# GlusterFS中文文档

王国强整理于2019/1/7

# 一、简介

### 1.1 什么是Gluster？

Gluster是一个可扩展的分布式文件系统，可将来自多个服务器的磁盘存储资源聚合到一个全局命名空间中。可根据您的存储消耗需求快速配置额外的存储。它将自动故障转移作为主要功能。

### 1.2 优点

1. 可扩展到几千兆字节
2. 处理成千上万的客户
3. POSIX兼容
4. 使用商用硬件
5. 可以使用任何支持扩展属性的ondisk文件系统
6. 可使用NFS和SMB等行业标准协议访问
7. 提供复制，配额，地域复制，快照和位置检测
8. 允许针对不同工作负载的优化
9. 开源

企业可以根据需求扩展规模、容量、性能，没有供应商锁定、跨企业预置型、公共

云和混合环境。Gluster用于数千家企业的生产，涵盖媒体，医疗保健，政府，教育，Web 2.0和金融服务。

# 二、安装GlusterFS - 快速入门指南

本文件的目的

本文档旨在提供首次设置GlusterFS的分步指南。出于本指南的目的，需要使用Fedora 26（或更高版本）虚拟机实例。

如果您想要更详细的演练，其中包含使用不同方法（在本地虚拟机，EC2和裸设备中）和不同发行版进行安装的说明，请查看安装指南。

使用Ansible部署和管理GlusterFS

如果您已经是Ansible用户，并且更习惯使用Ansible设置分布式系统，我们建议您跳过所有这些并转移到gluster-ansible存储库，它提供了大部分细节以使系统运行得更快。

## 2.1 部署GlusterFS

#### 第1步 - 至少有三个节点

1）Fedora 26（或更高版本）上名为“server1”，“server2”和“server3”的3个节点；

2）一个有效的网络连接；

3）每个虚拟机上至少有两个虚拟磁盘，一个用于操作系统安装，另一个用于服务GlusterFS存储（sdb）。这将模拟真实世界的部署，您可能希望将GlusterFS存储与操作系统安装分开。

4）在每台服务器上设置NTP，以便在文件系统之上正常运行许多应用程序。

注意：GlusterFS将其动态生成的配置文件存储在/var/lib/glusterd。如果在任何时候GlusterFS无法写入这些文件（例如，当后备文件系统已满时），它将至少导致系统的不稳定行为; 或者更糟糕的是，让您的系统完全脱机。建议为目录创建单独的分区，/var/log以减少发生这种情况的可能性。

#### 第2步 - 格式化并安装块

在所有节点上执行此步骤，“server {1,2,3}”

注意：我们将使用XFS文件系统作为后端块。但Gluster旨在处理任何支持扩展属性的文件系统。

以下示例假定brick将驻留在/ dev / sdb1上。

mkfs.xfs -i size=512 /dev/sdb1

mkdir -p /data/brick1

echo '/dev/sdb1 /data/brick1 xfs defaults 1 2' >> /etc/fstab

mount -a && mount

您现在应该看到sdb1挂载在/ data / brick1

#### 第3步 - 安装GlusterFS

安装软件

yum install glusterfs-server

启动GlusterFS管理守护程序：

service glusterd start

service glusterd status

glusterd.service - LSB: glusterfs server

Loaded: loaded (/etc/rc.d/init.d/glusterd)

Active: active (running) since Mon, 13 Aug 2012 13:02:11 -0700; 2s ago

Process: 19254 ExecStart=/etc/rc.d/init.d/glusterd start (code=exited, status=0/SUCCESS)

CGroup: name=systemd:/system/glusterd.service

├ 19260 /usr/sbin/glusterd -p /run/glusterd.pid

├ 19304 /usr/sbin/glusterfsd --xlator-option georep-server.listen-port=24009 -s localhost...

└ 19309 /usr/sbin/glusterfs -f /var/lib/glusterd/nfs/nfs-server.vol -p /var/lib/glusterd/...

