# 在Linux[Filesystem in Userspace]上运行ZFS

Linux 与文件系统具有有趣的关系。因为 Linux 是开放式的，所以它往往是下一代文件系统和创新文件系统理念的关键开发平台。两个有趣的最新示例包括可大规模扩展的 Ceph 和连续快照文件系统 nilfs2（当然，主力文件系统，比如第四个扩展文件系统 [ext4] 的演化）。它还可以运行一些旧的文件系统 — DOS VFAT、Macintosh(HPFS)、VMS ODS-2 和 Plan-9 的远程文件系统协议。在 Linux 支持的所有文件系统中，有一个您可能会因其实现的功能而非常感兴趣：Oracle 的 Zettabyte 文件系统（Zettabyte File System，ZFS）。

ZFS 是由 Sun Microsystems（在 Jeff Bonwick 下）设计和开发的，在 2004 年首次公布，并在 2005 年集成到 Sun Solaris 中）。虽然将最流行的开放式操作系统与谈论最多的、功能最丰富的文件系统配对在一起是最理想的匹配，但是许可证问题限制了集成。Linux 受 GNU 公共许可证（General Public License，GPL）保护，而 ZFS 是遵循由 Sun 的通用开发和发布许可证（Common Development and Distribution License，CDDL）的。这些许可证协议的目标不同，各自的限制会有些冲突。所幸，这并不意味着您作为 Linux 用户不能享受 ZFS 及其提供的功能。

本文探究了在 Linux 中使用 ZFS 的两种方法。第一种使用了用户空间文件系统（Filesystem in Userspace，FUSE）系统来推动 ZFS 文件系统到用户空间以便避免许可证问题。第二种方法是一个 ZFS 本机端口，用于集成到 Linux 内核，同时避免知识产权问题。

**您可在何处找到 ZFS？**

现在，您不但可以在 OpenSolaris 内（也包括在 CDDL 下）本地发现 ZFS，而且还可以在具有补充许可的其他操作系统中发现 ZFS。例如，您可以在 FreeBSD 中发现 ZFS（自 2007 年开始）。ZFS 曾经是 Darwin 的一部分（Berkeley 软件分发（Berkeley Software Distribution [BSD]）、NeXTSTEP 和 CMU 的 Mach 3 微内核的衍生），但后来被删除了。

**ZFS 简介**

将 ZFS 称为文件系统有点名不副实，因为它在传统意义上不仅仅是个文件系统。ZFS 将逻辑卷管理器的概念与功能丰富的和可大规模扩展的文件系统结合起来。让我们开始先探索一些 ZFS 所基于的原则。首先，ZFS 使用池存储模型，而不是传统的基于卷的模型。这意味着 ZFS 视存储为可根据需要动态分配（和缩减）的共享池。这优于传统模型，在传统模型中，文件系统位于卷上，使用独立卷管理器来管理这些资产。ZFS 内嵌入的是重要功能集（如快照、即写即拷克隆、连续完整性检查和通过 RAID-Z 的数据保护）的实现。更进一步，可以在 ZFS 卷的顶端使用您自己最喜爱的文件系统（如 ext4）。这意味着您可以获得那些 ZFS 的功能，如独立文件系统中的快照（该文件系统可能并不直接支持它们）。

但是 ZFS 不只是组成有用文件系统的功能集合。相反，它是构建出色文件系统的集成和补充功能的集合。让我们来看看其中的一些功能，然后再看看它们的一些实际应用。

**存储池**

正如前面所讨论的，ZFS 合并了卷管理功能来提取底层物理存储设备到文件系统。ZFS 对存储池（称为 *zpools*）进行操作，而不是直接查看物理块设备，存储池构建自虚拟驱动器，可由驱动器或驱动器的一部分物理地进行表示。此外，可以动态构造这些池，甚至这些池正在活跃地使用时也可以。

**即写即拷**

ZFS 使用即写即拷模型来管理存储中的数据。虽然这意味着数据永远不会写入到位（从来没有被覆盖），而是写入新块并更新元数据来引用数据。即写即拷有利的原因有多个（不仅仅是因为它可以启用的快照和克隆等一些功能）。由于从来不覆盖数据，这可以更简单地确保存储永远不会处于不一致的状态（因为在新的写入操作完成以后较早的数据仍保留）。这允许 ZFS 基于事务，且更容易实现类似原子操作等的功能。

