

第 18 章 Vinum 卷管理程序

(欢迎提出意见和建议：freebsdhandbook@163.com)

无论你有什么样的磁盘，它们总是有一些限制：

- 它们可能容量太小。
- 它们可能速度太慢。
- 它们可能也不太可靠。

Vinum 是一个卷管理工具，是能够解决这三个问题的虚拟磁盘驱动程序。让我们来仔细地看看这些问题。已经有针对这些问题的很多方法被提出和实现了：

磁盘容量太小

磁盘越大，存储的数据也就越多。你经常会发现你需要一个比你可使用的磁盘大得多的文件系统。

无可否认，这个问题已经没有十年前那样严峻了，但它仍然存在。通过创建一个在许多磁盘上存储数据的抽象设备，一些系统可以解决这个问题。

访问瓶颈

现代系统经常需要用一种高度并发的方式来访问数据。

例如，巨大的 FTP 或 HTTP 服务器可以支持数以千计的并发会话，可以有多个连到外部世界的 100 Mbit/s 的连接，这远远地超过了绝大多数磁盘的数据传输速率。

当前的磁盘驱动器最高可以以 30 MB/s 的速度传输数据，但这个值在一个有许多不受约束的进程访问一个驱动器的环境中变得并不重要，它们可能只完成了这些值的一小部分。在这样一种情况下，从磁盘子系统的角度来看问题就更加有趣：重要的参数是在子系统上的负荷，换句话说就是传输占用了驱动器多少时间。

在任何磁盘传输中，驱动器必须先寻道，等待磁头访问第一个扇区，然后执行传输。这些动作看起来可能很细小：我们感觉不到有任何打断。

考虑一个典型的大约 10Kb 的传输：现在的高性能磁盘平均寻道时间是 6ms。最快的驱动器可以旋转在 10,000 rpm，所以平均转动潜伏期是 3ms。在 30 MB/s 时，它自己的传输大约花费 350μs，几乎没有什么可以和配置时间相比较。在这样一种情况下，最有效的传输率会降到 1 MB/s，很明显非常依赖于传输的大小。

对于这个瓶颈的一般和明显的解决方法是采用多个磁盘：而不是只使用一个大磁盘，它使用几个比较小的磁盘联合起来形成一个大的磁盘。每个磁盘都可以独立地进行传输，所以通过使用多个磁盘大大提高了数据吞吐量。

所要求的吞吐量的提高当然要比磁盘的数量小得多：虽然每个驱动器能并行传输，但没有办法确保请求能够平均分布到每个驱动器上。不可避免一个驱动器的负载可能比另一个要高得多。

磁盘的负载平衡非常依赖于驱动器上数据的共享方式。在下面的讨论中，将把磁盘存储想象成一个巨大的用编号来设定地址的数据扇区，有点像一本书的页。最明显的方法是把虚拟磁盘分成许多连续的扇区组，每个扇区大小就是独立的磁盘大小，用这种方法来存储数据，就像把一本厚厚的书分成很多小的章节。这个方法叫做串联，它有一个优点就是磁盘不需要有任何特定的大小关系。当访问到的虚拟磁盘根据它的地址空间来分布的时候，它能工作得很好。当访问集中在一个比较小的区域的时候，性能的提高没有显著的改进。图 17-1 举例说明了用串联组织的方式来分配存储单元的顺序。

