O Gerenciador de Volume vinum

Índice

1. Sinopse

Não importa o tipo de disco, sempre há problemas em potencial. Os discos podem ser muito pequenos, muito lentos ou pouco confiáveis para atender aos requisitos do sistema. Enquanto os discos estão ficando maiores, também ficam maiores os requisitos para armazenamento de dados. Geralmente, é necessário um sistema de arquivos maior que a capacidade de um disco. Várias soluções para esses problemas foram propostas e implementadas.

Um método é através do uso de vários discos, e às vezes discos redundantes. Além de suportar várias placas e controladoras para sistemas `RAID` (Redundant Array of Independent Disks), o sistema básico do FreeBSD inclui o gerenciador de volumes vinum, um driver de dispositivo de bloco que implementa discos virtuais e aborda esses três problemas. O vinum oferece mais flexibilidade, desempenho e confiabilidade do que o armazenamento em disco tradicional e implementa os modelos `RAID`-0, `RAID`-1 e `RAID`-5, tanto individualmente quanto combinados.

Este capítulo fornece uma visão geral dos possíveis problemas com o armazenamento em disco tradicional e uma introdução ao gerenciador de volumes vinum.

Começando com o FreeBSD 5, o vinum foi reescrito para se encaixar na Arquitetura GEOM, mantendo as idéias originais, a terminologia e os metadados no disco. Esta reescrita é chamada *gvinum* (para *GEOM vinum*). Enquanto este capítulo usa o termo vinum, qualquer invocação de comandos deve ser executada com o `gvinum`. O nome do módulo do kernel mudou do original vinum.ko para geom_vinum.ko, e todos os device nodes residem em /dev/gvinum em vez de /dev/vinum. A partir do FreeBSD 6, a implementação original do vinum não está mais disponível no código base.

2. Gargalos de Acesso

Sistemas modernos frequentemente precisam acessar dados de uma maneira altamente concorrente. Por exemplo, grandes servidores FTP ou HTTP podem manter milhares de sessões simultâneas e ter múltiplas conexões de 100 Mbit/s para o mundo externo, muito além da taxa de transferência sustentada da maioria dos discos.

As unidades de disco atuais podem transferir dados sequencialmente a até 70 MB/s, mas esse valor é de pouca importância em um ambiente em que muitos processos independentes acessam uma unidade e onde podem obter apenas uma fração desses valores. Nesses casos, é mais interessante visualizar o problema do ponto de vista do subsistema de disco. O parâmetro importante é a carga que uma transferência coloca no subsistema ou o tempo pelo qual uma transferência ocupa as unidades envolvidas na transferência.

Em qualquer transferência de disco, a unidade deve primeiro posicionar as cabeças, aguardar que o primeiro setor passe sob a cabeça de leitura e depois realizar a transferência. Essas ações podem ser consideradas atômicas, pois não faz sentido interrompê-las.

Considere uma transferência típica de cerca de 10 kB: a geração atual de discos de alto desempenho pode posicionar as cabeças em uma média de 3,5 ms. As unidades mais rápidas giram a 15.000 rpm, portanto a latência rotacional média (meia revolução) é de 2 ms. A 70 MB/s, a própria transferência leva cerca de 150 µs, quase nada em comparação com o tempo de posicionamento. Nesse caso, a taxa de transferência efetiva cai para pouco mais de 1 MB/s e é claramente altamente dependente do tamanho da transferência.

A solução tradicional e óbvia para esse gargalo é "mais eixos": em vez de usar um disco grande, use vários discos menores com o mesmo espaço de armazenamento agregado. Cada disco é capaz de se posicionar e transferir de forma independente, portanto, o rendimento efetivo aumenta em um fator próximo ao número de discos usados.

A melhoria real da taxa de transferência é menor que o número de discos envolvidos. Embora cada unidade seja capaz de transferir em paralelo, não há como garantir que as solicitações sejam distribuídas uniformemente pelas unidades. Inevitavelmente, a carga em uma unidade será maior que em outra.

A uniformidade da carga nos discos é fortemente dependente da maneira como os dados são compartilhados entre as unidades. Na discussão a seguir, é conveniente pensar no armazenamento em disco como um grande número de setores de dados que são endereçáveis por número, mais ou menos como as páginas de um livro. O método mais óbvio é dividir o disco virtual em grupos de setores consecutivos do tamanho dos discos físicos individuais e armazená-los dessa maneira, mais ou menos como pegar um livro grande e dividi-lo em seções menores. Esse método é chamado de *concatenação* e tem a vantagem de os discos não precisarem ter nenhum relacionamento de tamanho específico. Ele funciona bem quando o acesso ao disco virtual é distribuído uniformemente sobre seu espaço de endereço. Quando o acesso é concentrado em uma área menor, a melhoria é menos acentuada. Concatenated Organization ilustra a seqüência na qual as unidades de armazenamento são alocadas em uma organização concatenada.

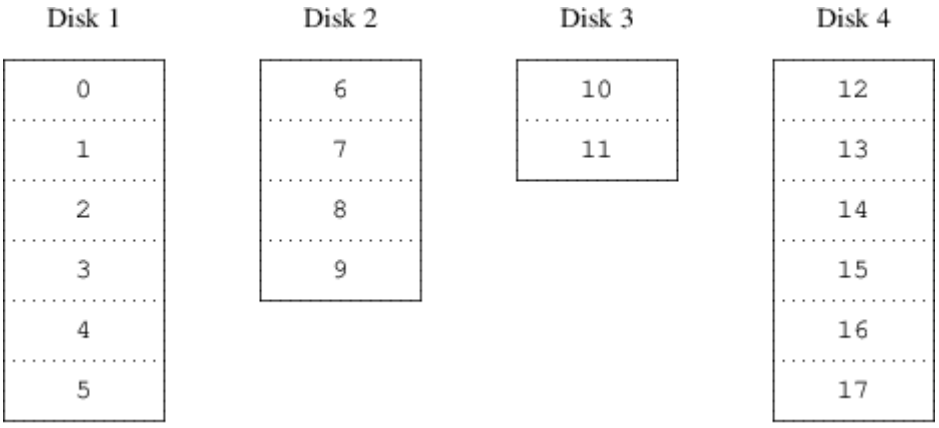

Figura 1. Organização Concatenada

Um mapeamento alternativo é dividir o espaço de endereço em componentes menores e de tamanhos iguais e armazená-los sequencialmente em diferentes dispositivos. Por exemplo, os primeiros 256 setores podem ser armazenados no primeiro disco, os próximos 256 setores no próximo disco e assim por diante. Depois de preencher o último disco, o processo é repetido até que os discos estejam cheios. Este mapeamento é chamado *striping* ou RAID-0.