#### 第4步 - 配置防火墙

节点上的gluster进程需要能够相互通信。要简化此设置，请在每个节点上配置防火墙以接受来自其他节点的所有流量。

iptables -I INPUT -p all -s <ip-address> -j ACCEPT

其中ip-address是另一个节点的地址。

#### 第5步 - 配置信任池

来自“server1”

gluster peer probe server2

gluster peer probe server3

注意：使用主机名时，需要从另一台服务器探测第一 台服务器以设置其主机名。

来自“server2”

gluster peer probe server1

注意：建立此池后，只有受信任的成员才能将新服务器探测到池中。新服务器无法探测池，必须从池中进行探测。

检查server1上的对等体状态

gluster peer status

你应该看到这样的东西（UUID会有所不同）

Number of Peers: 2

Hostname: server2

Uuid: f0e7b138-4874-4bc0-ab91-54f20c7068b4

State: Peer in Cluster (Connected)

Hostname: server3

Uuid: f0e7b138-4532-4bc0-ab91-54f20c701241

State: Peer in Cluster (Connected)

#### 第6步 - 设置GlusterFS卷

在所有服务器上：

mkdir -p /data/brick1/gv0

从任何单一服务器：

gluster volume create gv0 replica 3 server1:/data/brick1/gv0 server2:/data/brick1/gv0 server3:/data/brick1/gv0

gluster volume start gv0

确认卷显示“已启动”：

gluster volume info

您应该看到类似的内容（卷ID会有所不同）：

Volume Name: gv0

Type: Replicate

Volume ID: f25cc3d8-631f-41bd-96e1-3e22a4c6f71f

Status: Started

Snapshot Count: 0

Number of Bricks: 1 x 3 = 3

Transport-type: tcp

Bricks:

Brick1: server1:/data/brick1/gv0

Brick2: server2:/data/brick1/gv0

Brick3: server3:/data/brick1/gv0

Options Reconfigured:

transport.address-family: inet

注意：如果卷未显示“已启动”，/var/log/glusterfs/glusterd.log则应检查下面的文件 以便调试和诊断情况。可以在配置的一个或所有服务器上查看这些日志。

#### 第7步 - 测试GlusterFS卷

对于此步骤，我们将使用其中一个服务器来装入卷。通常，您可以从外部计算机（称为“客户端”）执行此操作。由于使用此方法需要在客户端计算机上安装其他软件包，因此我们将使用其中一个服务器作为首先进行测试的简单位置，就像它是“客户端”一样。

mount -t glusterfs server1:/gv0 /mnt

for i in `seq -w 1 100`; do cp -rp /var/log/messages /mnt/copy-test-$i; done

首先，检查客户端挂载点：

ls -lA /mnt/copy\* | wc -l

您应该看到返回100个文件。接下来，检查每台服务器上的GlusterFS块安装点：

ls -lA /data/brick1/gv0/copy\*

您应该使用我们在此处列出的方法在每台服务器上看到100个文件。如果没有复制，在仅分发卷（此处未详述）中，您应该在每个卷上看到大约33个文件。

## 2.2 安装Gluster

对于基于RPM的分发，如果您将使用InfiniBand，请将glusterfs RDMA包添加到安装中。对于基于RPM的系统，yum / dnf用作安装方法以满足外部依赖性，例如compat-readline5

### 2.2.1 对于Debian

将GPG密钥添加到apt：

wget -O - http://download.gluster.org/pub/gluster/glusterfs/LATEST/rsa.pub | apt-key add -

如果rsa.pub在上述位置不可用，请在此处查看https://download.gluster.org/pub/gluster/glusterfs/3.12/rsa.pub并将GPG密钥添加到apt：

wget -O - https://download.gluster.org/pub/gluster/glusterfs/3.12/rsa.pub | apt-key add -

添加来源：

DEBID=$(grep 'VERSION\_ID=' /etc/os-release | cut -d '=' -f 2 | tr -d '"')

DEBVER=$(grep 'VERSION=' /etc/os-release | grep -Eo '[a-z]+')

DEBARCH=$(dpkg --print-architecture)

echo deb https://download.gluster.org/pub/gluster/glusterfs/LATEST/Debian/${DEBID}/${DEBARCH}/apt ${DEBVER} main > /etc/apt/sources.list.d/gluster.list

