvert must wait for the kernel to finish
    # synchronising or merging data, they check and report progress
    # at intervals of this number of seconds.  The default is 15 seconds.
    # If this is set to 0 and there is only one thing to wait for, there
    # are no progress reports, but the process is awoken immediately the
    # operation is complete.
    polling_interval = 15
}


####################
```

```
# Advanced section #
####################

# Metadata settings
#
# metadata {
    # Default number of copies of metadata to hold on each PV.  0, 1 or 2.
    # You might want to override it from the command line with 0
    # when running pvcreate on new PVs which are to be added to large VGs.

    # pvmetadatacopies = 1

    # Default number of copies of metadata to maintain for each VG.
    # If set to a non-zero value, LVM automatically chooses which of
    # the available metadata areas to use to achieve the requested
    # number of copies of the VG metadata.  If you set a value larger
    # than the the total number of metadata areas available then
    # metadata is stored in them all.
    # The default value of 0 ("unmanaged") disables this automatic
    # management and allows you to control which metadata areas
    # are used at the individual PV level using 'pvchange
    # --metadataignore y/n'.

    # vgmetadatacopies = 0

    # Approximate default size of on-disk metadata areas in sectors.
    # You should increase this if you have large volume groups or
    # you want to retain a large on-disk history of your metadata changes.

    # pvmetadatasize = 255

    # List of directories holding live copies of text format metadata.
    # These directories must not be on logical volumes!
    # It's possible to use LVM2 with a couple of directories here,
    # preferably on different (non-LV) filesystems, and with no other
    # on-disk metadata (pvmetadatacopies = 0). Or this can be in
    # addition to on-disk metadata areas.
    # The feature was originally added to simplify testing and is not
    # supported under low memory situations - the machine could lock up.
    #
    # Never edit any files in these directories by hand unless you
    # you are absolutely sure you know what you are doing! Use
    # the supplied toolset to make changes (e.g. vgcfgrestore).

    # dirs = [ "/etc/lvm/metadata", "/mnt/disk2/lvm/metadata2" ]
#}

# Event daemon
#
dmeventd {
    # mirror_library is the library used when monitoring a mirror device.
    #
    # "libdevmapper-event-lvm2mirror.so" attempts to recover from
    # failures.  It removes failed devices from a volume group and
    # reconfigures a mirror as necessary. If no mirror library is
    # provided, mirrors are not monitored through dmeventd.
```

```
    mirror_library = "libdevmapper-event-lvm2mirror.so"

    # snapshot_library is the library used when monitoring a snapshot
device.
    #
    # "libdevmapper-event-lvm2snapshot.so" monitors the filling of
    # snapshots and emits a warning through syslog when the use of
    # the snapshot exceeds 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot is filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"

    # Full path of the dmeventd binary.
    #
    # executable = "/sbin/dmeventd"
}
```

APÊNDICE C. TAGS DE OBJETOS DO LVM

Uma tag do LVM é uma palavra que pode ser usada para agrupar objetos do LVM2 do mesmo tipo. As tags (marcações) podem ser anexadas à objetos tais como volumes físicos, grupos de volumes e volumes lógicos. As tags podem ser anexadas à hosts em uma configuração de cluster. Os snapshots não podem ser marcados (tagged).

As tags podem ser fornecidas em uma linha de comando no lugar de argumentos do PV, VG ou LV. As tags devem ser pré-fixadas com @ para evitar ambiguidade. Cada tag é expandida através da substituição de todos os objetos que possuem esta tag e que são do tipo esperado por sua posição na linha de comando.

A partir do lançamento do Red Hat Enterprise Linux 6.1, os rótulos LVM são segmentos de até 1024 caractéres (para lançamentos anteriores o límite máximo era 128 caractéres). As tags LVM não podem iniciar com um hífen.

Um rótulo válido pode consistir de uma variação limitada de somente caractéres. Para o lançamento do Red Hat Enterprise Linux 6.0, os caractéres permitidos são [A-Za-z0-9_+.-]. A partir do lançamento do Red Hat Enterprise Linux 6.1, a lista de caractéres permitidos foi extendida e os rótulos podem conter os caractéres "/", "=", "!", ":", "#", and "&".

Somente objetos em um grupo de volume podem ser marcados (tagged). Volumes físicos perdem suas tags caso sejam removidos de um grupo de volume. Isto acontece porque as tags são armazenadas como parte de um grupo de volume de metadados e são removidas quando um volume físico é removido. Snapshots não podem ser marcados.

O comando a seguir lista todos os volumes lógicos com as tags **database**.

```
lvs @database
```

C.1. ADICIONANDO E REMOVENDO TAGS DE OBJETOS

Para adicionar ou remover tags de volumes físicos, use a opção **--addtag** ou **--deltag** do comando **pvchange**.

Para adicionar ou remover tags de grupos de volumes, use a opção **--addtag** ou **--deltag** dos comandos **vgchange** ou **vgcreate**

Para adicionar ou remover marcações de volumes lógicos, use a opção **--addtag** ou **--deltag** dos comandos **lvchange** ou **lvcreate**.

Com o lançamento do Red Hat Enterprise Linux 6.1, você pode especificar múltiplos argumentos **--addtag** e **--deltag** dentro de um único comando **pvchange**, **vgchange** ou **lvchange**. Por exemplo, o seguinte comando apaga os rótulos **T9** e **T10** e adiciona os rótulos **T13** e **T14** grupo de volume **grant**.