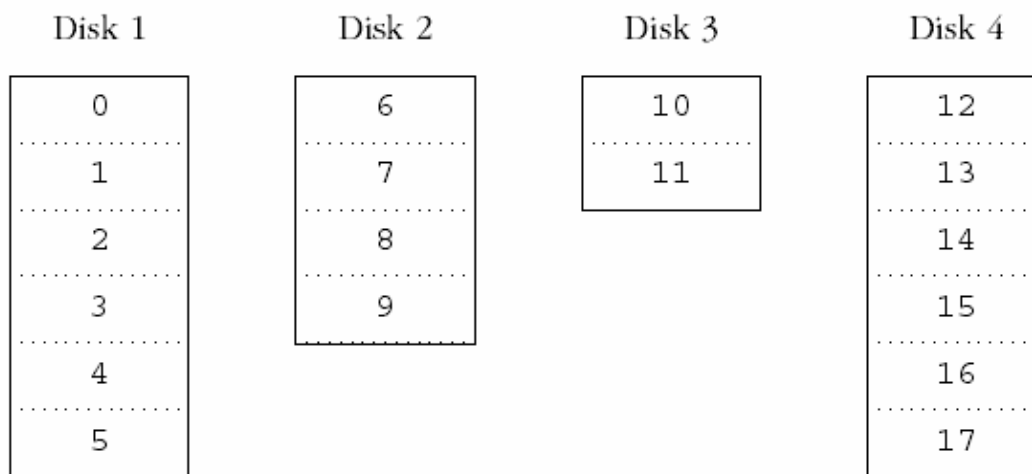


图 18-1 串联组织

另外一种影射方法是把地址空间分布在比较小的，容量一样的磁盘上，从而能够在不同的设备上存储它们。例如，前 256 个扇区可能存储在第一个磁盘上，接着的 256 个扇区存储在另一个磁盘上等等。写满最后一个磁盘后，进程会重复以前的工作，直到所有的磁盘被写满。这个影射叫做分段 (strip) 或 RAID-0 (RAID 代表 Redundant Array of Inexpensive Disks，提供多种容错的形式)，但后面这个术语可能会有些让人误解：它不提供冗余功能。分段要求很精确地定位数据，通过多个磁盘进行数据传输的时候，它可能会引起额外的 I/O 负载，但它也可能提供更多的恒负载。图 18-2 显示了用分段形式分配的存储单元的顺序。

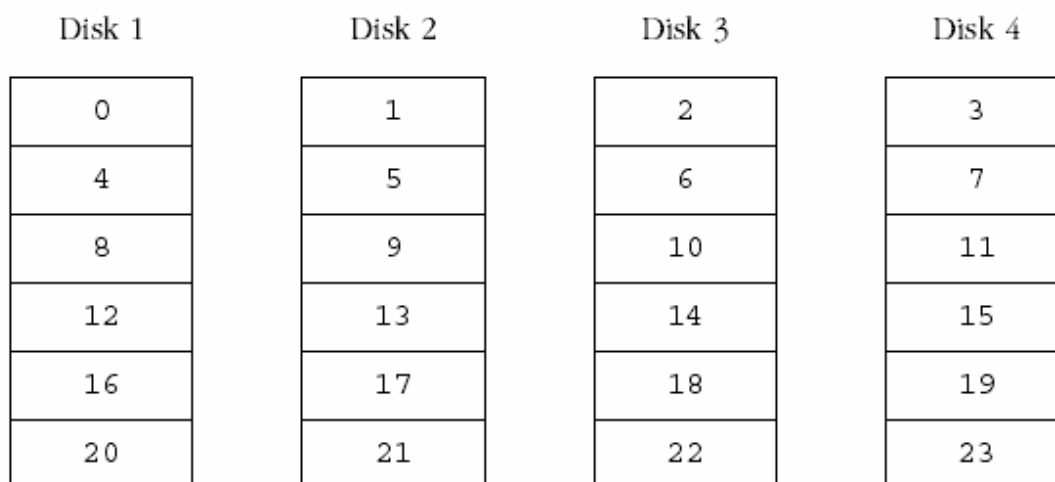


图 18-2 分段组织

数据的完整性

当前磁盘的最后一个问题是它们不太可靠。虽然磁盘驱动器的可靠性在过去几年有了很大的提高，它们仍然是会损坏的服务器的最核心组件。当它们发生故障的时候，结果可能是灾难性的：替换一个坏的磁盘驱动器，然后恢复数据可能要花费几天时间。

解决这个问题的传统方法是建立镜像，在不同的物理硬件上对数据做两个拷贝。由于 RAID 的出现，所以这个技术也被叫做 RAID-1。任何写到卷的数据也会被写到镜像上，所以可以从任何一个拷贝读取数据，如果其中有一个失败了，数据就可以在其他驱动器上访问到。

镜像有两个问题：

- 价格。它需要两倍的存储容量。
- 性能影响。写入操作必须在两个驱动器上执行，所以它们花费两倍的带宽。读取数据并不会影响性能：看起来它们会更快。

Vinum 目标

为了解决这些问题，Vinum 提出了一个四层的目标结构：

- 最显著的目标是虚拟磁盘，叫做卷。卷本质上与一个 UNIX 磁盘驱动器有同样的属性，虽然它们是有些不太一样。它们没有大小的限制。
- 卷下面是 plex，每一个表示卷的所有地址空间。在层次结构中的这个水平能够提供冗余功能。
- 可以把 plex 想象成用一个镜像排列的方式组织起来的独立磁盘，每个都包含同样的数据。

- 由于 Vinum 存在于 UNIX 磁盘存储框架中，所以它也可能使用 UNIX 分区作为多个磁盘 plex 的组成部分，但事实上这并不可靠：UNIX 磁盘只能有有限数量的分区。取而代之，Vinum 把一个简单的 UNIX 分区分解成叫做 subdisk 的相邻区域，它可以使用这个来为 plex 建立块。
- Subdisk 位于 Vinum 驱动器上，当前的 UNIX 分区。Vinum 驱动器可以包含很多的 subdisk。除了驱动器开始的一小块区域用来存储配置和描述信息以外，整个驱动器都可以用于存储数据。

