

# ?? 3. ?(Pools)

ZFS 的 池 zpool 是 ZFS 的 基础 单元 , 池 的 大小 可以 任意 扩展 . 池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

## ZFS ??

UFS 的 extfs 的 池 大小 由 池 的 物理 大小 决定 . 池 的 大小 由 池 的 物理 大小 决定 . NTFS 的 FAT 的 池 大小 由 池 的 物理 大小 决定 .

ZFS 的 池 大小 由 池 的 物理 大小 决定 . 池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

池 的 大小 由 池 的 物理 大小 决定 , ZFS 的 池 大小 由 池 的 物理 大小 决定 ! 池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

ZFS 的 池 大小 由 池 的 物理 大小 决定 . 池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

ZFS 的 池 大小 由 池 的 物理 大小 决定 , 池 的 大小 由 池 的 物理 大小 决定 .

## ????, RAID ? ? (Stripes, RAID, and Pools)

池 的 大小 由 池 的 物理 大小 决定 . ZFS 的 池 大小 由 池 的 物理 大小 决定 .

RAID 10 的寫入速度會比 RAID 5 快，因為 RAID 10 的寫入是並行的，而 RAID 5 的寫入是串行的。RAID 10 的讀取速度也會比 RAID 5 快，因為 RAID 10 的讀取是並行的，而 RAID 5 的讀取是串行的。RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都很快，而且它的容錯能力也很強，因為它需要兩個磁碟才能讀取或寫入一個資料塊，所以只要其中一個磁碟沒有壞，資料就可以讀取或寫入。

?? (Viewing Pools)

要查看 zpool 的狀態，可以使用 `zpool list` 命令。

```
$ zpool list
```

NAME	SIZE	ALLOC	FREE	EXPANDSZ	FRAG	CAP	DEDUP	HEALTH	ALTR00T
db	2.72T	1.16G	2.72T	-	0%	0%	1.00x	ONLINE	-
zroot	920G	17.3G	903G	-	2%	1%	1.00x	ONLINE	-



在 2015 年之前，ZFS 的 VDEV 只能由一个物理设备组成。这意味着，如果你想要提高性能或冗余，你必须使用多个物理设备。ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。

在 2015 年，OpenZFS 引入了 VDEV 的虚拟化支持。这意味着，你可以将多个物理设备虚拟化为一个 VDEV。这允许你在一个物理设备上运行多个 VDEV，从而提高硬件利用率。

ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。

VDEV ?? (Removing VDEVs)

在 2015 年之前，ZFS 的 VDEV 只能由一个物理设备组成。这意味着，如果你想要提高性能或冗余，你必须使用多个物理设备。ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。

在 2015 年，OpenZFS 引入了 VDEV 的虚拟化支持。这意味着，你可以将多个物理设备虚拟化为一个 VDEV。这允许你在一个物理设备上运行多个 VDEV，从而提高硬件利用率。

? ?? ? ??? ?? ?? (Pools Alignment and Disk Sector Size)

ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。

在 2015 年，OpenZFS 引入了 VDEV 的虚拟化支持。这意味着，你可以将多个物理设备虚拟化为一个 VDEV。这允许你在一个物理设备上运行多个 VDEV，从而提高硬件利用率。

??? ?? (Partition Alignment)

在 2015 年之前，ZFS 的 VDEV 只能由一个物理设备组成。这意味着，如果你想要提高性能或冗余，你必须使用多个物理设备。ZFS 的 RAID-Z 功能允许你在多个 VDEV 上分布数据，从而提高冗余和性能。

在 2015 年，OpenZFS 引入了 VDEV 的虚拟化支持。这意味着，你可以将多个物理设备虚拟化为一个 VDEV。这允许你在一个物理设备上运行多个 VDEV，从而提高硬件利用率。



`/etc/sysctl.conf` 文件 添加 `sysctl vfs.zfs.min_auto_ashift=12` 并 重启 ashift 服务。

```
$ sysctl vfs.zfs.min_auto_ashift=12
```

在 系统 启动 时 添加 配置 , 在 文件 `/etc/sysctl.conf` 添加 配置 如下。

在 系统 启动 时 添加 `FreeBSD 10.1` 配置 如下。在 `FreeBSD` 系统 中 , `sysctl` 配置 如下 添加 `ashift=12` 配置 如下。