即写即拷设计的一个有趣的副作用是文件系统的所有写入都成为顺序写入（因为始终进行重新映射）。此行为避免存储中的热点并利用顺序写入的性能（比随机写入更快）。

**数据保护**

可以使用 ZFS 的众多保护方案之一来保护由虚拟设备组成的存储池。您不但可以跨两个或多个设备（RAID 1）来对池进行镜像，通过奇偶校验来保护该池（类似于 RAID 5），而且还可以跨动态带区宽度（后面详细介绍）来镜像池。基于池中设备数量，ZFS 支持各种不同的的奇偶校验方案。例如，您可以通过 RAID-Z (RAID-Z 1) 来保护三个设备；对于四个设备，您可以使用 RAID-Z 2（双重奇偶校验，类似于 RAID6）。对于更大的保护来说，您可以将 RAID-Z 3 用于更大数量的磁盘进行三重奇偶校验。

为提高速度（不存在错误检测以外的数据保护），您可以跨设备进行条带化（RAID 0）。您还可以创建条带化镜像（来镜像条带化设备），类似于 RAID 10。

ZFS 的一个有趣属性随 RAID-Z、即写即拷事务和动态条带宽度的组合而来。在传统的 RAID 5 体系结构中，所有磁盘都必须在条带内具有其自己的数据，或者条带不一致。因为没有方法自动更新所有磁盘，所以这可能产生众所周知的 *RAID 5 写入漏洞问题*（其中在 RAID 集的驱动器中条带是不一致的）。假设 ZFS 处理事务且从不需要写入到位，则写入漏洞问题就消除了。此方法的另外一个便捷性体现在磁盘出现故障且需要重建时。传统的 RAID 5 系统使用来自该集中其他磁盘的数据来重建新驱动器的数据。RAID-Z 遍历可用的元数据以便只读取有关几何学的数据并避免读取磁盘上未使用的空间。随着磁盘变得更大以及重建次数的增加，此行为变得更加重要。

**校验和**

虽然数据保护提供了在故障时重新生成数据的能力，但是这并不涉及处于第一位的数据的有效性。ZFS 通过为写入的每个块的元数据生成 32 位校验和（或 256 位散列）解决了此问题。在读取块时，将验证此校验和以避免静默数据损坏问题。在有数据保护（镜像或 AID-Z）的卷中，可自动读取或重新生成备用数据。

**用于完整性的标准方法**

T10 为被称为*数据完整性字段*（Data Integrity Field，DIF）的端对端完整性提供相似的机制。此机制建议一个字段，包含块的循环冗余校验和存储在磁盘上的其他元数据以避免静默数据损坏。DIF 的一个有趣属性是，您将在许多存储控制器中发现对它的硬件支持，所以此进程完全从主机处理器卸载。

在 ZFS 上校验和与元数据存储在一起，所以可以检测并更正错位写入 — 如果提供数据保护（RAID-Z）—。

**快照和克隆**

由于 ZFS 的即写即拷性质，类似快照和克隆的功能变得易于提供。因为 ZFS 从不覆盖数据而是写入到新的位置，所以可以保护较早的数据（但是在不重要的情况下被标记为删除以逆转磁盘空间）。*快照* 就是旧块的保存以便及时维护给定实例中的文件系统状态。这种方法也是空间有效的，因为无需复制（除非重新写入文件系统中的所有数据）。*克隆* 是一种快照形式，在其中获取可写入的快照。在这种情况下，由每一个克隆共享初始的未写入块，且被写入的块仅可用于特定文件系统克隆。

**可变块大小**

传统的文件系统由匹配后端存储（512 字节）的静态大小的块组成。ZFS 为各种不同的使用实现了可变块大小（通常大小达到 128KB，但是您可以变更此值）。可变块大小的一个重要使用是压缩（因为压缩时的结果块大小理想情况下将小于初始大小）。除了提供更好的存储网络利用外，此功能也使存储系统中的浪费最小化（因为传输更好的数据到存储需要更少的时间）。