O `RAID` oferece várias formas de tolerância a falhas, embora o RAID-0 seja um pouco enganador, pois não fornece redundância. O striping requer um pouco mais de esforço para localizar os dados e pode causar carga de I/O (INPUT/OUTPUT) adicional, onde uma transferência é distribuída por vários discos, mas também pode fornecer uma carga mais constante nos discos. Striped Organization ilustra a seqüência na qual as unidades de armazenamento são alocadas em uma organização distribuída.

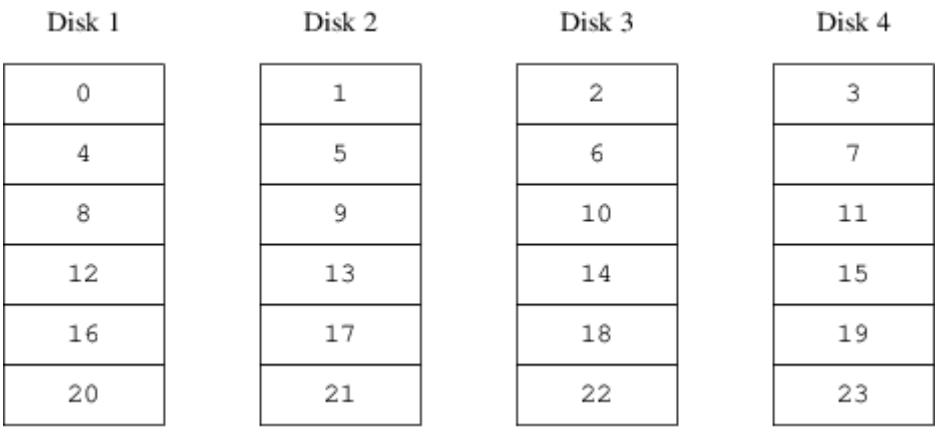

Figura 2. Organização do modo distribuido (Striped)

3. Integridade de dados

O problema final com os discos é que eles não são confiáveis. Embora a confiabilidade tenha aumentado tremendamente nos últimos anos, as unidades de disco ainda são o componente central mais provável de um servidor para falhar. Quando o fazem, os resultados podem ser catastróficos e substituir uma unidade de disco com falha e a restauração de dados pode resultar em tempo de inatividade do servidor.

Uma abordagem para esse problema é o *mirroring (espelhamento)*, ou RAID-1, que mantém duas cópias dos dados em diferentes hardwares físicos. Qualquer gravação no volume grava em ambos os discos; uma leitura pode ser satisfeita de qualquer um, portanto, se uma unidade falhar, os

dados ainda estarão disponíveis na outra unidade.

O mirroring tem dois problemas:

- Requer o dobro de armazenamento em disco que uma solução não redundante.
- As gravações devem ser executadas em ambas as unidades, então ela usa o dobro da largura de banda de um volume não espelhado. As leituras não sofrem uma penalidade de desempenho e podem até ser mais rápidas.

Uma solução alternativa é a *parity (paridade)*, implementada nos níveis RAID 2, 3, 4 e 5. Destes, o RAID-5 é o mais interessante. Como implementado no vinum, é uma variante em uma organização striped que dedica um bloco de cada distribuição à paridade de um dos outros blocos. Como implementado por vinum, um plex RAID-5 é semelhante a um plex striped, exceto que ele implementa RAID-5 incluindo um bloco de paridade em cada stripe. Conforme exigido pelo RAID-5, o local desse bloco de paridade muda de um stripe para o próximo. Os números nos blocos de dados indicam os números de blocos relativos.

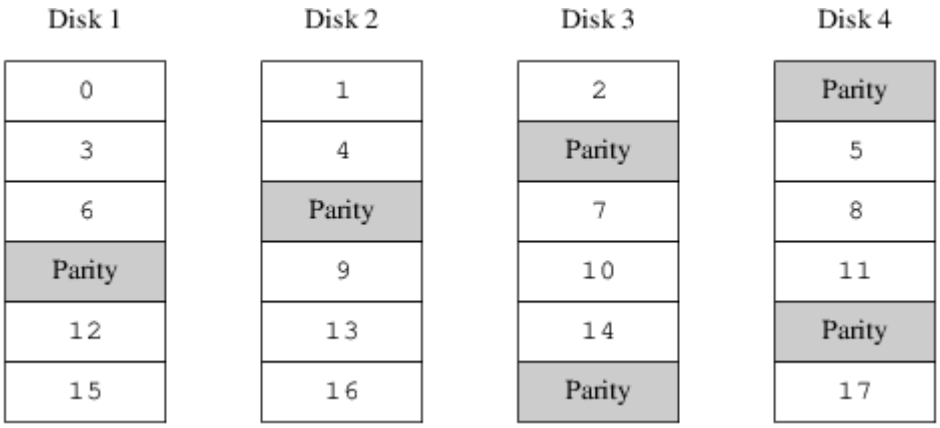

Figura 3. Organização RAID-5

Comparado ao mirroring, o RAID-5 tem a vantagem de exigir significativamente menos espaço de armazenamento. O acesso de leitura é semelhante ao das organizações distribuídas, mas o acesso de gravação é significativamente mais lento, aproximadamente 25% do desempenho de leitura. Se uma unidade falhar, a matriz pode continuar a operar no modo degradado, onde uma leitura de uma das unidades acessíveis restantes continua normalmente, mas uma leitura da unidade com falha é recalculada a partir do bloco correspondente de todas as unidades restantes.

4. Objetos do vinum

A fim de resolver estes problemas, o vinum implementa uma hierarquia de quatro níveis de objetos:

- O objeto mais visível é o disco virtual, chamado *volume*. Os volumes têm essencialmente as mesmas propriedades de uma unidade de disco UNIX®, embora haja algumas pequenas diferenças. Por um lado, eles não têm limitações de tamanho.
- Os volumes são compostos de *plexes*, cada um dos quais representa o espaço de endereço total de um volume. Este nível na hierarquia fornece redundância. Pense em plexes como discos individuais em uma matriz espelhada, cada um contendo os mesmos dados.