更新包列表：

apt-get update

安装：

apt-get install glusterfs-server

### 2.2.2 对于Ubuntu

安装软件：

sudo apt-get install software-properties-common

然后添加社区GlusterFS PPA：

sudo add-apt-repository ppa:gluster/glusterfs-3.8

sudo apt-get update

最后，安装包：

sudo apt-get install glusterfs-server

*注意：Ubuntu 12.04 LTS，12.10,13.10和14.04 LTS存在包*

### 2.2.3 对于Red Hat / CentOS

CentOS和其他RHEL克隆的RPM可从CentOS Storage SIG镜像获得。

有关更多安装详细信息，请参阅CentOS Storage SIG的[Gluster快速入门指南](https://wiki.centos.org/SpecialInterestGroup/Storage/gluster-Quickstart)。

对于Fedora

安装Gluster包：

dnf install glusterfs-server

完成安装后，您可以继续进行[配置](https://docs.gluster.org/en/latest/Install-Guide/Configure/)部分。

对于Arch Linux

安装Gluster包：

pacman -S glusterfs

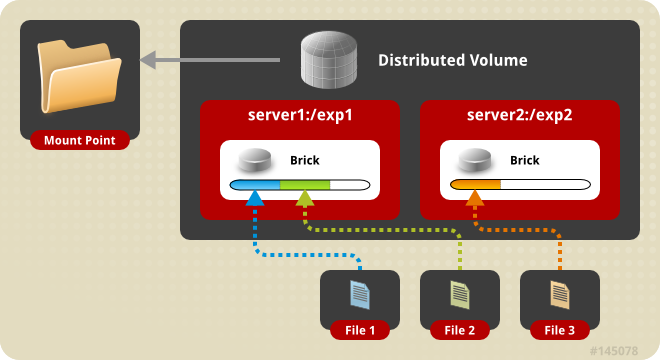
# 三、架构

## 3.1、卷的类型

卷是块的集合，大多数gluster文件系统操作都发生在卷上。Gluster文件系统根据要求支持不同类型的卷。有些卷适用于扩展存储大小，有些用于提高性能，有些用于提高性能。

### 3.1.1 **分布式Glusterfs卷**

这是默认的glusterfs卷，即，如果未指定卷的类型，则在创建卷时，默认选项是创建分布式卷。这里，文件分布在卷中的各个块中。因此file1可能只存储在brick1或brick2中，但不能存储在两者中。因此没有数据冗余。这种存储卷的优点是扩展容易和廉价。然而，这也意味着brick故障将导致数据完全丢失，并且必须依靠底层硬件来保护数据丢失。



**创建分布式卷**

**gluster volume create NEW-VOLNAME [transport [tcp | rdma | tcp，rdma]] NEW-BRICK ......**

**例如**，使用TCP创建具有四个存储服务器的分布式卷。

gluster volume create test-volume server1:/exp1 server2:/exp2 server3:/exp3 server4:/exp4

Creation of test-volume has been successful

Please start the volume to access data

**显示卷信息**

#gluster volume info

Volume Name: test-volume

Type: Distribute

Status: Created

Number of Bricks: 4

Transport-type: tcp

Bricks:

Brick1: server1:/exp1

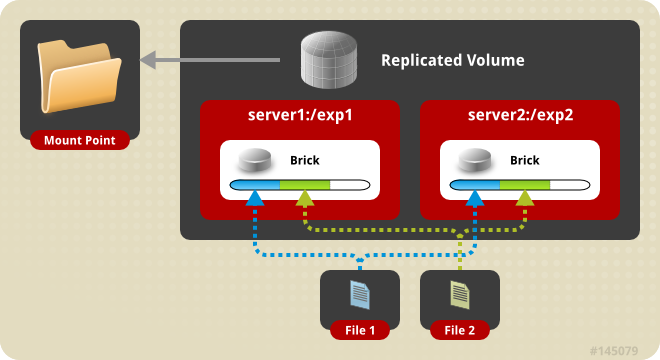
Brick2: server2:/exp2

Brick3: server3:/exp3

Brick4: server4:/exp4

### 3.1.2 **复制Glusterfs卷**

在本卷中，我们克服了分布式卷中面临的数据丢失问题。在这里，所有brick都保留了精确的数据副本。在创建卷时，客户端可以决定卷中的副本数。所以我们需要至少有两块块来创建一个包含2个复制品或至少3个块的卷来创建3个复制品。这种卷的一个主要优点是，即使一块砖出现故障，仍然可以从其复制的块访问数据。这样的卷的优点是可靠性和数据冗余。