## ?? FreeBSD Ashift (Older FreeBSD Ashift)

10.1 版本 `FreeBSD` 系统 中 添加 `FreeBSD` 系统 中 添加 `ashift sysctl` 配置 如下 , `ZFS` 系统 中 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 , 在 系统 启动 时 添加 配置 如下。

在 系统 启动 时 添加 配置 如下 添加 配置 如下。在 `FreeBSD` 系统 中 `GEOM` 系统 中 `gnop(8)` 系统 中 添加 配置 如下 添加 配置 如下。在 `gnop` 系统 中 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 (在 系统 启动 时 添加 配置 如下) 。"在 系统 启动 时 添加 配置 如下 , 4096 系统 中 添加 配置 如下。"在 `gnop` 系统 中 添加 配置 如下。在 系统 启动 时 添加 配置 如下 `zpool` 系统 中 添加 配置 如下。在 `/dev/gpt/zfs0` 系统 中 添加 配置 如下 添加 配置 如下。

```
$ gnop create -S 4096 /dev/gpt/zfs0
```

在 系统 启动 时 添加 `/dev/gpt/zfs0.nop` 系统 中 添加 配置 如下。在 系统 启动 时 添加 `VDEV` 系统 中 添加 配置 如下 `ZFS` 系统 中 添加 配置 如下 `VDEV` 系统 中 添加 配置 如下。在 系统 启动 时 添加 配置 如下 `ZFS` 系统 中 添加 配置 如下 添加 配置 如下 , 在 系统 启动 时 添加 配置 如下 添加 配置 如下 添加 配置 如下。

```
$ zpool create compost mirror gpt/zfs0.nop gpt/zfs1
```

`gnop(8)` 系统 中 添加 配置 如下 添加 配置 如下。在 `gnop(8)` 系统 中 添加 配置 如下 添加 配置 如下 `ZFS` 系统 中 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 添加 配置 如下 `ZFS` 系统 中 添加 配置 如下 添加 配置 如下 添加 配置 如下。

## ?? ?? ???? (Creating Pools and VDEVs)

`zpool(8)` 系统 中 添加 配置 如下 添加 配置 如下。在 `zpool(8)` 系统 中 添加 配置 如下 `VDEV` 系统 中 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 `RAID-Z` 系统 中 添加 配置 如下 , 在 系统 启动 时 添加 配置 如下。在 系统 启动 时 添加 配置 如下 `VDEV` 系统 中 添加 配置 如下。

在 系统 启动 时 添加 配置 如下 `ashift` 系统 中 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 添加 配置 如下。在 系统 启动 时 添加 配置 如下 "set ashift" 系统 中 添加 配置 如下。

## ?? ???? (Sample Drives)

```
0####   ##    #####     #   ##      #   ##       ##    ##          ##        #   #  
##### . #####         ##   #####   . ####   #####           ##   ##   #####   #####  
##### zfs#   ##      # GPT #####         . # #####           ## 1GB ##   ##### gpart(8)#  
##   ## ZFS #####      ## 6## 1TB #####           #####
```

```
$ gpart create -s gpt da0
$ gpart add -a 1m -slg -l sw0 -t freebsd-swap da0
$ gpart add -a 1m -l zfs0 -t freebsd-zfs da0
```

```
$ gpart show -l da0
=>      40  1953525088  da0  GPT  (932G)
      40           2008    -  free  -  (1.0M)
     2048      2097152    1  sw0  (1.0G)
    2099200  1951424512    2  zfs0  (931G)
1953523712      1416    -  free  -  (708K)
```

 GPT  ZFS     .  

????? ? (Striped Pools)

```

00  0000  00  0000  0000  000  00  000  00000
0000  0000  000  000  00000  0000  00  00000
00  00  000  00000  . 00  000  00  000000  0000  00  000  . 0000  500
0000  0000  000  00  00000  .