- Como o vinum existe dentro do framework de armazenamento em disco UNIX®, seria possível usar as partições UNIX® como bloco de construção para plexes de vários discos. Na verdade, isso acaba sendo muito inflexível, pois os discos UNIX® podem ter apenas um número limitado de partições. Em vez disso, o vinum subdivide uma única partição UNIX®, a *unidade*, em áreas contíguas chamadas *subdiscos* , que são usados como blocos de construção para plexes.
- Subdiscos residem em vinum *drives*, atualmente partições UNIX®. Unidades vinum podem conter qualquer número de subdiscos. Com exceção de uma pequena área no início da unidade, que é usada para armazenar informações de configuração e estado, a unidade inteira está disponível para armazenamento de dados.

As seções a seguir descrevem a maneira como esses objetos fornecem a funcionalidade necessária do vinum.

4.1. Considerações sobre o tamanho do volume

Os plexes podem incluir vários subdiscos distribuídos por todas as unidades na configuração vinum. Como resultado, o tamanho de uma unidade individual não limita o tamanho de um plex ou de um volume.

4.2. Armazenamento de Dados Redundantes

O vinum implementa o espelhamento anexando vários plexes a um volume. Cada plex é uma representação dos dados em um volume. Um volume pode conter entre um e oito plexes.

Embora um plex represente os dados completos de um volume, é possível que partes da representação estejam fisicamente ausentes, seja por design (por não definir um subdisco para partes do plex) ou por acidente (como resultado da falha de representação). Contanto que pelo menos um plex possa fornecer os dados para o intervalo de endereços completo do volume, o volume estará totalmente funcional.

4.3. Quais são as organizações disponíveis para um Plex?

O vinum implementa a concatenação e o striping no nível plex:

- Um *plex concatenado* usa o espaço de endereço de cada subdisco um de cada vez. Plexes concatenados são os mais flexíveis, pois podem conter qualquer número de subdiscos e os subdiscos podem ser de tamanho diferente. O plex pode ser estendido adicionando subdiscos adicionais. Eles exigem menos tempo de CPU do que os plexes distribuídos, embora a diferença na sobrecarga de CPU não seja mensurável. Por outro lado, eles são mais suscetíveis a hot spots, nos quais um disco é muito ativo e outros ficam ociosos.
- Um *plex striped* distribui os dados uniformemente entre cada subdisco. Os subdiscos devem ser todos do mesmo tamanho e deve haver pelo menos dois subdiscos para distingui-los de um plex concatenado. A maior vantagem dos plexes striped é que eles reduzem os hot spots. Ao escolher uma faixa de tamanho ideal, de cerca de 256 kB, a carga pode ser nivelada nas unidades de componentes. Estender um complexo adicionando novos subdiscos é algo tão complicado que o

vinum não o implementa.

vinum Plex Organizations resume as vantagens e desvantagens de cada organização plex.

Tabela 1. Organizações Plex do vinum

Tipo plex	Subdiscos mínimos	Pode adicionar subdiscos	Deve ser de tamanho igual	Aplicação
concatenado	1	sim	não	Armazenamento de dados grandes com flexibilidade máxima de posicionamento e desempenho moderado
striped	2	não	sim	Alto desempenho em combinação com acesso altamente concorrente

5. Alguns exemplos

O vinum mantém um *banco de dados de configuração* que descreve os objetos conhecidos de um sistema individual. Inicialmente, o usuário cria o banco de dados de configuração a partir de um ou mais arquivos de configuração usando gvinum(8). O vinum armazena uma cópia de seu banco de dados de configuração em cada *dispositivo* de disco sob seu controle. Este banco de dados é atualizado em cada mudança de estado, de modo que uma reinicialização restaura com precisão o estado de cada objeto vinum.

5.1. O arquivo de configuração

O arquivo de configuração descreve objetos vinum individuais. A definição de um volume simples pode ser:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

Este arquivo descreve quatro objetos vinum:

- A linha *drive* descreve uma partição de disco (*drive*) e sua localização relativa ao hardware subjacente. É dado o nome simbólico *a*. Essa separação de nomes simbólicos de nomes de dispositivos permite que os discos sejam movidos de um local para outro sem confusão.
- A linha *volume* descreve um volume. O único atributo obrigatório é o nome, neste caso *myvol*.

- A linha *plex* define um plex. O único parâmetro requerido é a organização, neste caso *concat*. Nenhum nome é necessário, pois o sistema gera automaticamente um nome a partir do nome do volume, adicionando o sufixo *.px*, onde *x* é o número de o plex no volume. Assim, este plex será chamado *myvol.p0*.
- A linha *sd* descreve um subdisco. As especificações mínimas são o nome de uma unidade na qual irá armazená-lo e o tamanho do subdisco. Nenhum nome é necessário porque o sistema atribui automaticamente nomes derivados do nome do plex adicionando o sufixo *.sx*, onde *x* é o número do subdisco no plex. Assim, vinum dá ao subdisco o nome *myvol.p0.s0*.

Depois de processar este arquivo, o gvinum(8) produz a seguinte saída:

```
# gvinum -> create config1
Configuration summary
Drives:         1 (4 configured)
Volumes:        1 (4 configured)
Plexes:         1 (8 configured)
Subdisks:       1 (16 configured)

  D a                     State: up       Device /dev/da3h      Avail: 2061/2573 MB
(80%)

  V myvol                 State: up       Plexes:       1 Size:      512 MB

  P myvol.p0            C State: up       Subdisks:     1 Size:      512 MB

  S myvol.p0.s0           State: up       PO:        0  B Size:      512 MB
```

Esta saída mostra o formato de listagem breve de gvinum(8). Ele está representado graficamente em A Simple vinum Volume.