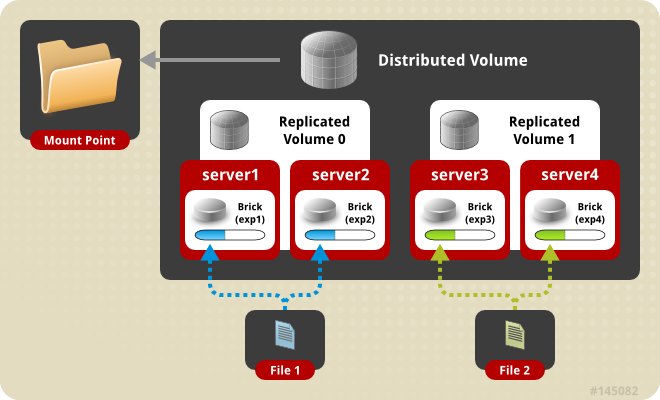
**创建复制卷**

**gluster volume create NEW-VOLNAME [replica COUNT] [transport [tcp | rdma | tcp，rdma]] NEW-BRICK ......**

例如，要创建具有两个存储服务器的复制卷：gluster volume create test-volume replica 2 transport tcp server1:/exp1 server2:/exp2

### 3.1.3 **分布式复制Glusterfs卷**

此卷文件分布在复制的块集中。块数量必须是副本数量的倍数。此外，我们指定砖的顺序很重要，因为相邻的砖成为彼此的复制品。当需要由于冗余和扩展存储而导致的数据高可用性时，使用这种类型的卷。因此，如果有8个块和副本计数2，则前两个块将成为彼此的复制品，然后是接下来的两个块，依此类推。该卷表示为4×2。同样，如果有8块和4块复制品，则4块成为彼此的复制品，我们将该卷表示为2x4。



**创建分布式复制卷：**

**#gluster volume create NEW-VOLNAME [replica COUNT] [transport [tcp | rdma | tcp，rdma]] NEW-BRICK ......**

**例如**，带有双向镜像的四个节点分布式（复制）卷：

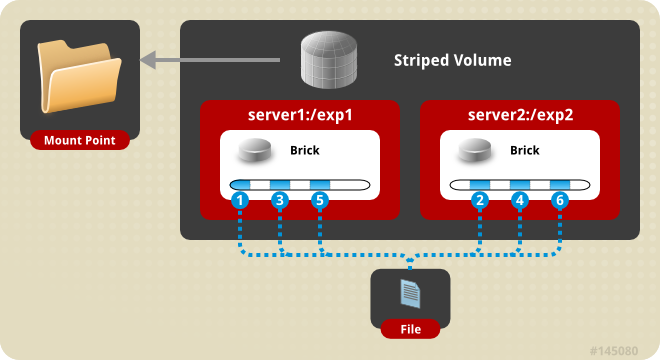
# gluster volume create test-volume replica 2 transport tcp server1:/exp1 server2:/exp2 server3:/exp3 server4:/exp4

Creation of test-volume has been successful

Please start the volume to access data

### 3.1.4 **条带Glusterfs卷**

考虑将大型文件存储在brick中，许多客户端经常同时访问该块。这将导致单块砖上的负载过大，从而降低性能。在条带化体积中，数据在将其分成不同的条带后存储在brick中。因此，大文件将被分成较小的块（等于卷中的块数），每个块存储在一块砖中。现在可以分配负载，并且可以更快地获取文件，但不提供数据冗余。



**创建条带卷**

#gluster volume create NEW-VOLNAME [stripe COUNT] [transport [tcp | dma | tcp,rdma]] NEW-BRICK...