在压缩以外，支持可变块大小还意味着您可以针对所期望的特定工作量优化块大小，以便改进性能。

**其他功能**

ZFS 并入了许多其他功能，如重复数据删除（最小化数据重复）、可配置的复制、加密、缓存管理的自适应更换缓存以及在线磁盘清理（标识并修复在不使用保护时可以修复的潜在错误）。它通过巨大的可扩展性来实现该功能，支持 16 千兆兆个字节的可寻址存储（264 字节）。

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-zfs/#ibm-pcon)

**在 Linux 上使用 ZFS**

现在您已经了解了 ZFS 背后的一些抽象概念，让我们在实践中看看其中的一些概念。本演示使用了 ZFS-FUSE。FUSE 是一种机制，允许您在没有内核代码（除 FUSE 内核模块和现有的文件系统代码以外）情况下在用户空间中实现文件系统。该模块为用户和文件系统实现提供从内核文件系统接口到用户空间的桥梁。首先，安装 ZFS-FUSE 包（下面的演示针对 Ubuntu）。

**安装 ZFS-FUSE**

安装 ZFS-FUSE 很简单，尤其是在使用 apt 的 Ubuntu 上。下面的命令行安装了您开始使用 ZFS-FUSE 所需的一切：

$ **sudo apt-get install zfs-fuse**

此命令行安装 ZFS-FUSE 和所有其他依赖包（ 我的也需要 libaiol），为新的程序包执行必要的设置并启动 zfs-fuse 守护进程。

**使用 ZFS-FUSE**

在此演示中，您使用环回设备以便在主机操作系统内将磁盘仿真为文件。要开始此操作，请通过 dd 实用程序（参见清单 1）创建这些文件（使用 /dev/zero 作为源）。在创建了四个磁盘映像之后，使用 losetup 将磁盘映像与环路设备关联在一起。

**清单 1. 使用 ZFS-FUSE 的设置**

$ **mkdir zfstest**

$ **cd zfstest**

$ **dd if=/dev/zero of=disk1.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 1.235 s, 54.3 MB/s

$ **dd if=/dev/zero of=disk2.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 0.531909 s, 126 MB/s

$ **dd if=/dev/zero of=disk3.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 0.680588 s, 98.6 MB/s

$ **dd if=/dev/zero of=disk4.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 0.429055 s, 156 MB/s

$ **ls**

disk1.img disk2.img disk3.img disk4.img

$ **sudo losetup /dev/loop0 ./disk1.img**

$ **sudo losetup /dev/loop1 ./disk2.img**

$ **sudo losetup /dev/loop2 ./disk3.img**

$ **sudo losetup /dev/loop3 ./disk4.img**

$

有了四台设备作为您的 ZFS 块设备（总大小 256MB），使用 zpool 命令来创建您的池。您可以使用 zpool 命令来管理 ZFS 存储池，不过您将看到，您可以将其用于各种其他目的。下面的命令要求通过四个设备创建 ZFS 存储池并通过 RAID-Z 提供数据保护。在此命令后为一个列表请求，以便提供您池中的数据（参见清单 2）。

**清单 2. 创建 ZFS 池**

$ **sudo zpool create myzpool raidz /dev/loop0 /dev/loop1 /dev/loop2 /dev/loop3**

$ **sudo zfs list**

NAME USED AVAIL REFER MOUNTPOINT

myzpool 96.5K 146M 31.4K /myzpool

$

您还可以研究池的一些属性，如清单 3 所示，其代表默认值。对于其他事项，您可以查看可用容量和已使用的部分。（为了简洁，此代码已经被压缩。）

**清单 3. 查看存储池的属性**

$ **sudo zfs get all myzpool**

NAME PROPERTY VALUE SOURCE

myzpool type filesystem -

myzpool creation Sat Nov 13 22:43 2010 -

myzpool used 96.5K -

myzpool available 146M -

myzpool referenced 31.4K -

myzpool compressratio 1.00x -

myzpool mounted yes -

myzpool quota none default

myzpool reservation none default

myzpool recordsize 128K default

myzpool mountpoint /myzpool default

myzpool sharenfs off default

myzpool checksum on default

myzpool compression off default

myzpool atime on default

myzpool copies 1 default

myzpool version 4 -

...