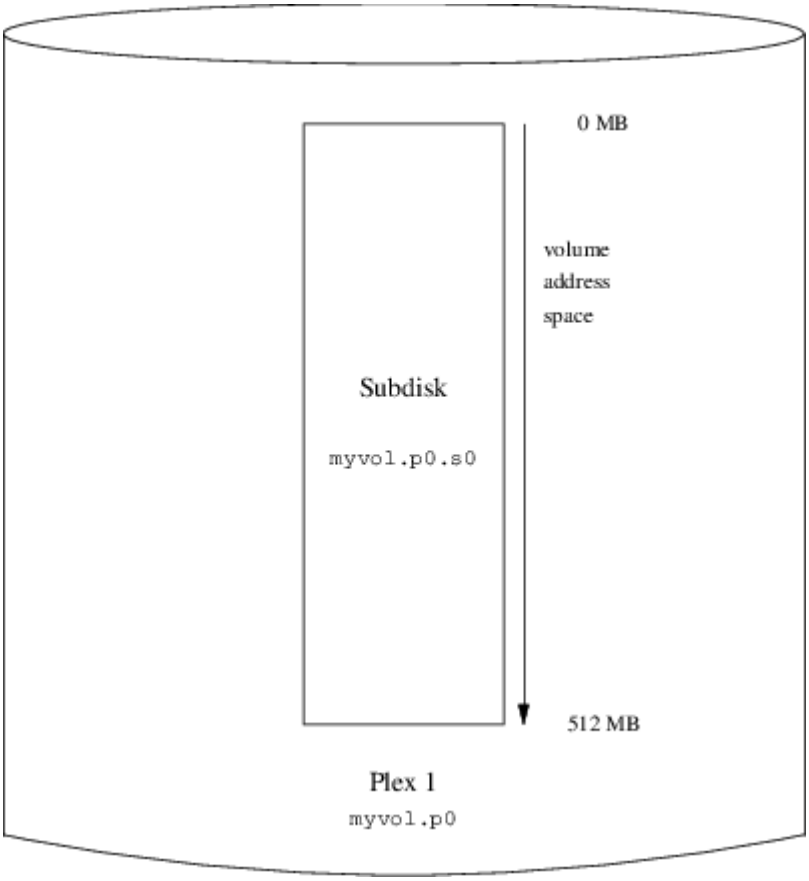

Figura 4. Um volume vinum simples

Esta figura, e as que se seguem, representam um volume, que contém os plexes, que por sua vez contém os subdiscos. Neste exemplo, o volume contém um plex e o plex contém um subdisco.

Este volume específico não tem nenhuma vantagem específica sobre uma partição de disco convencional. Ele contém um único plex, por isso não é redundante. O plex contém um único subdisco, portanto, não há diferença na alocação de armazenamento de uma partição de disco convencional. As seções a seguir ilustram vários métodos de configuração mais interessantes.

5.2. Maior Resiliência: Espelhamento

A resiliência de um volume pode ser aumentada pelo espelhamento. Ao dispor um volume espelhado, é importante garantir que os subdiscos de cada plex estejam em unidades diferentes, de modo que uma falha no dispositivo não derrubará os dois plexes. A configuração a seguir espelha um volume:

```
drive b device /dev/da4h
volume mirror
  plex org concat
    sd length 512m drive a
  plex org concat
    sd length 512m drive b
```

Neste exemplo, não foi necessário especificar uma definição de drive *a* novamente, já que o vinum registra todos os objetos em seu banco de dados de configuração. Depois de processar esta

definição, a configuração se parece com:

```
    Drives:         2 (4 configured)
    Volumes:        2 (4 configured)
    Plexes:         3 (8 configured)
    Subdisks:       3 (16 configured)

    D a                     State: up       Device /dev/da3h       Avail: 1549/2573 MB
(60%)
    D b                     State: up       Device /dev/da4h       Avail: 2061/2573 MB
(80%)

    V myvol                 State: up       Plexes:       1 Size:        512 MB
    V mirror                State: up       Plexes:       2 Size:        512 MB

    P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p1           C State: initializing     Subdisks:     1 Size:        512
MB

    S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
    S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
    S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
```

A Mirrored vinum Volume mostra a estrutura graficamente.

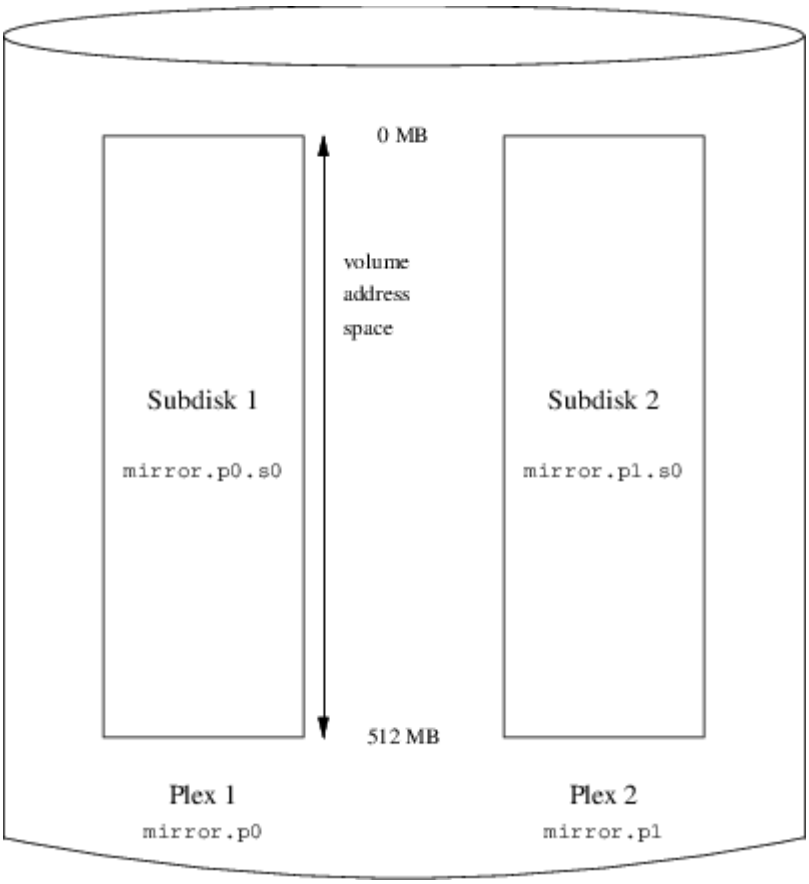

Figura 5. Um volume vinum espelhado

Neste exemplo, cada plex contém os 512 MB completos do espaço de endereço. Como no exemplo anterior, cada plex contém apenas um único subdisco.