下面的章节描述了这些目标提供了 Vinum 所要求的功能的方法。

卷的大小要求

Plex 用 Vinum 配置方法可以把多个 subdisk 分布在所有的驱动上。结果，每个独立的驱动器的大小都不会限制 plex 的大小，从而不限制卷的大小。

多余的数据存储

Vinum 通过给一个卷连上多个 plex 来完成镜象的功能。每个 plex 是一个在一个卷中的数据的描述。一个卷可以包含一个到八个 plex。

虽然一个 plex 描述了一个卷的所有数据，但可能描述的部分被物理地丢失了，可能是设计的问题（没有为 plex 部分定义一个 subdisk），也可能是意外的故障（由于驱动器的故障导致）。只要至少有一个 plex 能够为卷的完全地址范围提供数据，卷就能够正常工作。

性能问题

Vinum 在 plex 水平既执行串联也执行分段：

- 一个连接的 plex 按顺序使用每个 subdisk 的地址空间。
- 一个分段的 plex 在每个 subdisk 上划分数据。Subdisk 必须是大小一样的，为了从一个连接的 plex 中区分开它，必须至少有两个 subdisk。

哪种 plex 组织？

FreeBSD 4.6 提供的 Vinum 版本能实现两种 plex：

- 串联的 plex 更加灵活：它们可以包含任何数量的 subdisk，subdisk 也可能有不同的长度。Plex 可以通过添加额外的 subdisk 来得到扩展。它们比分段 plex 需要更少的 CPU 时钟，但是在 CPU 上的负载的差异是无法测量的。另一方面，它们的负载可能不平衡，一个磁盘可能负载很重，而其他的可能很空闲。
- 分段（RAID-0）plex 的最大优点是它们减少了负载不平衡的情况：通过选择一个最合适大小的分段（大约是 256KB），你甚至可以在各个组成的驱动器上降低负载。这种方

法的缺点是在 subdisk 上非常复杂的编码和限制：它们必须是同样大小，通过添加新的 subdisk 来扩展一个 plex 是非常复杂的，以至 Vinum 当前没有实现它。Vinum 利用一个额外的，价值不高的限制：一个分段的 plex 必须有至少两个 subdisk，因为否则的话，它是无法从一个连接的 plex 进行区分的。

表 18-3 总结一下每个 plex 组织的优点和缺点

Plex 类型	最小 subdisks	可以添加的 subdisk	必须相同大小	应用
串联	1	yes	no	带有很大弹性和适中性能的大数据量存储。
分段	2	no	yes	支持大量并发访问的高性能。

图 18-3 Vinum plex 组织

一些例子

Vinum 维护着一个描述以一个独立系统为目标的配置数据库。最初，用户通过 vinum 工具程序的帮助从一个或多个配置文件创建配置数据库。Vinum 在它的控制下在每个磁盘 slice (Vinum 叫 device) 上存储一个它的配置数据库的拷贝。这个数据库在每个状态变化的时候被升级，以便能精确地回复每个 Vinum 目标的状态。

配置文件

配置文件描述了独立的 Vinum 目标。一个简单卷的定义可能是这样的：

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

这个文件描述了四个 Vinum 目标：

- drive 行描述了一个磁盘分区 (驱动器)，和与下面的硬件相关的它的位置。它给出了一个符号名 a。这个与设备名称分开的符号名允许磁盘从一个位置移动到另一个位置而不会搞混。
- volume 那行描述了一个卷。唯一的必须属性是名称，在这个例子中是 myvol。
- plex 行定义了一个 plex。唯一需要的参数是组织，在这个例子中是 concat。没有名称是必然的：系统自动通过添加 suffix.px 来从卷名称产生一个名字，这里的 x 是在卷中的 plex 的编号。而这个 plex 将被叫做 myvol.p0。
- sd 行描述了一个 subdisk。最小的说明是存储 subdisk 的驱动器名称，和 subdisk 的长度。

对于 plex，没有名称也是必然的：系统自动通过添加 suffix.sx 来分配源自 plex 的名称，

这里 x 是 plex 中 subdisk 的编号。Vinum 给这个 subdisk 命名为 myvol.p0.s0。

处理完这个文件后，vinum 会产生下面的输出：

```
vinum -> 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
```