```

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create compost gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs4 gpt/zfs4
```

```

[ ] [ ] [ ] [ ] [ ] [ ] . [ ] zpool status [ ] [ ] [ ] [ ] [ ] [ ] .

```

```
$ zpool status
  pool: compost
  state: ONLINE
    scan: none requested
config:

NAME            STATE READ WRITE CKSUM
compost         ONLINE   0     0     0
  gpt/zfs0      ONLINE   0     0     0
```

```
gpt/zfs1  ONLINE   0      0      0
gpt/zfs2  ONLINE   0      0      0
gpt/zfs3  ONLINE   0      0      0
gpt/zfs4  ONLINE   0      0      0
```

5個 物理 1個 物理 . 1個 物理 1個 VDEV物理 .

1個 物理 物理 物理 1個 物理 . 1個 物理 1個 物理 VDEV 1個 物理 物理  
物理 , VDEV 1個 物理 物理 . 物理 1個 物理 物理 物理 .  
1個 物理 1個 物理 物理 .

## ?? ? (Mirrored Pools)

物理 物理 1個 物理 1個 物理 物理 . 物理 1個 物理 物理  
物理 物理 物理 1個 物理 物理 物理 . 1個 物理 1個 物理 物理 , 1個 物理  
物理 .

物理 **zpool create** 物理 1個 物理 物理 . **mirror** 物理 物理 物理 物理 物理  
物理 . 1個 物理 1個 物理 ashift 物理 .

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create reflect mirror gpt/zfs0 gpt/zfs1
```

**zpool status** 物理 物理 物理 .

```
$ zpool status
pool: reflect
state: ONLINE
scan: none requested
config:

NAME        STATE READ WRITE CKSUM
reflect     ONLINE   0      0      0
mirror-0    ONLINE   0      0      0
  gpt/zfs0  ONLINE   0      0      0
  gpt/zfs1  ONLINE   0      0      0

errors: No known data errors
```

**zpool** 物理 物理 mirror-0物理 1個 物理 物理 . mirror-0 物理 VDEV物理 . 1個 VDEV物理  
1個 物理 , 1個 **gpt/zfs0** **gpt/zfs1** 物理 物理 . 物理 1個 物理 物理 物理 物理  
1個 物理 . 物理 1個 物理 1個 物理 物理 物理 .



```
$ zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3
```

我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 (FreeBSD Mastery: Advanced ZFS 第 10 章 第 10.1 节)。

RAID-Z Pools

在 RAID-Z 池中，数据被写入多个磁盘，并且每个磁盘都包含完整的数据副本。RAID-Z 池使用 `zpool create` 命令来创建。
 我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。

我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create raidz1 gpt/zfs0 gpt/zfs1 gpt/zfs2
```

我们使用 `zpool status` 命令来查看 RAID-Z 池的状态。

```
$ zpool status bucket
pool: bucket
state: ONLINE
scan: none requested
config:

NAME        STATE READ WRITE CKSUM
bucket      ONLINE  0     0     0
raidz1-0    ONLINE  0     0     0
gpt/zfs0    ONLINE  0     0     0
gpt/zfs1    ONLINE  0     0     0
gpt/zfs2    ONLINE  0     0     0
```

我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。

```
$ zpool create bucket raidz3 gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4 gpt/zfs5
```

我们使用 `zpool status` 命令来查看 RAID-Z 池的状态。

```
$ zpool status
pool: bucket
```

```
state: ONLINE
  scan: none requested
config:

NAME          STATE  READ WRITE CKSUM
bucket        ONLINE    0     0     0
raidz3-0      ONLINE    0     0     0
  gpt/zfs0    ONLINE    0     0     0
  gpt/zfs1    ONLINE    0     0     0
...
```

□□ □□ □□ □□ VDEV□ □□□□ . □□ □□ □□ VDEV□ □□□□ □□ □□ □□ ?

## ?? VDEV ? (Multi-VDEV Pools)

[illegible]
















```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create barrel mirror gpt/zfs0 gpt/zfs1 mirror gpt/zfs2 gpt/zfs3
```

```
# Create zpool barrel
zpool create barrel zpool(8) barrel # Create mirror pool gpt/zfs0
# Mirror pool gpt/zfs1
zpool mirror gpt/zfs1 VDEV1 VDEV2 # Mirror pool gpt/zfs2
zpool mirror gpt/zfs2 VDEV3 VDEV4 # Mirror pool gpt/zfs3
zpool mirror gpt/zfs3 VDEV5 VDEV6
```

```
$ zpool status barrel
  pool: barrel
state: ONLINE
  scan: none requested
config:

NAME        STATE READ WRITE CKSUM
barrel      ONLINE  0     0     0
  mirror-0  ONLINE  0     0     0
    gpt/zfs0 ONLINE  0     0     0
    gpt/zfs1 ONLINE  0     0     0
```