5.3. Otimizando o desempenho

O volume espelhado no exemplo anterior é mais resistente a falhas do que um volume não espelhado, mas seu desempenho é menor, pois cada gravação no volume requer uma gravação nas duas unidades, utilizando uma grande parte da largura de banda total do disco. As considerações de desempenho exigem uma abordagem diferente: em vez de espelhar, os dados são distribuídos em quantas unidades de disco forem possíveis. A configuração a seguir mostra um volume com um plex distribuído em quatro unidades de disco:

```
    drive c device /dev/da5h
    drive d device /dev/da6h
    volume stripe
    plex org striped 512k
      sd length 128m drive a
      sd length 128m drive b
      sd length 128m drive c
      sd length 128m drive d
```

Como antes, não é necessário definir as unidades que já são conhecidas por vinum. Depois de processar esta definição, a configuração se parece com:

```
    Drives:         4 (4 configured)
    Volumes:        3 (4 configured)
    Plexes:         4 (8 configured)
    Subdisks:       7 (16 configured)

    D a                     State: up       Device /dev/da3h        Avail: 1421/2573
MB (55%)
    D b                     State: up       Device /dev/da4h        Avail: 1933/2573
MB (75%)
    D c                     State: up       Device /dev/da5h        Avail: 2445/2573
MB (95%)
    D d                     State: up       Device /dev/da6h        Avail: 2445/2573
MB (95%)

    V myvol                 State: up       Plexes:       1 Size:        512 MB
    V mirror                State: up       Plexes:       2 Size:        512 MB
    V striped               State: up       Plexes:       1 Size:        512 MB

    P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p1           C State: initializing     Subdisks:     1 Size:        512
MB
    P striped.p1            State: up       Subdisks:     1 Size:        512 MB
```

```
S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
S striped.p0.s0         State: up       PO:        0  B Size:        128 MB
S striped.p0.s1         State: up       PO:      512 kB Size:        128 MB
S striped.p0.s2         State: up       PO:     1024 kB Size:        128 MB
S striped.p0.s3         State: up       PO:     1536 kB Size:        128 MB
```

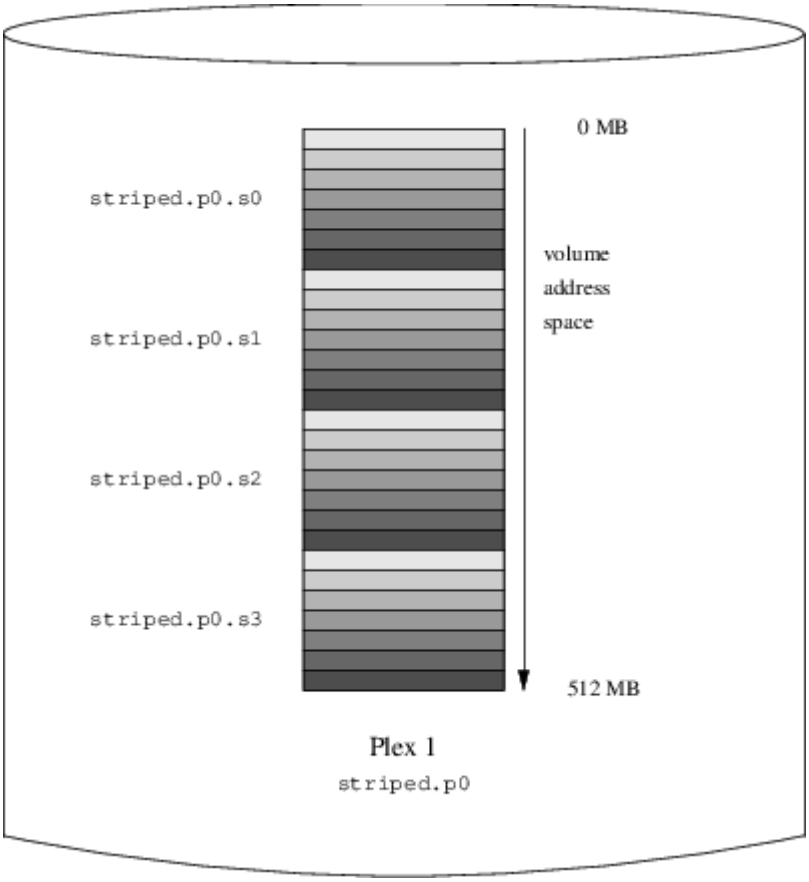

Figura 6. Um volume vinum concatenado

Este volume é representado em A Striped vinum Volume. A escuridão das strips indica a posição dentro do espaço de endereço plex, onde as faixas mais claras vêm primeiro e as mais escuras por último.

5.4. Resiliência e Desempenho

Com hardware suficiente, é possível construir volumes que mostrem maior resiliência e melhor desempenho em comparação com as partições padrão UNIX®. Um arquivo de configuração típico pode ser:

```
volume raid10
  plex org striped 512k
    sd length 102480k drive a
    sd length 102480k drive b
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
```

```
      plex org striped 512k
        sd length 102480k drive c
        sd length 102480k drive d
        sd length 102480k drive e
        sd length 102480k drive a
        sd length 102480k drive b
```

Os subdiscos do segundo plex são compensados por duas unidades daquelas do primeiro plex. Isso ajuda a garantir que as gravações não vão para os mesmos subdiscos, mesmo que uma transferência passe por duas unidades.

A Mirrored, Striped vinum Volume representa a estrutura deste volume.