这个输出显示了 vinum 简要的列表格式。在图 18-4 中它用图形来表示这一点。

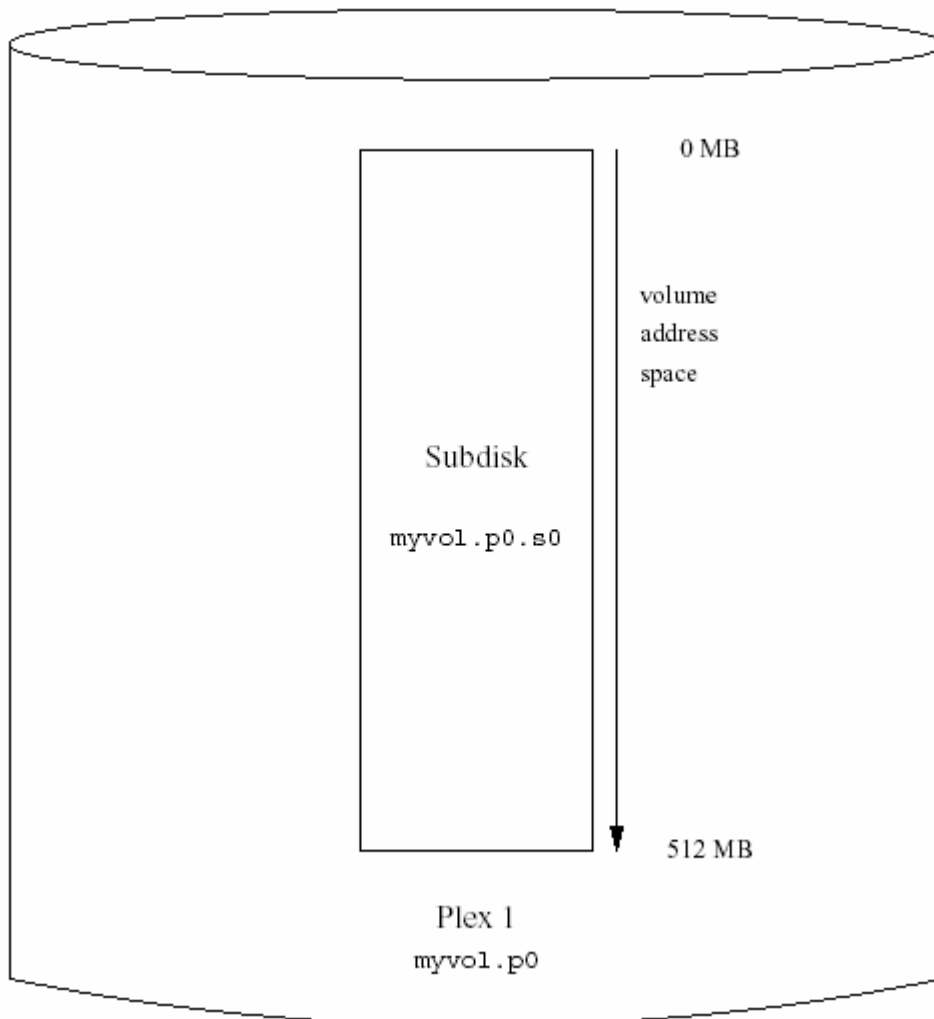


图 18-4 一个简单的 Vinum 卷

下面这个图显示了一个由按顺序排列的 subdisk 组成的 plex。在这个小小的例子中，卷包含一个 plex，plex 包含一个 subdisk。

这个特殊的卷与一个传统的磁盘分区没有什么特别的优势。下面的章节会描述到几个非常有趣的配置方法。

增强的可靠性：镜象

一个卷的可靠性是通过镜象来得到提高的。当划分一个镜象卷的时候，确保每个 plex 的 subdisk 在不同的驱动器上是很重要的，以至一个驱动器的失败将影响到两个 plex。下面的配置可以镜象一个卷：

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

在这个例子中，不需要再指定一个驱动器 a 的定义，因为 Vinum 在它的配置数据库中保存了所有目标的轨迹。处理完这个定义以后，配置是这样的：

```
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
```

图 18-5 用图形显示了结构情况。

在这个例子中，每个 plex 包含最大 512MB 的地址空间。正如前面的例子显示的，每个 plex 只包含一个简单的 subdisk。

调整性能

在前面的例子中镜象卷比一个非镜象卷有更好的可靠性，但它的性能会有所降低：每次写到卷时要求同时写到两个驱动器，用掉所有的磁盘带宽。性能的提高需要不同的方法：代替镜象，数据可以尽可能地分段到多个驱动器上。下面的配置显示了一个用 plex 分段到四个磁盘驱动器上的卷：

```
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
```