**例如**，要跨两个存储服务器创建条带卷：

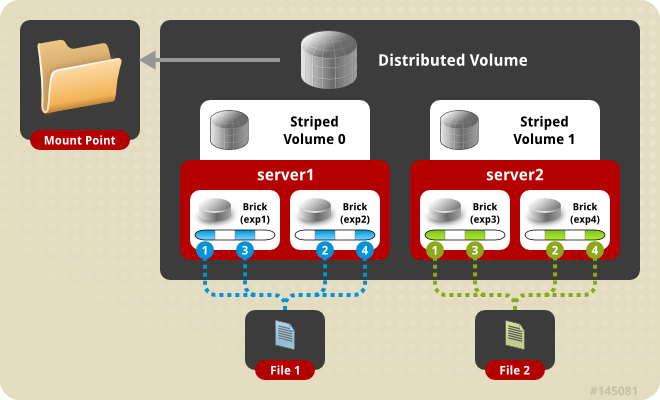
# gluster volume create test-volume stripe 2 transport tcp server1:/exp1 server2:/exp2

Creation of test-volume has been successful

Please start the volume to access data

### 3.1.5 **分布式条带Glusterfs卷**

类似于条纹Glusterfs卷，除了条纹现在可以分布在更多数量的brick上。但是块的数量必须是条纹数量的倍数。因此，如果我们想要增加卷大小，我们必须在多个条带数中添加块。



**创建分布式条带卷：**

**#gluster volume create NEW-VOLNAME [stripe COUNT] [transport [tcp | rdma | tcp，rdma]] NEW-BRICK ......**

例如，要在八个存储服务器上创建分布式条带卷：

# gluster volume create test-volume stripe 4 transport tcp

 server1:/exp1 server2:/exp2 server3:/exp3 server4:/exp4 server5:/exp5 server6:/exp6 server7:/exp7 server8:/exp8

Creation of test-volume has been successful

Please start the volume to access data.

### 3.1.6 分散Glusterfs卷

**创建分散的卷**

分散的卷基于纠删码。它会在卷中的多个brick中对文件的编码数据进行条带化，并添加一些冗余。您可以使用分散的卷来获得可配置的可靠性级别，同时最大限度地减少空间浪费。

**冗余**

每个分散的卷都具有在创建卷时定义的冗余值。此值确定在不中断卷操作的情况下可以丢失多少块砖。它还使用以下公式确定卷的可用空间量：

<Usable size> = <Brick size> \* (#Bricks - Redundancy)

分散组的所有砖应具有相同的容量，否则当最小的砖变满时，分散组中不允许有额外的数据。

值得注意的是，具有3个brick和冗余1的配置将比具有10个brick和冗余1（90％）的配置具有更少的可用空间（占总物理空间的66.7％）。然而，第一个比第二个更安全（大致是第二个配置失败的概率，如果比第一个配置大4.5倍）。

例如，即使两块砖无法进入，由6块4TB砖和2块冗余组成的分散体积也将完全可操作。但是，第三个无法访问的brick会降低音量，因为它无法读取或写入。卷的可用空间将等于16TB。

GlusterFS中的纠删码的实现将冗余限制为小于#Bricks / 2的值（或等效地，冗余\* 2 <#Bricks）。冗余等于brick数量的一半将几乎相当于副本2卷，并且在这种情况下复制卷可能表现更好。

**Optimal volumes**

纠删码在性能方面最糟糕的事情之一是RMW（读 - 修改 - 写）循环。擦除代码以特定大小的块运行，并且不能与较小的块一起使用。这意味着如果用户发出未填满整个块的文件的一部分，则需要从文件的当前内容中读取剩余部分，合并它们，计算更新的编码块，最后，写出结果数据。

这会增加延迟，从而降低性能。某些GlusterFS性能xlator可以帮助减少甚至消除某些工作负载的此问题，但在针对特定用例使用分散卷时应该考虑到这一点。

分散卷的当前实现使用的块大小取决于块数和冗余：512 \*（#Bricks - redundancy）字节。该值也称为条带大小。

使用为条带大小提供2的幂的#Bridge /冗余组合将使分散卷在大多数工作负载中表现更好，因为以两个为一的块（例如数据库，虚拟机和许多块）写入信息更为典型应用程序）。