```
vgchange --deltag T9 --deltag T10 --addtag T13 --addtag T14 grant
```

C.2. TAGS DE HOST

Em uma configuração de cluster, você pode definir as tags de host nos arquivos de configuração. Se você definir **hosttags = 1** na seção **tags**, uma tag de host é definida automaticamente usando o hostname da máquina. Isto permite que você utilize um arquivo de configuração comum que pode ser

replicado em todas as suas máquinas para que sejam cópias idênticas do arquivo, mas o comportamento pode diferir entre máquinas de acordo com o hostname.

Para informações sobre os arquivos de configuração, veja o Apêndice B, *Os arquivos de Configuração do LVM*.

Para cada marca de host, um arquivo de configuração extra é lido se ele existir: lvm_*hosttag*.conf. Se o arquivo definir novas tags, então arquivos de configuração futuros serão adicionados à lista de arquivos para serem lidos.

Por exemplo, a entrada a seguir no arquivo de configuração sempre define **tag1** e define **tag2** se o hostname é **host1**.

```
tags { tag1 { }  tag2 { host_list = ["host1"] } }
```

C.3. CONTROLANDO A ATIVAÇÃO COM TAGS

Você pode especificar no arquivo de configuração que somente certos volumes lógicos devem ser ativados naquele host. Por exemplo, a seguinte entrada age como um filtro para requisições de ativação (tais como **vgchange -ay**) e somente ativa o **vg1/lvol0** e qualquer volume lógico ou grupos de volume com a tag **database** no metadados naquele host.

```
activation { volume_list = ["vg1/lvol0", "@database" ] }
```

Existe uma combinação especial "@*" que causa uma coincidência somente se qualquer tag de metadados coincidir com qualquer tag de host naquela máquina.

Como um outro exemplo, considere uma situação onde cada máquina no cluster possui a seguinte entrada no arquivo de configuração:

```
tags { hosttags = 1 }
```

Se você quiser ativar **vg1/lvol2** somente no host **db2**, faça o seguinte:

1. Execute o **lvchange --addtag @db2 vg1/lvol2** de qualquer host no cluster.
2. Execute **lvchange -ay vg1/lvol2**.

Esta solução inclui o armazenamento de hostnames dentro dos metadados de grupo de volume.

APÊNDICE D. METADADOS DE GRUPO DE VOLUME LVM

Os detalhes de configuração de um grupo de volume são referidos como metadados. Por padrão, uma cópia idêntica dos metadados é mantida em todas as áreas de metadados em todos os volumes físicos dentro de um grupo de volume. Os metadados de grupo de volume LVM é pequeno e armazenado como ASCII.

Se um grupo de volume contém muitos volumes físicos, ter muitas cópias redundantes dos metadados é ineficiente. É possível criar um volume físico sem qualquer cópia de metadados usando a opção **--metadatacopies 0** do comando **pvcreate**. Uma vez que selecionou o número de cópias de metadados que o volume físico vai conter, você não pode mais mudar até o final. Selecionar 0 cópias pode resultar em atualizações mais rápidas em mudanças de configuração. Observe no entanto, que sempre todo grupo de volume deve conter ao menos um volume físico com uma área de metadados (a menos que você esteja usando as configurações avançadas que permitem que você armazene os metadados do grupo de volume em um sistema de arquivo). Se você pretende dividir o grupo de volume no futuro, todo grupo de volume precisa de ao menos uma cópia de metadados.

Os metadados do núcleo estão armazenados em ASCII. Uma área de metadados é um buffer circular. Os novos metadados são adicionados aos metadados antigos e depois o indicador ao inicio dele é atualizado.

Você pode especificar o tamanho da área de metadados com a opção **--metadatasize** do comando **pvcreate**. O tamanho padrão é muito pequeno para os grupos de volume com muitos volumes lógicos ou volumes físicos.

D.1. O RÓTULO DO VOLUME FÍSICO

Por padrão, o comando **pvcreate** coloca o rótulo de volume físico no 2o. setor de 512 bytes. Este rótulo pode ser opcionalmente colocado em qualquer um dos primeiros quatro setores, desde que as ferramentas de LVM que procuram por um rótulo de volume físico, verifiquem os primeiros 4 setores. O rótulo de volume físico começa com a série **LABELONE**.

O rótulo de volume físico contém:

- O Volume Físico UUID
- O tamanho do dispositivo de bloco em bytes
- A lista do NULL finalizada de locais de área de dados.
- Listas de NULL-terminated de locais de área de metadados.

As posições dos metadados são armazenadas como offset e tamanho (em bytes). Existe espaço no rótulo para 15 posições, mas as ferramentas do LVM atualmente utilizam 3: uma área de dados única mais até 2 áreas de metadados.

D.2. CONTEÚDO DOS METADADOS

Os metadados de grupo de volume contém:

- Informações sobre como e quando ele foi criado
- Informações sobre o grupo de volume:

Informações do grupo de volume contém:

- Nome e ID único
- O número da versão que é incrementado sempre que o metadado é atualizado
- Alguma propriedade: Leitura/Escrita? Redimensionável?
- Qualquer limite administrativo no número de volumes físicos e volumes lógicos que possa conter
- O tamanho da extensão (em unidades de setores que são definidos como 512 bytes)
- Uma lista desordenada de volumes físicos criando o grupo de volume, cada um com:
  - Seu UUID, usado para determinar o dispositivo de bloco contendo-o
  - Qualquer propriedade, tal como se o volume físico é alocável
  - O offset para o iníciar da primeira extensão dentro do volume físico (em setores)
  - O número de extensões
- Uma lista desordenada de volumes lógicos, cada uma consistindo de:
  - Uma lista ordenada de segmentos de volumes lógicos. Para cada segmento os metadados incluem um mapeamento aplicado à uma lista ordenada de segmentos de volumes físicos ou segmentos de volumes lógicos.

D.3. EXEMPLO DE METADADOS

A seguir um exemplo de metadados do grupo de volume LVM para um grupo de volume chamado **myvg**.