正如前面所说，不一定要定义已经知道 vinum 的驱动器。处理完这个定义之后，配置是这样的：

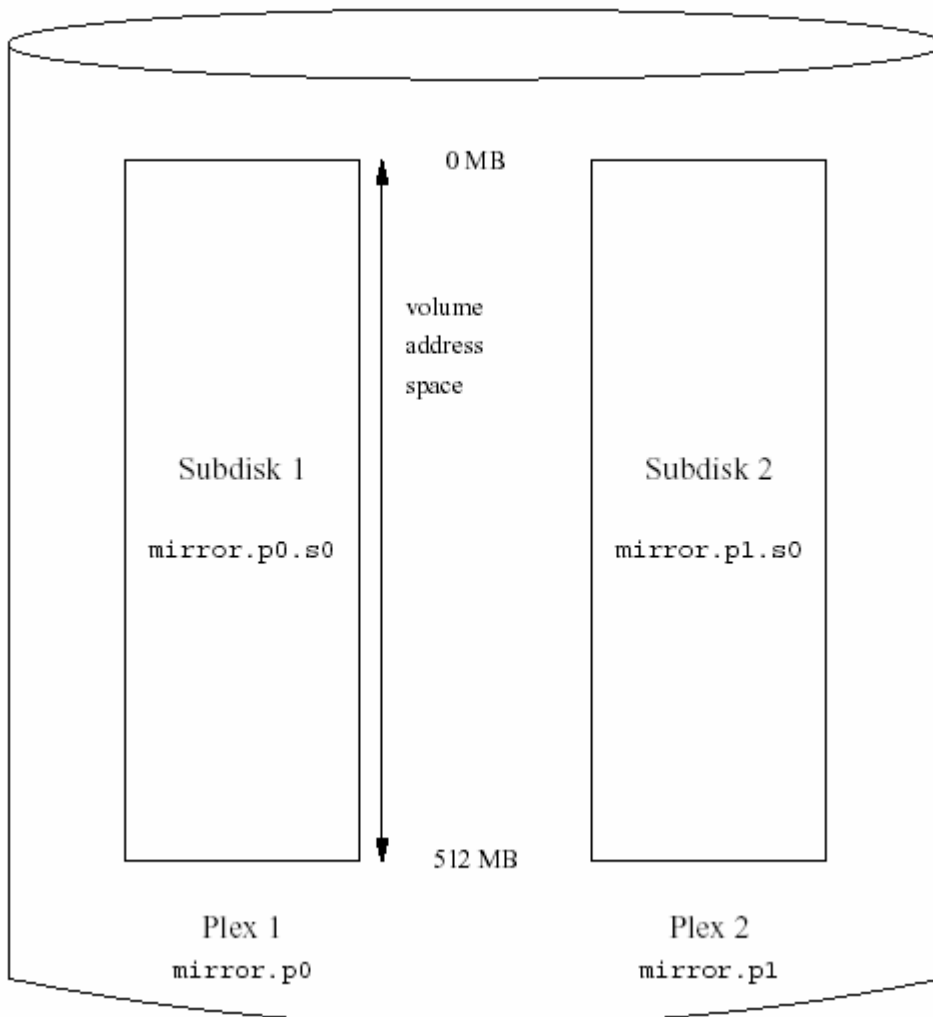


图 18-5 一个镜象 Vinum 卷

```

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

```

这个卷在图 18-6 中描述出来了。用黑块区分的分段指出了在 plex 地址空间中的位置：

颜色最浅的分段在第一个，最深的在最后。

可靠性和性能

有了足够的硬件，就可以建立起与标准的 UNIX 分区相比既提高了可靠性又提高了性能的卷。一个典型的配置文件可能是这样的：

```

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
```

第二个 plex 的 subdisk 经由两个来自第一个 plex 的驱动器发生偏移 :这可以帮助确保即使一个传输通过两个驱动器也不会到达同样的 subdisk 上。