myzpool primarycache all default

myzpool secondarycache all default

myzpool usedbysnapshots 0 -

myzpool usedbydataset 31.4K -

myzpool usedbychildren 65.1K -

myzpool usedbyrefreservation 0 -

$

现在，让我们实际使用 ZFS 池。首先，在您的池中创建目录，然后在该目录中启用压缩（使用 zfs set 命令）。其次，将文件复制到其中。我已经选择了大小 120KB 左右的文件来查看 ZFS 压缩的效果。请注意您的池挂载在根目录上，因此就像您的根文件系统内的目录一样加以处理。一旦复制该文件，您就可以列出它来表示文件已存在（但与原来同样大小）。通过使用 dh 命令，您可以看到文件大小为原来的一半，这说明 ZFS 已经将其压缩。您还可以查看 compressratio 属性，了解您的池有多少已经被压缩（使用默认压缩程序，gzip）。清单 4 显示了该压缩。

**清单 4. 演示 ZFS 压缩**

$ **sudo zfs create myzpool/myzdev**

$ **sudo zfs list**

NAME USED AVAIL REFER MOUNTPOINT

myzpool 139K 146M 31.4K /myzpool

myzpool/myzdev 31.4K 146M 31.4K /myzpool/myzdev

$ **sudo zfs set compression=on myzpool/myzdev**

$ **ls /myzpool/myzdev/**

$ **sudo cp ../linux-2.6.34/Documentation/devices.txt /myzpool/myzdev/**

$ **ls -la ../linux-2.6.34/Documentation/devices.txt**

-rw-r--r-- 1 mtj mtj 118144 2010-05-16 14:17 ../linux-2.6.34/Documentation/devices.txt

$ **ls -la /myzpool/myzdev/**

total 5

drwxr-xr-x 2 root root 3 2010-11-20 22:59 .

drwxr-xr-x 3 root root 3 2010-11-20 22:55 ..

-rw-r--r-- 1 root root 118144 2010-11-20 22:59 devices.txt

$ **du -ah /myzpool/myzdev/**

60K /myzpool/myzdev/devices.txt

62K /myzpool/myzdev/

$ **sudo zfs get compressratio myzpool**

NAME PROPERTY VALUE SOURCE

myzpool compressratio 1.55x -

$

最后，让我们看看 ZFS 的自修复功能。请回想在您创建池时，您要求四个设备具有 RAID-Z。通过使用 zpool status 命令您可以检查池的状态, 如清单 5 所示。如清单所示，您可以看到池的元素（RAID-Z 1 以及四个设备）。

**清单 5. 检查池状态**

$ **sudo zpool status myzpool**

pool: myzpool

state: ONLINE

scrub: none requested

config:

NAME STATE READ WRITE CKSUM

myzpool ONLINE 0 0 0

raidz1 ONLINE 0 0 0

loop0 ONLINE 0 0 0

loop1 ONLINE 0 0 0

loop2 ONLINE 0 0 0

loop3 ONLINE 0 0 0

errors: No known data errors

$

现在，让我们强制执行一个错误到池中。对于此演示来说，转到后台并损坏组成设备的磁盘文件（disk4.img，通过 loop3 设备显示在 ZFS 中）。使用 dd 命令将整个设备清零（参见清单 6）。

**清单 6. 损坏 ZFS 池**

$ **dd if=/dev/zero of=disk4.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 1.84791 s, 36.3 MB/s

$

ZFS 目前未意识到损坏，但是您可以通过请求池的清理，强制池发现问题。如清单 7 所示，ZFS 现在认识到（loop3 设备的）损坏并建议操作以便替换该设备。还请注意在 ZFS 通过 RAID-Z 自我更正时，池仍然在线，您仍然可以访问您的数据。

**清单 7. 清理并检查池**

$ **sudo zpool scrub myzpool**

$ **sudo zpool status myzpool**

pool: myzpool

state: ONLINE

status: One or more devices could not be used because the label is missing or

invalid. Sufficient replicas exist for the pool to continue

functioning in a degraded state.