```
mirror-1  ONLINE   0      0      0
gpt/zfs2  ONLINE   0      0      0
gpt/zfs3  ONLINE   0      0      0
```

我们使用 mirror-0 和 mirror-1 两个 VDEV 来创建 RAID-10。ZFS 的 VDEV 可以包含多个物理设备，也可以包含其他 VDEV。FreeBSD 的 GEOM 系统支持 RAID，但 ZFS 的 RAID 功能更强大。ZFS 的 RAID-Z 功能可以创建 RAID-Z1、RAID-Z2 和 RAID-Z3。

```
$ zpool create vat raidz1 gpt/zfs0 gpt/zfs1 gpt/zfs2 raidz1 gpt/zfs3 gpt/zfs4 gpt/zfs5
```

我们使用 RAID-Z1 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 功能可以创建 RAID-Z1。我们使用 gpt/zfs0, gpt/zfs1, gpt/zfs2 三个 VDEV 来创建 RAID-Z1。我们使用 gpt/zfs3, gpt/zfs4, gpt/zfs5 三个 VDEV 来创建 RAID-Z1。

```
$ zpool status vat
```

```
...
```

```
config:
```

NAME	STATE	READ	WRITE	CKSUM
vat	ONLINE	0	0	0
raidz1-0	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
gpt/zfs1	ONLINE	0	0	0
gpt/zfs2	ONLINE	0	0	0
raidz1-1	ONLINE	0	0	0
gpt/zfs3	ONLINE	0	0	0
gpt/zfs4	ONLINE	0	0	0
gpt/zfs5	ONLINE	0	0	0

我们使用 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 功能可以创建 RAID-Z1。我们使用 gpt/zfs0, gpt/zfs1, gpt/zfs2 三个 VDEV 来创建 RAID-Z1。我们使用 gpt/zfs3, gpt/zfs4, gpt/zfs5 三个 VDEV 来创建 RAID-Z1。

我们使用 RAIDZ 来创建 RAID-Z1。ZFS 的 RAID-Z1 功能可以创建 RAID-Z1。我们使用 gpt/zfs0, gpt/zfs1, gpt/zfs2 三个 VDEV 来创建 RAID-Z1。我们使用 gpt/zfs3, gpt/zfs4, gpt/zfs5 三个 VDEV 来创建 RAID-Z1。

我们使用 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 功能可以创建 RAID-Z1。我们使用 gpt/zfs0, gpt/zfs1, gpt/zfs2 三个 VDEV 来创建 RAID-Z1。我们使用 gpt/zfs3, gpt/zfs4, gpt/zfs5 三个 VDEV 来创建 RAID-Z1。

?? ?? ?? (Using Log Devices)

2個 1個 1個 , ZFS 1個 1個 1個 1個 /1個 1個 1個 1個 1個 1個 1個 1個  
1個 1個 . 1個 1個 1個 1個 1個 1個 1個 1個 SSD 1個 . **zpool(8)** 1個 1個  
1個 **log**, 1個 1個 **cache** 1個 1個 . 1個 1個 1個 **log** 1個 **cache** 1個 1個 1個  
1個 1個 . 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個  
1個 .

```
$ zpool create scratch gpt/zfs0 log gtp/zlog0 cache gpt/zcache1
```

1個 1個 1個 1個 1個 .

```
$ zpool status scratch
```

...

config:

NAME	STATE	READ	WRITE	CKSUM
scratch	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
logs				
gpt/zlog0	ONLINE	0	0	0
cache				
gpt/zcache1	ONLINE	0	0	0

1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 1個 1個 1個 1個 1個  
1個 1個 . 1個 1個 1個 1個 ZFS 1個 1個 1個 1個 1個 1個 . 1個 ZIL 1個  
1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 *gpt/zfs0* 1個  
*gpt/zfs3* 1個 1個 1個 1個 1個 1個 1個 , 1個 1個 1個 *gpt/zlog0*  
*gpt/zlog1* 1個 1個 .

```
$ zpool create db mirror gpt/zfs0 gpt/zfs1 mirror gpt/zfs2 gpt/zfs3 log mirror gpt/zlog0  
gpt/zlog1
```

1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 1個 1個 1個 1個 1個  
1個 1個 1個 1個 , 1個 1個 1個 1個 1個 . 1個 1個 SSD 1個 1個 1個 1個 1個  
1個 1個 1個 1個 !

## ???? ?? VDEV (Mismatched VDEVs)

1個 1個 1個 1個 VDEV 1個 1個 1個 1個 , **zpool(8)** 1個 1個 1個 1個  
1個 .