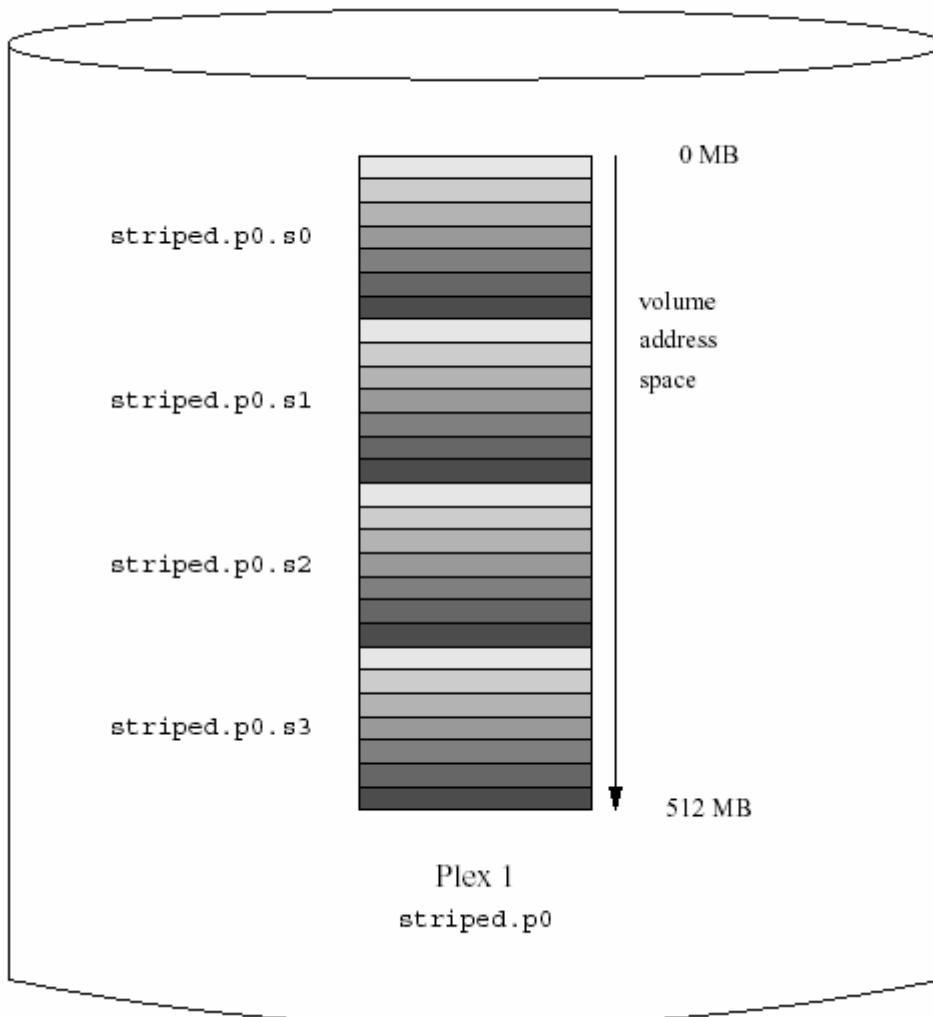


图 18-6 一个分段的 Vinum 卷

图 18-7 描述了这个卷的结构。

目标命名

正如前面所述，Vinum 给 plex 和 subdisk 分配了默认的名称，虽然它们可能会被忽略。并不建议忽略默认的名称：根据 VERITAS(R)卷管理程序的经验，它允许对目标任意命名，已经显示出这个灵活性并不能带来多少重要的好处，它可能会引起混乱。