action: Replace the device using 'zpool replace'.

see: http://www.sun.com/msg/ZFS-8000-4J

scrub: scrub completed after 0h0m with 0 errors on Sat Nov 20 23:15:03 2010

config:

NAME STATE READ WRITE CKSUM

myzpool ONLINE 0 0 0

raidz1 ONLINE 0 0 0

loop0 ONLINE 0 0 0

loop1 ONLINE 0 0 0

loop2 ONLINE 0 0 0

loop3 UNAVAIL 0 0 0 corrupted data

errors: No known data errors

$ **wc -l /myzpool/myzdev/devices.txt**

3340 /myzpool/myzdev/devices.txt

$

根据建议，引入新的设备到您的 RAID-Z 集以便充当新的容器。首先创建新的磁盘映像并通过 losetup 将其表示为设备。请注意此过程类似于将新的物理磁盘添加到集。然后，您使用 zpool replace 用新的设备（loop4）交换已损坏的设备（loop3）。检查池状态，您可以看到新设备具有一条消息，指示其上重新构建了数据（称为 *resilvering*）以及移到那里的数据量。还请注意池仍保持在线，没有错误（对用户可见）。最后，再次清理池；在检查其状态以后，您将看不到存在问题，如清单 8 所示。

**清单 8. 使用 zpool replace 修复池**

$ **dd if=/dev/zero of=disk5.img bs=64M count=1**

1+0 records in

1+0 records out

67108864 bytes (67 MB) copied, 0.925143 s, 72.5 MB/s

$ **sudo losetup /dev/loop4 ./disk5.img**

$ **sudo zpool replace myzpool loop3 loop4**

$ **sudo zpool status myzpool**

pool: myzpool

state: ONLINE

scrub: resilver completed after 0h0m with 0 errors on Sat Nov 20 23:23:12 2010

config:

NAME STATE READ WRITE CKSUM

myzpool ONLINE 0 0 0

raidz1 ONLINE 0 0 0

loop0 ONLINE 0 0 0

loop1 ONLINE 0 0 0

loop2 ONLINE 0 0 0

loop4 ONLINE 0 0 0 59.5K resilvered

errors: No known data errors

$ **sudo zpool scrub myzpool**

$ **sudo zpool status myzpool**

pool: myzpool

state: ONLINE

scrub: scrub completed after 0h0m with 0 errors on Sat Nov 20 23:23:23 2010

config:

NAME STATE READ WRITE CKSUM

myzpool ONLINE 0 0 0

raidz1 ONLINE 0 0 0

loop0 ONLINE 0 0 0

loop1 ONLINE 0 0 0

loop2 ONLINE 0 0 0

loop4 ONLINE 0 0 0

errors: No known data errors

$

此简短演示探究了通过文件系统进行的卷管理的整合，并展示了管理 ZFS（即使是故障时）有多简单。

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-zfs/#ibm-pcon)

**其他 Linux-ZFS 可能性**

FUSE 中 ZFS 的优点是易于开始使用 ZFS，但是它的缺点是效率低。这种效率上的缺点是每个 I/O 都要求多个用户内核转换的结果。但是考虑到 ZFS 的流行，有另外一种提供更好性能的选项。

针对 Linux 内核的 ZFS 本地端口正在 Lawrence Livermore 国家实验室中处于顺利开发当中。虽然此端口仍然缺乏一些元素，如 ZFS 便携式操作系统接口（适用于 UNIX®） 层，但是这正在开发中。其端口提供大量有用的功能，尤其是当您有兴趣与 Lustre 一起使用 ZFS 时。（要了解更多细节，参阅 [参考资料](http://www.ibm.com/developerworks/cn/linux/l-zfs/#resources)。）

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-zfs/#ibm-pcon)

**前景展望**

希望本文已经引起您进一步挖掘 ZFS 的欲望。从早前的演示中，您可以容易地设置 ZFS 并在大多数 Linux 发行版中运行 — 甚至是在拥有一些限制的内核中。虽然快照和克隆等主题并未在此处演示，但是 [参考资料](http://www.ibm.com/developerworks/cn/linux/l-zfs/#resources) 部分提供了有关此主题的有趣的文章的链接。最后，Linux 和 ZFS 是最先进的技术，很难将它们分开。