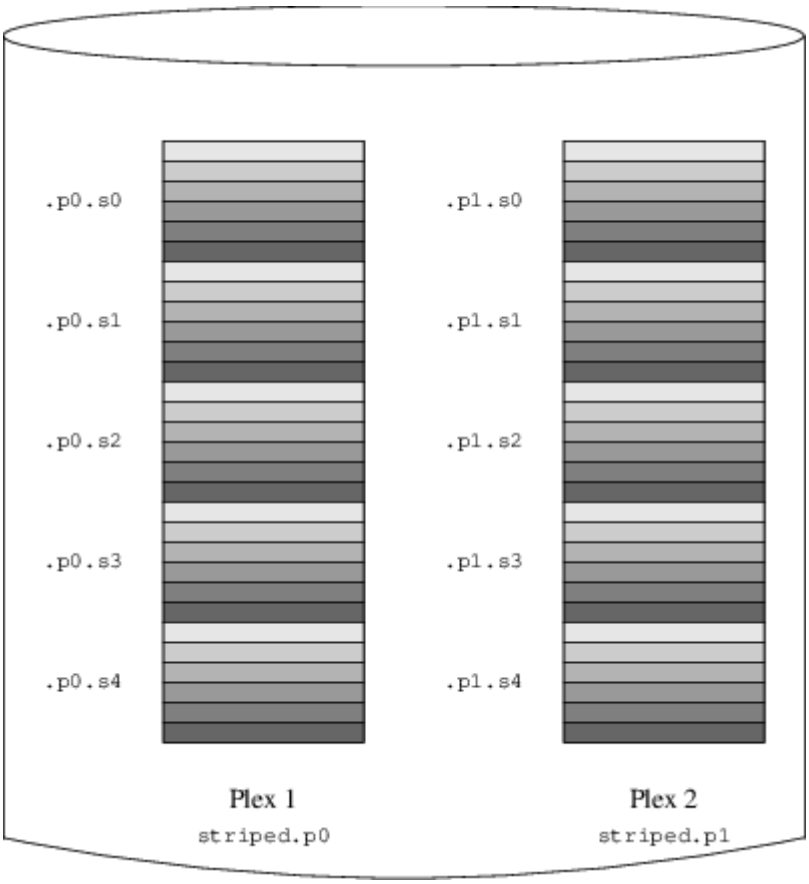

Figura 7. Um volume vinum espelhado e concatenado

6. Nomeação de Objetos

O vinum atribui nomes padrões a plexes e subdiscos, embora eles possam ser substituídos. Substituir os nomes padrões não é recomendado, pois não isso traz nenhuma vantagem significativa e pode causar confusão.

Os nomes podem conter qualquer caractere não-branco, mas é recomendado restringi-los a letras, dígitos e caracteres de sublinhado. Os nomes de volumes, plexes e subdiscos podem ter até 64 caracteres e os nomes das unidades podem ter até 32 caracteres.

Os objetos vinum são designados a device nodes na hierarquia /dev/gvinum. A configuração mostrada acima faria com que o vinum criasse os seguintes device nodes:

- Entradas de dispositivos para cada volume. Estes são os principais dispositivos usados pelo vinum. A configuração acima incluiria os dispositivos /dev/gvinum/myvol, /dev/gvinum/mirror, /dev/gvinum/striped, /dev/gvinum/raid5 e o /dev/gvinum/raid10.
- Todos os volumes recebem entradas diretas em /dev/gvinum/.
- Os diretórios /dev/gvinum/plex, e /dev/gvinum/sd são aqueles que contém device nodes para cada plex e para cada subdisco, respectivamente.

Por exemplo, considere o seguinte arquivo de configuração:

```
    drive drive1 device /dev/sd1h
    drive drive2 device /dev/sd2h
    drive drive3 device /dev/sd3h
    drive drive4 device /dev/sd4h
    volume s64 setupstate
      plex org striped 64k
        sd length 100m drive drive1
        sd length 100m drive drive2
        sd length 100m drive drive3
        sd length 100m drive drive4
```

Depois de processar este arquivo, o gvinum(8) cria a seguinte estrutura em /dev/gvinum:

```
    drwxr-xr-x  2 root  wheel       512 Apr 13
16:46 plex
    crwxr-xr--  1 root  wheel   91,   2 Apr 13 16:46 s64
    drwxr-xr-x  2 root  wheel       512 Apr 13 16:46 sd

    /dev/vinum/plex:
    total 0
    crwxr-xr--  1 root  wheel   25, 0x10000002 Apr 13 16:46 s64.p0

    /dev/vinum/sd:
    total 0
    crwxr-xr--  1 root  wheel   91, 0x20000002 Apr 13 16:46 s64.p0.s0
    crwxr-xr--  1 root  wheel   91, 0x20100002 Apr 13 16:46 s64.p0.s1
    crwxr-xr--  1 root  wheel   91, 0x20200002 Apr 13 16:46 s64.p0.s2
    crwxr-xr--  1 root  wheel   91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

Embora seja recomendado que os plexes e subdiscos não sejam atribuídos a nomes específicos, as unidades vinum devem ser nomeadas. Isso possibilita mover uma unidade para um local diferente e ainda reconhecê-la automaticamente. Os nomes dos drives podem ter até 32 caracteres.

6.1. Criando sistemas de arquivos

Para o sistema os volumes são idênticos aos discos, com uma exceção. Ao contrário das unidades UNIX®, o vinum não particiona os volumes, e, portanto, não contêm uma tabela de partições. Isso exigiu modificação em alguns dos utilitários de disco, notavelmente no newfs(8), para que ele não

tente interpretar a última letra de um nome do volume vinum como um identificador de partição. Por exemplo, uma unidade de disco pode ter um nome como /dev/ad0a ou /dev/da2h. Esses nomes representam a primeira partição (a) no primeiro (0) disco IDE (ad) e a oitava partição (h) no terceiro (2) disco SCSI (da) respectivamente. Por outro lado, um volume vinum pode ser chamado de /dev/gvinum/concat, que não tem relação com o nome da partição.

Para criar um sistema de arquivos neste volume, use newfs(8):

```
# newfs /dev/gvinum/concat
```

7. Configurando o vinum

O kernel GENERIC não suporta o vinum. É possível compilar um kernel customizado que inclua o suporte estático ao vinum, mas isso não é recomendado. A maneira padrão de iniciar o vinum é como um módulo do kernel. O uso do kldload(8) não é necessário porque quando o gvinum(8) é iniciado, ele verifica se o módulo já foi carregado e, se ele não tiver sido, ele será carregado automaticamente.

7.1. Começando

O vinum armazena as informações de configuração nos slices dos discos essencialmente da mesma forma que nos arquivos de configuração. Ao ler a partir do banco de dados de configuração, o vinum reconhece um número de palavras-chave que não são permitidas nos arquivos de configuração. Por exemplo, uma configuração de disco pode conter o seguinte texto:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1572864b
```

```
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 16777216b
```

As diferenças óbvias aqui são a presença de informações explícitas de localização e nomeação, as quais são ambas permitidas mas desencorajadas, e as informações sobre os estados. O vinum não armazena informações sobre unidades nas informações de configuração. Ele encontra as unidades varrendo as unidades de disco configuradas em busca de partições com um rótulo vinum. Isso permite que o vinum identifique as unidades corretamente, mesmo que elas tenham recebido diferentes IDs de unidade UNIX®.