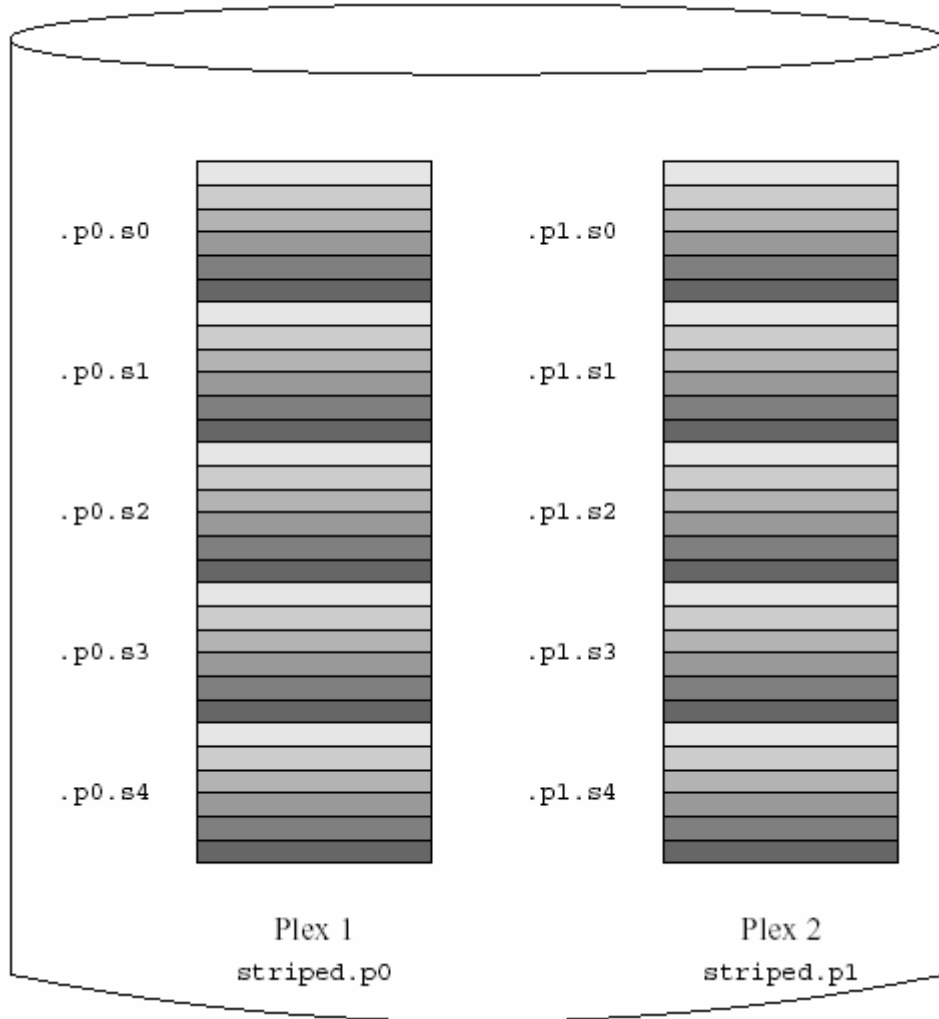


图 18-7 一个镜像，分段的 Vinum 卷

名称可以包含任何没有空格的字符，但建议把它们严格限定在字母，数字和下划线上。

卷，plex 和 subdisk 的名称最长是 64 个字符，驱动器的名称最长是 32 个字符。

Vinum 目标以/dev/vinum 的层次来分配设备节点。上面显示的配置将使 Vinum 创建下面的设备节点：

- 依次被 vinum 和 Vinum 守护程序使用的控制设备 /dev/vinum/control 和 /dev/vinum/controld。
- 每个卷的块和字符设备记录。这些是 Vinum 使用的主设备。块设备名称是卷的名称，而字符设备名按照 BSD 的传统来以字符 r 作为名称。然而，上面的配置将包含块设备 /dev/vinum/myvol，/dev/vinum/mirror，/dev/vinum/striped，/dev/vinum/raid5 和 /dev/vinum/raid10，字符设备/dev/vinum/rmyvol，/dev/vinum/rmirror，/dev/vinum/rstriped，/dev/vinum/rraid5 和/dev/vinum/rraid10。这里有一个明显的问题：可能有两个分别叫做 r

和 `rr` 的卷，但可能会有一个创建设备节点 `/dev/vinum/rr` 的冲突：它是一个卷 `r` 的字符设备还是卷 `rr` 的块设备呢？当前，`Vinum` 不会区分这个冲突：第一个定义的卷将获得名称。

- 对于每个驱动器，有一个带有记录的目录 `/dev/vinum/drive`。这些记录实际上是到相应磁盘节点的符号连接。
- 对于每个卷，有一个带有记录的目录 `/dev/vinum/volume`。它包含每个 `plex` 的子目录，按顺序包含着每个 `subdisk` 的子目录。
- 目录 `/dev/vinum/plex` 和 `/dev/vinum/sd`，`/dev/vinum/rsd` 分别包含了每个 `plex` 的块设备节点和针对 `subdisk` 的字符设备节点。

例如，看看下面的配置文件：

```
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
```

处理完这个文件，`vinum` 在 `/dev/vinum` 中创建下面的结构：

```
brwx----- 1 root wheel 25, 0x40000001 Apr 13 16:46 Control
brwx----- 1 root wheel 25, 0x40000002 Apr 13 16:46 control
brwx----- 1 root wheel 25, 0x40000000 Apr 13 16:46 controld
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 drive
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 plex
crwxr-xr-- 1 root wheel 91, 2 Apr 13 16:46 rs64
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 rsd
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 rvol
brwxr-xr-- 1 root wheel 25, 2 Apr 13 16:46 s64
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 sd
drwxr-xr-x 3 root wheel 512 Apr 13 16:46 vol
```

`/dev/vinum/drive:`

total 0

```
lrwxr-xr-x 1 root wheel 9 Apr 13 16:46 drive1 -> /dev/sd1h
lrwxr-xr-x 1 root wheel 9 Apr 13 16:46 drive2 -> /dev/sd2h
lrwxr-xr-x 1 root wheel 9 Apr 13 16:46 drive3 -> /dev/sd3h
lrwxr-xr-x 1 root wheel 9 Apr 13 16:46 drive4 -> /dev/sd4h
```

The Complete FreeBSD(4th)

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

/dev/vinum/rsd:
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

/dev/vinum/rvol:
total 0
crwxr-xr-- 1 root wheel 91, 2 Apr 13 16:46 s64

/dev/vinum/sd:
total 0
brwxr-xr-- 1 root wheel 25, 0x20000002 Apr 13 16:46 s64.p0.s0
brwxr-xr-- 1 root wheel 25, 0x20100002 Apr 13 16:46 s64.p0.s1
brwxr-xr-- 1 root wheel 25, 0x20200002 Apr 13 16:46 s64.p0.s2
brwxr-xr-- 1 root wheel 25, 0x20300002 Apr 13 16:46 s64.p0.s3