```
$ zpool create daftie raidz gpt/zfs0 gpt/zfs1 gpt/zfs2 mirror gpt/zfs3 gpt/zfs4 gpt/zfs5  
invalid vdev specification  
use '-f' to override the following errors:
```

mismatched replication level: both raidz and mirror vdevs are present

**zpool(8)** 提供了一種簡單的方法，用於創建和管理 ZFS 池。要創建一個池，請使用 **zpool create** 命令。例如，要創建一個名為 **db** 的池，並使用四個 VDEV（每個 VDEV 由一個 16TB 的 SATA 硬碟組成），請執行以下命令：

**ZFS** 池的創建過程如下：

```
zpool create db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

## ??? ??? (Reusing Providers)

在創建池時，如果指定的 VDEV 已經存在於另一個池中，則會收到錯誤消息。要解決此問題，請使用 **-f** 選項強制創建池，即使 VDEV 已經存在。

```
$ zpool create db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
invalid vdev specification
use '-f' to override the following errors:
/dev/gpt/zfs3 is part of exported pool 'db'
```

要強制創建池，請使用 **-f** 選項。例如，要強制創建名為 **db** 的池，請執行以下命令：

```
zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

```
$ zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

**ZFS** 池的創建過程如下：

```
zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

## ? ??? (Pool Integrity)

**ZFS** 池的完整性檢查過程如下：

```
fsck(8) 提供了一種簡單的方法，用於檢查 ZFS 池的完整性。要檢查池的完整性，請使用 zpool status 命令。例如，要檢查名為 db 的池的完整性，請執行以下命令：
```

## ZFS ??? (ZFS Integrity)

要檢查池的完整性，請使用 **zpool status** 命令。例如，要檢查名為 **db** 的池的完整性，請執行以下命令：



```
$ zpool scrub zroot
```

zpool status 命令用于检查 ZFS 池的状态。它显示了池的扫描进度、扫描速度、扫描时间以及扫描结果。

```
$ zpool status
...
scan: scrub in progress since Tue Feb 24 11:52:23 2015
12.8G scanned out of 17.3G at 23.0M/s, 0h3m to go
0 repaired, 74.08% done
...
```

zpool scrub -s 命令用于强制立即开始扫描。它会在指定的池中启动扫描，并显示扫描进度。

```
$ zpool scrub -s zroot
```

zpool scrub 命令用于检查 ZFS 池的扫描进度。它显示了池的扫描进度、扫描速度、扫描时间以及扫描结果。

## ??? ?? (Scrub Frequency)

ZFS 池的扫描频率可以通过 zpool scrub 命令来查看。它显示了池的扫描频率、扫描速度、扫描时间以及扫描结果。

zpool scrub 命令用于检查 ZFS 池的扫描进度。它显示了池的扫描进度、扫描速度、扫描时间以及扫描结果。

## ? ?? (Pool Properties)

ZFS 池的属性可以通过 zpool get all 命令来查看。它显示了池的属性、属性值以及属性来源。

zpool get all 命令用于查看 ZFS 池的属性。它显示了池的属性、属性值以及属性来源。

## ? ?? ?? (Viewing Pool Properties)

zpool get all 命令用于查看 ZFS 池的属性。它显示了池的属性、属性值以及属性来源。

```
$ zpool get all zroot
NAME    PROPERTY  VALUE          SOURCE
zroot   scrub     on             default
```

```
zroot  size      920G  -
zroot  capacity  1%    -
zroot  altroot   -      default
zroot  health    ONLINE -
...
```

zpool 是 zfs 的池，是 zfs 文件系统的基础。zpool 是 zfs 的容器，zfs 文件系统是建立在 zpool 之上的。zpool 的大小可以设置，zfs 文件系统的大小是随着 zpool 的大小而变化的。zpool 的健康状态可以查看，zfs 文件系统的大小可以查看。zpool 的大小可以设置，zfs 文件系统的大小是随着 zpool 的大小而变化的。zpool 的健康状态可以查看，zfs 文件系统的大小可以查看。

查看 zpool 的大小可以使用 `zpool get` 命令。

```
$ zpool get size
NAME      PROPERTY  VALUE  SOURCE
db        size      2.72T  -
zroot     size      920G   -
```

zpool 的大小可以查看，zfs 文件系统的大小可以查看。

Changing Pool Properties

zpool 的属性可以设置，zfs 文件系统的属性也可以设置。zpool 的属性可以使用 `zpool set` 命令设置，zfs 文件系统的属性可以使用 `zfs set` 命令设置。zpool 的属性可以设置，zfs 文件系统的属性也可以设置。