**参考资料**

**学习**

* 来自 Jeff Bonwick 和 Bill More 的优秀演示文稿提供了 ZFS 的详细概述并解释了为什么它是 [文件系统中最后的工作](http://hub.opensolaris.org/bin/download/Community+Group+zfs/docs/zfslast.pdf)。
* 在各种针对 Solaris 和 ZFS 的 Oracle Web 站点中，您可以了解有关 ZFS 的更多信息。[OpenSolaris ZFS 社区站点](http://hub.opensolaris.org/bin/view/Community+Group+zfs/WebHome) 提供有关 ZFS 的有用信息以及何处可以了解更多信息。Wikipedia 还提供很好的、紧凑的 [ZFS 简介](http://en.wikipedia.org/wiki/ZFS)。您可以阅读 [RAID-Z from Jeff Bonwick](http://blogs.sun.com/bonwick/entry/raid_z) 和其通过传统的 RAID 5 所解决的特定问题。
* FUSE 为文件系统的开发和执行提供用户空间框架。虽然 FUSE 与 ZFS 一起使用（如本文 ZFS-FUSE 所示），但是它还被广泛地用作试验文件系统开发的方式。您可以在 [使用 FUSE 开发自己的文件系统](http://www.ibm.com/developerworks/cn/linux/l-fuse/)（Sumit Singh，developerWorks，2006 年 2 月）中了解有关 FUSE 和文件系统开发的更多信息。
* 虽然将 ZFS 集成到 Linux 中的最简单的方法之一是 Solaris 实现的直接端口，但是许可证的争论排除了此方法。您可以在 Wikipedia 中的 [GPL](http://en.wikipedia.org/wiki/GNU_General_Public_License) 和 [CDDL](http://en.wikipedia.org/wiki/CDDL) 中了解有关许可证的更多信息。
* [FreeBSD 手册](http://www.freebsd.org/doc/handbook/filesystems-zfs.html) 就 ZFS 在 BSD 中的应用做了很好的介绍。
* 除了在 FUSE 上运行 ZFS 以外，在 Linux 内核中具有一个 ZFS 的本地实现。[Linux 上的 ZFS](http://zfsonlinux.org/) 项目正在扩大且已经提供了令人印象深刻的功能集。
* 虽然 ZFS 对写入存储的每个块提供校验和，但是还有一个名为 *DIF* 的标准化 SCSI 端对端完整性方案。您可以在 [来自 Oracle 有关数据完整性的演示文稿](http://www.usenix.org/event/lsf07/tech/petersen.pdf) 或 [Linux 内核的发展](http://www.ibm.com/developerworks/cn/linux/l-kernel-advances/)（M. Tim Jones，developerWorks，2009 年 3 月）中了解有关 DIF 的更多信息。
* 对于任何已经阅读过 Tim 在 developerWorks 上的其他文章的人来说，您已经了解到他是一位文件系统的爱好者。查看有关 Linux 文件系统所有方面的以下文章：
  + [Linux 文件系统剖析](http://www.ibm.com/developerworks/cn/linux/l-linux-filesystem/)（2007 年 10 月）
  + [下一代 Linux 文件系统：NiLFS(2) 和 exofs](http://www.ibm.com/developerworks/cn/linux/l-nilfs-exofs/)（2009 年 10 月）
  + [Ceph：一个 Linux PB 级分布式文件系统](http://www.ibm.com/developerworks/cn/linux/l-ceph/)（2010 年 5 月）
  + [剖析 ext4](http://www.ibm.com/developerworks/cn/linux/l-anatomy-ext4/)（2009 年 2 月）
  + [Linux 日志文件系统剖析](http://www.ibm.com/developerworks/cn/linux/l-journaling-filesystems/)（2008 年 6 月）
  + [Linux 虚拟系统文件交换器剖析](http://www.ibm.com/developerworks/cn/linux/l-virtual-filesystem-switch/)（2009 年 8 月）

**在 Linux 上运行 ZFS**

# 在 Linux 上运行 ZFS

# 在 Linux 上运行 ZFS