7.1.1. Inicialização automática

O *Gvinum* apresenta sempre uma inicialização automática assim que o módulo do kernel é carregado, através do loader.conf(5). Para carregar o módulo *Gvinum* no momento da inicialização, adicione `geom_vinum_load="YES"` ao arquivo /boot/loader.conf.

Quando o vinum é iniciado com `gvinum start`, o vinum lê o banco de dados de configuração de uma das unidades vinum. Em circunstâncias normais, cada unidade contém uma cópia idêntica do banco de dados de configuração, portanto, não importa qual unidade é lida. Após uma falha, no entanto, o vinum deve determinar qual unidade foi atualizada mais recentemente e ler a configuração desta unidade. Em seguida, atualiza a configuração, se necessário, de unidades progressivamente a partir das mais antigas.

8. Usando o vinum para o sistema de arquivos raiz

Para uma máquina que tenha sistemas de arquivos totalmente espelhados usando vinum, é desejável também espelhar o sistema de arquivos raiz. Efetuar esta configuração é menos trivial do que espelhar um sistema de arquivos arbitrário porque:

- O sistema de arquivos raiz deve estar disponível muito cedo durante o processo de inicialização, portanto a infraestrutura vinum já deve estar disponível no momento.
- O volume que contém o sistema de arquivos raiz também contém a auto-inicialização do sistema e o kernel. Eles devem ser lidos usando os utilitários nativos do sistema, como o BIOS, que muitas vezes não pode ser instruído sobre os detalhes do vinum.

Nas seções a seguir, o termo "volume raiz" é geralmente usado para descrever o volume vinum que contém o sistema de arquivos raiz (/).

8.1. Iniciando o vinum cedo o suficiente para o sistema de arquivos raiz

O vinum deve estar disponível no início da inicialização do sistema pois o loader(8) deve ser capaz de carregar o módulo do kernel vinum antes de iniciar o kernel. Isto pode ser feito colocando esta linha no /boot/loader.conf:

```
geom_vinum_load="YES"
```

8.2. Tornando um volume raiz baseado em vinum acessível ao Bootstrap

O bootstrap atual do FreeBSD tem apenas 7.5 KB de código e não entende as estruturas internas do vinum. Isso significa que não é possível analisar os dados de configuração vinum ou descobrir os elementos de um volume de inicialização. Assim, algumas soluções alternativas são necessárias para fornecer ao código de inicialização a ilusão de que ele está trabalhando com uma partição padrão `a` que contém o sistema de arquivos raiz.

Para que isso seja possível, os seguintes requisitos devem ser atendidos para o volume raiz:

- O volume raiz não pode ser uma stripe ou `RAID` -5.
- O volume raiz não deve conter mais de um subdisco concatenado por plex.

Observe que é desejável e possível usar vários plexes, cada um contendo uma réplica do sistema de arquivos raiz. O processo de bootstrap usará apenas uma réplica para localizar o bootstrap e todos os arquivos de inicialização, até que o kernel monte o sistema de arquivos raiz. Cada subdisco dentro desses plexes precisa da sua própria ilusão de partição `a`, para que o respectivo dispositivo seja inicializável. Não é estritamente necessário que cada uma dessas falsas partições `a` estejam localizadas no mesmo deslocamento dentro de seu dispositivo, em comparação com outros dispositivos contendo plexes do volume raiz. No entanto, é provavelmente uma boa ideia criar os volumes vinum dessa forma para que os dispositivos espelhados resultantes sejam simétricos, para evitar confusão.

Para configurar essas partições `a` para cada dispositivo contendo parte do volume raiz, é necessário o seguinte:

1. A localização, offset desde o início do dispositivo, e o tamanho do subdisco desse dispositivo que faz parte do volume raiz precisam ser examinados, usando o comando:

   ```
   # gvinum l -rv root
   ```

   Os offsets (deslocamentos) e tamanhos do vinum são medidos em bytes. Eles devem ser divididos por 512 para obter os números de blocos que serão usados pelo `bsdlabel`.

2. Execute este comando para cada dispositivo que participa do volume raiz:

```
# bsdlabel -e devname
```

No comando acima *devname* deve ser o nome do disco, como da0 para discos sem uma tabela de slices, ou o nome da slice, como ad0s1.

Se já existir uma partição `a` no dispositivo a partir de um sistema de arquivos raiz pré-vinum, ela deve ser renomeada para outra coisa para que permaneça acessível (apenas nesse caso), mas ela não será mais usada por padrão para inicializar o sistema. Um sistema de arquivos raiz atualmente montado não pode ser renomeado, portanto, de forma que o processo ser executado quando o sistema for inicializado a partir de uma mídia "Fixit" ou em um processo de duas etapas em que, em um espelho, o disco que ainda não foi inicializado é manipulado primeiro.

O offset da partição vinum neste dispositivo (se houver) deve ser adicionado ao deslocamento do respectivo subdisco de volume raiz neste dispositivo. O valor resultante se tornará o valor do `offset` para a nova partição `a`. O valor do `size` para esta partição também pode ser obtido a partir do cálculo acima. O `fstype` deve ser `4.2BSD`. Os valores de `fsize`, `bsize` e `cpg` devem ser escolhidos para corresponder ao sistema de arquivos atual, embora eles sejam relativamente sem importância dentro deste contexto.

Desta forma, uma nova partição `a` será estabelecida sobrepondo a partição vinum neste dispositivo. O `bsdlabel` só permitirá essa sobreposição se a partição vinum tiver sido marcada corretamente usando o modo fstype do `vinum`.