```
$ zpool set comment="Main OS files" zroot
```

zpool 的属性可以查看，zfs 文件系统的属性也可以查看。

```
$ zpool get comment
NAME      PROPERTY  VALUE          SOURCE
db        comment   -              default
zroot     comment   Main OS files  local
```

zpool 的属性可以查看，zfs 文件系统的属性也可以查看。zpool 的属性可以查看，zfs 文件系统的属性也可以查看。zpool 的属性可以查看，zfs 文件系统的属性也可以查看。



```
$ zpool set comment="-" zroot
# zpool get comment
NAME    PROPERTY  VALUE  SOURCE
db       comment   -       default
zroot    comment   -       local
```

zpool 命令可以设置 zpool 的属性，比如设置 comment 属性。

zpool 命令还可以设置 zpool 的 mount 属性，比如设置 canmount 属性为 off，这样 zpool 就不会自动挂载了。

```
$ zpool create -o altroot=/mnt -0 canmount=off -m none zroot /dev/gpt/disk0
```

zpool 命令的 **altroot** 属性指定了 zpool 的根目录，这里指定为 /mnt。zpool 命令的 **canmount** 属性指定了 zpool 是否可以被挂载，这里指定为 off。zpool 命令的 **m** 属性指定了 zpool 的 mount 策略，这里指定为 none。zpool 命令的 **0** 属性指定了 zpool 的 mount 选项，这里指定为 none。

## ? (Pool History)

zpool 命令可以查看 zpool 的历史记录，比如查看 zpool 的创建时间、设置时间、挂载时间等。

zpool 命令的 **history** 属性可以查看 zpool 的历史记录。

```
$ zpool history zroot
History for 'zroot':
2014-01-07.04:12:05 zpool create -o altroot=/mnt -0 canmount=off -m none zroot mirror /
dev/gpt/disk0.nop /dev/gpt/disk1.nop
2014-01-07.04:12:50 zfs set checksum=fletcher4 zroot
2014-01-07.04:13:00 zfs set atime=off zroot
...
```

FreeBSD 系统使用 ZFS 文件系统，zpool 命令可以管理 ZFS 文件系统。

zpool 命令可以设置 zpool 的属性，比如设置 comment 属性。

```
...
2015-03-12.14:36:35 zpool set comment=Main OS files zroot
2015-03-12.14:43:45 zpool set comment=- zroot
```

comment 属性可以设置 zpool 的 comment 属性，比如设置 comment 属性为 Main OS files。

zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。

## Zpool ?? ?? ??? (Zpool Maintenance Automation)

FreeBSD 的 **periodic(8)** 是用于定期执行任务的工具。它可以根据配置文件 `periodic.conf` 中的配置，定期执行各种任务。ZFS 的 `daily_status_zfs_enable` 配置项，用于启用 ZFS 的每日状态检查任务。

```
daily_status_zfs_enable="YES"
```

通过 **periodic(8)** 配置，可以定期执行 `zpool status -x` 命令，检查所有 ZFS 池的健康状态。如果所有池都是健康的，会输出 "all pools are healthy."。

为了更详细地检查 ZFS 池的状态，可以在 `periodic.conf` 中添加 `daily_status_zfs_zpool_list=yes`。这样，`periodic(8)` 就会定期执行 `zpool status -x` 命令，并输出详细的池状态信息。

FreeBSD 的 `periodic.conf` 文件中，已经配置了 `daily_status_zfs_enable=YES`。这表示，FreeBSD 的 `periodic(8)` 工具，会定期执行 ZFS 的每日状态检查任务。35 秒后，会再次检查 ZFS 池的状态。

```
daily_scrub_zfs_enable="YES"
```

FreeBSD 的 `periodic.conf` 文件中，已经配置了 `daily_scrub_zfs_enable=YES`。这表示，FreeBSD 的 `periodic(8)` 工具，会定期执行 ZFS 的每日数据校验任务。35 秒后，会再次检查 ZFS 池的状态。

```
daily_scrub_zfs_pools="zroot prod test"
```

通过 `daily_scrub_zfs_pools` 配置项，可以指定要检查的 ZFS 池。例如，`daily_scrub_zfs_pools="zroot prod test"` 表示，要检查 `zroot`、`prod` 和 `test` 三个池。