这些组合被认为是最佳的。

例如，具有6块砖和冗余2的配置将具有512 \*（6-2）= 2048字节的条带大小，因此它被认为是最佳的。具有7块砖和冗余2的配置将具有2560字节的条带大小，对于许多写入需要RMW循环（当然这总是取决于用例）。

**创建分散的卷**

# gluster volume create [disperse [<count>]] [redundancy <count>] [transport tcp | rdma | tcp,rdma]

可以通过指定分散集中的砖数，通过指定冗余砖的数量或两者来创建分散的卷。

如果未指定分散，或者缺少<count>，则整个卷将被视为由命令行中枚举的所有砖组成的单个分散集。

如果未指定冗余，则会自动计算为最佳值。如果此值不存在，则假定为“1”并显示警告消息：

# gluster volume create test-volume disperse 4 server{1..4}:/bricks/test-volume

There isn't an optimal redundancy value for this configuration. Do you want to create the volume with redundancy 1 ? (y/n)

在自动计算冗余并且不等于“1”的所有情况下，都会显示一条警告消息：

# gluster volume create test-volume disperse 6 server{1..6}:/bricks/test-volume

The optimal redundancy for this configuration is 2. Do you want to create the volume with this value ? (y/n)

冗余必须大于0，并且砖的总数必须大于2 \* 冗余。这意味着分散的体积必须至少有3块砖。

如果未指定传输类型，则使用tcp作为缺省值。如果需要，您还可以设置其他选项，就像在其他卷类型中一样。

注意：

1）确保在尝试安装卷之前启动卷，或者在挂载挂起后启动客户端操作。

2）如果同一对等体上存在多个分散集块，则GlusterFS将无法创建分散卷。

# gluster volume create <volname> disperse 3 server1:/brick{1..3}

volume create: <volname>: failed: Multiple bricks of a replicate volume are present on the same server. This setup is not optimal.

Do you still want to continue creating the volume? (y/n)

force如果要在这种情况下创建卷，请使用命令末尾的选项。

### 3.1.7 分布式分散卷

分布式分散卷等效于分布式复制卷，但使用分散的子卷而不是复制的子卷。

**创建分布式分散卷：**

# gluster volume create disperse <count> [redundancy <count>] [transport tcp | rdma | tcp,rdma]

要创建分布式分散卷，必须使用disperse关键字和<count>，并且命令行中指定的brick数必须是分散计数的倍数。

冗余与分散卷中的完全相同。

如果未指定传输类型，则使用tcp作为缺省值。如果需要，您还可以设置其他选项，就像在其他卷类型中一样。

注意：

1）确保在尝试安装卷之前启动卷，或者在挂载挂起后启动客户端操作。

2）如果同一对等体上存在多个分散集块，则GlusterFS将无法创建分布式分散卷。

# gluster volume create <volname> disperse 3 server1:/brick{1..6}

volume create: <volname>: failed: Multiple bricks of a replicate volume are present on the same server. This setup is not optimal.

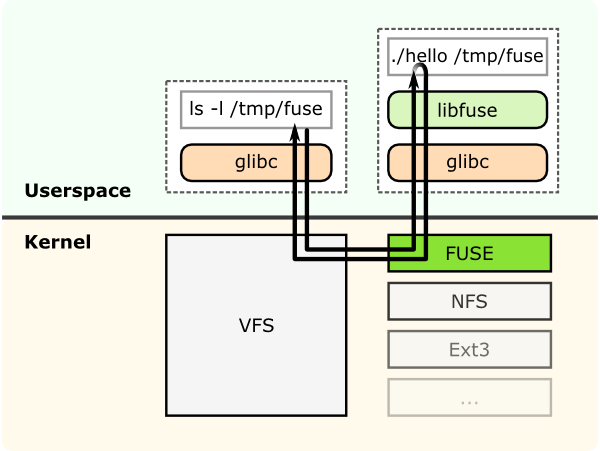
Do you still want to continue creating the volume? (y/n)

force如果要在这种情况下创建卷，请使用命令末尾的选项。

## 3.2、FUSE

GlusterFS是一个用户空间文件系统。这是GlusterFS开发人员最初做出的决定，因为将模块放入Linux内核是一个非常漫长而艰难的过程。

作为用户空间文件系统，GlusterFS与内核VFS交互，利用FUSE（用户空间中的文件系统）。很长一段时间，用户空间文件系统的实现被认为是不可能的。FUSE是为此而开发的解决方案。FUSE是一个内核模块，支持内核VFS和非特权用户应用程序之间的交互，它有一个可以从用户空间访问的API。使用此API，可以使用您喜欢的几乎任何语言编写任何类型的文件系统，因为FUSE和其他语言之间存在许多绑定。