```
# Generated by LVM2: Tue Jan 30 16:28:15 2007

contents = "Text Format Volume Group"
version = 1

description = "Created *before* executing 'lvextend -L+5G /dev/myvg/mylv
/dev/sdc'"

creation_host = "tng3-1"        # Linux tng3-1 2.6.18-8.el5 #1 SMP Fri Jan
26 14:15:21 EST 2007 i686
creation_time = 1170196095      # Tue Jan 30 16:28:15 2007

myvg {
        id = "0zd3UT-wbYT-lDHq-lMPs-EjoE-0o18-wL28X4"
        seqno = 3
        status = ["RESIZEABLE", "READ", "WRITE"]
        extent_size = 8192              # 4 Megabytes
        max_lv = 0
        max_pv = 0

        physical_volumes {

                pv0 {
                        id = "ZBW5qW-dXF2-0bGw-ZCad-2RlV-phwu-1c1RFt"
                        device = "/dev/sda"     # Hint only
```

```
                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv1 {
                        id = "ZHEZJW-MR64-D3QM-Rv7V-Hxsa-zU24-wztY19"
                        device = "/dev/sdb"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv2 {
                        id = "wCoG4p-55Ui-9tbp-VTEA-jO6s-RAVx-UREW0G"
                        device = "/dev/sdc"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv3 {
                        id = "hGlUwi-zsBg-39FF-do88-pHxY-8XA2-9WKIiA"
                        device = "/dev/sdd"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }
        }
        logical_volumes {

                mylv {
                        id = "GhUYSF-qVM3-rzQo-a6D2-o0aV-LQet-Ur9OF9"
                        status = ["READ", "WRITE", "VISIBLE"]
                        segment_count = 2

                        segment1 {
                                start_extent = 0
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv0", 0
                                ]
                        }
                        segment2 {
```

```
                                start_extent = 1280
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv1", 0
                                ]
                        }
                }
        }
}
```

APÊNDICE E. HISTÓRICO DE REVISÃO

Revisão 1-5.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revisão 1-5 **2012-07-18** **Anthony Towns**

Rebuild for Publican 3.0

Revisão 2.0-1 **Thu May 19 2011** **Steven Levine**

Lançamento inicial do Red Hat Enterprise Linux 6.1

Resolves: #694619
Documenta novas políticas de alocação `cling` quando extender um volume lógico.

Resolves: #682649
Adiciona avisos sobre executar comandos de criação de múltiplos espelhos em sucessão em volumes clusterizados.

Resolves: #674100
Adiciona o exemplo de resultado do comando `dmsetup ls --tree`

Resolves: #694607
Documenta o suporte para incluir múltiplos argumentos --addtag e --deltag em uma única linha de comando.

Resolves: #694604
Documenta o suporte para lista de caracteres expandida em rótulos.

Resolves: #694611
Documenta suporte para espelhos distribuídos.

Resolves: #694616
Documenta suporte para snapshots de volumes espelhados.

Resolves: #694618
Documenta suporte para snapshots de volumes exclusivamente ativados em cluster.

Resolves: #682648
Documenta que quando uma perna de espelho é realocada, a log do espelho pode ser movido também.

Resolves: #661530
Atualiza o exemplo do `cluster.conf` para um que documenta os recursos atuais.

Resolves: #642400
Adiciona nota sobre o gerenciamento do log de cluster sendo mantido por um nodo em cluster com uma ID de cluster mais baixa.

Resolves: #663462
Remove referências atualizadas para o monitor da máquina virtual Xen.

Revisão 1.0-1 **Wed Nov 10 2010** **Steven Levine**

Lançamento inicial do Red Hat Enterprise Linux 6

ÍNDICE REMISSIVO

Símbolos

D

E

F

G

I

L

M

R

S

T

U

V