```
daily_scrub_zfs_default_threshold="10"
```

通过 `daily_scrub_zfs_${poolname}_threshold` 配置项，可以指定每个池的阈值。例如，`daily_scrub_zfs_prod_threshold="7"` 表示，`prod` 池的阈值为 7。

```
daily_scrub_zfs_prod_threshold="7"
```

通过 `daily_scrub_zfs_${poolname}_threshold` 配置项，可以指定每个池的阈值。

## ? ?? (Removing Pools)

通过 `zpool destroy` 命令，可以删除 ZFS 池。例如，`zpool destroy prod` 表示，删除 `prod` 池。

```
$ zpool destroy test
```

zpool 的 destroy 命令会强制删除 zpool，即使它包含数据。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态。如果 zpool 处于 standby 状态，则可以直接删除。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态，然后再删除。

zpool 的 destroy 命令会强制删除 zpool，即使它包含数据。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态。如果 zpool 处于 standby 状态，则可以直接删除。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态，然后再删除。

## Zpool ?? ??? (Zpool Feature Flags)

ZFS 的 feature flags 是用于控制 ZFS 功能的标志。它们可以分为 enabled、active 和 disabled 三种状态。enabled 表示功能已启用，active 表示功能正在运行，disabled 表示功能已禁用。ZFS 的 feature flags 可以通过 zpool 命令进行管理。

zpool 命令的 feature 子命令用于管理 ZFS 的 feature flags。它包括 feature on、feature off、feature enable、feature disable 和 feature list 等子命令。feature on 用于启用 feature flag，feature off 用于禁用 feature flag，feature enable 用于启用 feature flag，feature disable 用于禁用 feature flag，feature list 用于列出 feature flags 的当前状态。

OpenZFS 的 feature flags 是用于控制 OpenZFS 功能的标志。它们可以分为 enabled、active 和 disabled 三种状态。enabled 表示功能已启用，active 表示功能正在运行，disabled 表示功能已禁用。OpenZFS 的 feature flags 可以通过 zpool 命令进行管理。

FreeBSD 的 zpool-features(7) 手册页提供了关于 ZFS 的 feature flags 的详细信息。它包括 feature on、feature off、feature enable、feature disable 和 feature list 等子命令。FreeBSD 的 zpool-features(7) 手册页还提供了关于 feature flags 的详细说明。

zpool 命令的 feature 子命令用于管理 ZFS 的 feature flags。它包括 feature on、feature off、feature enable、feature disable 和 feature list 等子命令。feature on 用于启用 feature flag，feature off 用于禁用 feature flag，feature enable 用于启用 feature flag，feature disable 用于禁用 feature flag，feature list 用于列出 feature flags 的当前状态。

## ?? ??? ?? (Viewing Feature Flags)

zpool 命令的 get 子命令用于查看 zpool 的属性。它包括 feature 属性，用于查看 feature flags 的当前状态。feature 属性的值是一个字符串，表示 feature flags 的当前状态。

```
$ zpool get all zroot | grep feature
zroot  feature@async_destroy  enabled  local
zroot  feature@empty_bpobj    active  local
zroot  feature@lz4_compress    active  local
...
```

1. 在创建 ZFS 池之前，需要确保磁盘已经正确格式化，并且没有文件系统。可以使用 `diskpart` 工具来管理磁盘。

2. 使用 `diskpart` 工具选择磁盘并初始化它。例如，对于磁盘 1，可以执行以下命令：

```

diskpart
list disk
select disk 1
format fs=ntfs

```

3. 创建 ZFS 池。使用 `zpool create` 命令来创建池。例如，创建一个名为 `myzpool` 的池，使用磁盘 1：

```

zpool create myzpool /dev/disk1

```

4. 验证池是否创建成功。使用 `zpool status` 命令来查看池的状态。

5. 在池上创建文件系统。使用 `zfs create` 命令来创建文件系统。例如，在 `myzpool` 池上创建一个名为 `myzfs` 的文件系统：

```

zfs create myzpool/myzfs

```

6. 挂载文件系统。使用 `mount` 命令来挂载文件系统。例如，将 `myzfs` 文件系统挂载到 `/mnt/myzfs`：

```

mount myzpool/myzfs /mnt/myzfs

```

7. 使用文件系统。现在，你可以使用 `/mnt/myzfs` 目录来存储数据。