/dev/vinum/vol:
total 1
vbrwxr-xr-- 1 root wheel 25, 2 Apr 13 16:46 s64
drwxr-xr-x 3 root wheel 512 Apr 13 16:46 s64.plex

/dev/vinum/vol/s64.plex:
total 1
brwxr-xr-- 1 root wheel 25, 0x10000002 Apr 13 16:46 s64.p0
drwxr-xr-x 2 root wheel 512 Apr 13 16:46 s64.p0.sd

/dev/vinum/vol/s64.plex/s64.p0.sd:
total 0
brwxr-xr-- 1 root wheel 25, 0x20000002 Apr 13 16:46 s64.p0.s0
brwxr-xr-- 1 root wheel 25, 0x20100002 Apr 13 16:46 s64.p0.s1
brwxr-xr-- 1 root wheel 25, 0x20200002 Apr 13 16:46 s64.p0.s2
brwxr-xr-- 1 root wheel 25, 0x20300002 Apr 13 16:46 s64.p0.s3
```

虽然建议 plex 和 subdisk 不需要分配特定的名称,但 vinum 驱动器必须要被命名。这使它可以移动一个驱动器到一个不同的位置,仍可以自动地认出它。驱动器的名称最长只能有 32 个字符。

创建文件系统

卷看起来系统是与磁盘一样的，只有一个例外。不像 UNIX 驱动器，Vinum 不划分卷，因而不包含一个分区表。这需要一些磁盘工具来作修改，特别是 newfs，它先前是设法解释一个卷名的最后一个字母作为一个分区标识符。例如，一个磁盘驱动器可能有一个像 /dev/wd0a 或 /dev/da2h 这样的名称。这些名称分别表示在第一个 IDE 磁盘 (wd0) 上的第一个分区 (a) 和在第 3 个 SCSI 磁盘 (da2) 上的第 8 个分区 (h)。根据惯例，一个 vinum 卷可以被叫做 /dev/vinum/concat，这是与一个分区名称不相关的名称。

通常，如果它不能理解，newfs 解释磁盘的名称，然后指出错误。例如：

```
# newfs /dev/vinum/concat
newfs: /dev/vinum/concat: can't figure out file system partition
```

为了在这个卷上创建一个文件系统，使用 newfs 时加上 -v 选项。

```
# newfs -v /dev/vinum/concat
```

配置 Vinum

GENERIC 内核不包含 Vinum。它可能建立了一个包含 Vinum 的特定内核，但这是并不推荐的。标准的启动 Vinum 的方法是用一个 kld。你不需要为 vinum 使用 kldload：当你启动 vinum 时，它检查模块是否被加载，如果它没有加载，它会自动加载它。

启动

Vinum 用与配置文件一样的形式来把配置信息存储在磁盘 slice 上。当从配置数据库读取数据时，Vinum 认出许多不允许出现在配置文件中的关键字。例如，一个磁盘配置可能包含下面的文本：

```
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 plexo
ffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b
driveoffset 265b plexo
ffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b
driveoffset 265b plexo
ffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b
```

```
driveoffset 265b plexo
ffset 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
```

这里明显的不同是外部定位信息和命名（这两个都是被允许的，但有限制，只能被用户使用）的出现，以及状态信息（这对用户没有用）。Vinum 在配置信息中不保存有关驱动器的信息：它通过用一个 Vinum 标签扫描配置磁盘驱动器来找到驱动器。既然它们已经被分配了不同的 UNIX 驱动器 ID，这就启用了 Vinum 来正确地识别驱动器。

自动启动

为了在你启动系统的时候能自动启动 Vinum，需要确保你在/etc/rc.conf 文件中有下面这行：

```
start_vinum="YES"          # set to YES to start vinum
```

如果你没有/etc/rc.conf 文件，就用这个内容创建一个。这将使系统在启动时自动加载 Vinum kld，然后启动任何在配置文件中提到的目标。在挂上文件系统之前需要先完成这个，所以可以在 Vinum 卷上自动 fsck 和挂上文件系统。

当你用 Vinum 启动命令启动 Vinum 时, Vinum 从一个 Vinum 驱动器上读取配置数据库。在通常情况下, 每个驱动器上包含一个配置数据库的拷贝, 所以并不介意读取哪个驱动器。然而, 在一次系统崩溃后, Vinum 必须决定哪个驱动器最近被升级了, 然后从这个驱动器读取配置。如果必要的话, 它接着从旧的驱动器升级配置文件。

```
XXXXXXX bmah          2002/01/11 15:55:59 PST
Modified files:
  share/man/man7      tuning.7
Log:
newfs -U enables softupdates beginning with FreeBSD 4.5.
PR:          33391
Submitted by:  Ceri <setantae@submonkey.net>
Revision  Changes  Path
1.43      +2 -2      src/share/man/man7/tuning.7
```