3. Temos agora uma falsa partição `a` em cada dispositivo que possui uma réplica do volume raiz. É altamente recomendável verificar o resultado usando um comando como:

```
# fsck -n /dev/devnamea
```

Deve ser lembrado que todos os arquivos contendo informações de controle devem ser relativos ao sistema de arquivos raiz no volume vinum e que, ao configurarmos um novo volume raiz vinum, ele pode não corresponder o sistema de arquivos raiz que está atualmente ativo. Então, em particular, o /etc/fstab e /boot/loader.conf precisam ser ajustados.

Na próxima reinicialização, o bootstrap deve descobrir as informações de controle apropriadas do novo sistema de arquivos raiz baseado no vinum e agir de acordo. No final do processo de inicialização do kernel, após todos os dispositivos terem sido anunciados, o aviso de destaque que mostra o sucesso desta configuração é uma mensagem como:

```
Mounting root from ufs:/dev/gvinum/root
```

8.3. Exemplo de uma configuração raiz baseada em vinum

Depois que o volume raiz vinum foi configurado, a saída de `gvinum l -rv root` pode parecer com:

```
...
Subdisk root.p0.s0:
        Size:        125829120 bytes (120 MB)
        State: up
        Plex root.p0 at offset 0 (0  B)
        Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
        Size:        125829120 bytes (120 MB)
        State: up
        Plex root.p1 at offset 0 (0  B)
        Drive disk1 (/dev/da1h) at offset 135680 (132 kB)
```

Os valores a serem observados são `135680` para o offset, relativo à partição /dev/da0h. Isso se traduz em 265 blocos de discos de 512 bytes nos termos do `bsdlabel`. Da mesma forma, o tamanho desse volume raiz é de 245760 blocos de 512 bytes. O /dev/da1h, contém a segunda réplica deste volume raiz, e possui uma configuração simétrica.

O bsdlabel para esses dispositivos pode se parecer com:

```
...
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:   245760      281    4.2BSD     2048 16384     0  # (Cyl.    0*- 15*)
  c: 71771688        0    unused        0     0        # (Cyl.    0 - 4467*)
  h: 71771672       16     vinum                      # (Cyl.    0*- 4467*)
```

Pode-se observar que o parâmetro `size` para a falsa partição `a` corresponde ao valor descrito acima, enquanto o parâmetro `offset` é a soma do deslocamento dentro da partição vinum`h`, e o offset desta partição dentro do dispositivo ou slice. Esta é uma configuração típica que é necessária para evitar o problema descrito em Nothing Boots, the Bootstrap Panics. A partição `a` inteira está completamente dentro da partição `h` que contém todos os dados vinum para este dispositivo.

No exemplo acima, todo o dispositivo é dedicado ao vinum e não há sobra de partição raiz pré-vinum.

8.4. Soluções de problemas

A lista a seguir contém algumas armadilhas e soluções conhecidas.

8.4.1. Sistema de bootstrap carrega, mas o sistema não

Se por algum motivo o sistema não continuar a inicialização, o bootstrap pode ser interrompido pressionando `espaço` no aviso de 10 segundos. A variável `vinum.autostart` do loader pode ser examinada digitando `show` e manipulada usando `set` ou `unset`.

Se o módulo do kernel vinum ainda não estava na lista de módulos para carregar automaticamente,

digite `load geom_vinum`.

Quando estiver pronto, o processo de inicialização pode ser continuado digitando-se `boot -as`, no qual `-as` solicita ao kernel que peça ao sistema de arquivos raiz para montar (`-a`) e fazer com que o processo de inicialização pare no modo single user(`-s`), em que o sistema de arquivos raiz é montado como somente leitura. Dessa forma, mesmo que apenas um plex de um volume multi-plex tenha sido montado, não estaremos arriscando nenhuma inconsistência de dados entre os plexes.

No prompt solicitando que um sistema de arquivos raiz seja montado, qualquer dispositivo que contenha um sistema de arquivos raiz válido pode ser inserido. Se o /etc/fstab estiver configurado corretamente, o padrão deve ser algo como `ufs:/dev/gvinum/root`. Uma opção alternativa típica seria algo como `ufs:da0d`, que poderia ser uma partição hipotética contendo o sistema de arquivos raiz pré-vinum. Deve-se tomar cuidado se uma das partições alias `a` for inserida aqui, para verificar se ela realmente faz referência aos subdiscos do dispositivo raiz vinum, porque em uma configuração espelhada, isso apenas montaria uma parte de um dispositivo raiz espelhado. Se este sistema de arquivos tiver que ser montado no modo read-write mais tarde, será necessário remover o(s) outro(s) plex(es) do volume raiz vinum, já que esses plexes carregariam dados inconsistentes.

8.4.2. Apenas o bootstrap primário carrega

Se o /boot/loader falhar ao carregar, mas o bootstrap primário ainda carregar (visível por um único traço na coluna esquerda da tela logo após o processo de boot ser iniciado), uma tentativa pode ser feita para interromper o bootstrap primário pressionando `espaço`. Isso fará com que o bootstrap pare no estágio dois. Uma tentativa pode ser feita aqui para inicializar uma partição alternativa, como a partição que contém o sistema de arquivos raiz anterior que foi movido de `a`.

8.4.3. Nada carrega e o Bootstrap entra em panic

Esta situação acontecerá se o bootstrap tiver sido destruído pela instalação do vinum. Infelizmente, o vinum acidentalmente deixa apenas 4 KB no início de sua partição livre antes de começar a escrever suas informações de cabeçalho vinum. No entanto, o estágio um e dois bootstraps mais o bsdlabel exigem 8 KB. Portanto, se uma partição vinum tiver sido iniciada no offset 0 dentro de uma slice ou disco que deveria ser inicializável, a configuração do vinum irá estragar o bootstrap.

Da mesma forma, se a situação acima foi recuperada, inicializando de uma mídia "Fixit", e o bootstrap foi reinstalado usando `bsdlabel -B` como descrito em Estágio Um e Estágio Dois, o bootstrap irá estragar o cabeçalho vinum e o vinum não encontrará mais seu(s) disco(s). Entretando nenhum dado de configuração real do vinum e nenhum volume vinum de dados foi descartado, sendo possível recuperá-los digitando de novo exatamente as mesmas configurações do vinum, a situação é difícil de corrigir de forma definitiva. Pois será necessário mover toda a partição vinum em pelo menos 4 KB, para que o cabeçalho vinum e o bootstrap do sistema não colidam mais.