FUSE的结构图

这显示了一个文件系统“hello world”，它被编译为创建二进制“hello”。它使用文件系统安装点/ tmp / fuse执行。然后用户在挂载点/ tmp / fuse上发出命令ls -l。此命令通过glibc到达VFS，并且由于mount / tmp / fuse对应于基于FUSE的文件系统，因此VFS将其传递给FUSE模块。FUSE内核模块在通过用户空间（libfuse）中的glibc和FUSE库后联系实际的文件系统二进制文件“hello”。结果由“hello”通过相同的路径返回并到达ls -l命令。

FUSE内核模块和FUSE库（libfuse）之间的通信是通过打开/ dev / fuse获得的特殊文件描述符。可以多次打开此文件，并将获取的文件描述符传递给mount syscall，以使描述符与挂载的文件系统相匹配。

## 3.3、Translator

Translators是GlusterFS提供的一种强大文件系统功能扩展机制，这一设计思想借鉴于GNU/Hurd微内核操作系统。GlusterFS中所有的功能都通过Translator机制实现，运行时以动态库方式进行加载，服务端和客户端相互兼容。

GlusterFS3.1.X中，主要包括以下几类Translator：

Cluster：存储集群分布，目前有AFR, DHT, Stripe三种方式

Debug：跟踪GlusterFS内部函数和系统调用

Encryption：简单的数据加密实现

Features：访问控制、锁、Mac兼容、静默、配额、只读、回收站等

Mgmt：弹性卷管理

Mount：FUSE接口实现

Nfs：内部NFS服务器

Performance：io-cache, io-threads, quick-read, read-ahead, stat-prefetch, sysmlink-cache,

write-behind等性能优化

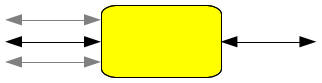
Protocol：服务器和客户端协议实现

Storage：底层文件系统POSIX接口实现

### **3.3.1 什么是“Translator”？**

1）Translator将来自用户的请求转换为存储请求。

\*一对一，一对多，一对零（例如缓存）



2）转换器可以通过以下方式修改请求：

将一种请求类型转换为另一种请求类型（在转换器之间的请求传输期间） 修改路径，标志，甚至数据（例如加密）

3）转换器可以拦截或阻止请求。（例如访问控制）

或者产生新的请求（例如预取）

### 3.3.2 Translators**如何运作？**

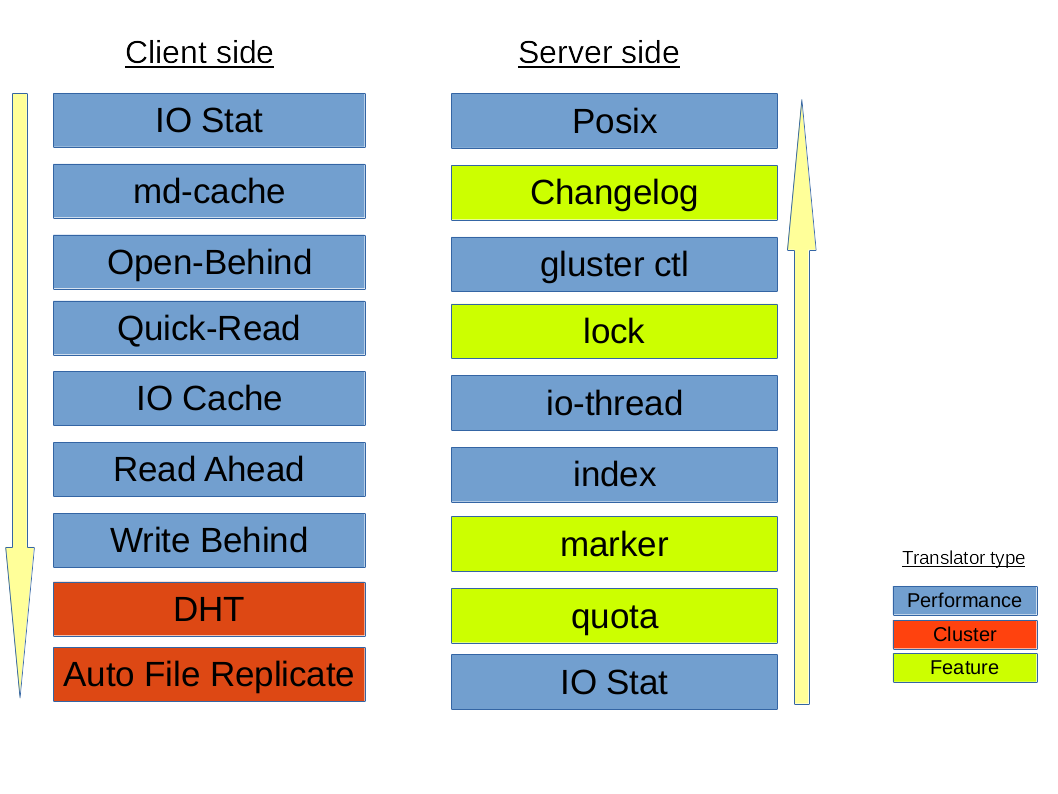
1. 共享对象；
2. 根据'volfile'动态加载；dlopen / dlsync 设置指向parent / children的指针 调用init（构造 函数）通过​​fops调用IO函数。
3. 验证/通过选项等的约定；
4. translator的配置（由于GlusterFS 3.1）是通过gluster命令行界面（cli）管理的，因此您无需知道将translator组合在一起的顺序。

### 3.3.3 Translator的类型

已知的Translators及其当前状态。

| **Translator类型** | **功能** |
| --- | --- |
| 存储 | 最低级别的转换器，存储和访问本地文件系统中的数据。 |
| 调试 | 提供错误和调试的接口和统计信息。 |
| 集群 | 处理与brick和节点的写入和读取相关的数据分发和复制。 |
| 加密 | 扩展转换器，用于存储数据的即时加密/解密。 |
| 协议 | 用于客户端/服务器通信协议的扩展转换器。 |
| 性能 | 调整转换器以调整工作负载和I / O配置文件。 |
| 绑定 | 添加可扩展性，例如Jeff Darcy编写的Python接口，用于扩展与GlusterFS的API交互。 |
| 系统 | 系统访问转换器，例如与文件系统访问控制接口。 |
| 调度 | I / O调度程序，用于确定如何跨集群系统分发新的写入操作。 |
| 特征 | 添加其他功能，例如配额，过滤器，锁等。 |

vol文件中translator的默认/一般层次结构：



所有连接在一起执行功能的转换器称为图形。左侧转换器包括**客户端栈**。右侧转换器包括**服务器栈**。

**glusterfs Translator可以细分为多个类别，但两个重要的类别是：Cluster和Performance Translator：**

数据/请求必须经历的最重要的一个和第一个**Translator**是**Fuse Translator**，属于**Mount**

**Translators**类别 。

**Cluster** Translators：

\* DHT（分布式哈希表）—— Distributed Hash Table

\* AFR（文件自动复制）—— Automatic File Replication

**Performance** Translators：

\* io-cache

\* io-threads

\* md-cache

\* OB（后读取）

\* QR（快速读取）

\* ra（预读）

\* wb（后写）

其他Feature Translators包括：

\*更改日志

\* locks ：GlusterFS具有锁定转换器，它提供以下内部锁定操作  inodelk，  entrylk被afr用来实现对彼此冲突的文件或目录的操作同步。

\*标记

\*配额

**Debug Translators**

\* trace - To trace the error logs generated during the communication amongst the translators.

\* io-stats

## 3.4 DHT（分布式哈希表）Translator

### **3.4.1 什么是DHT？**

DHT（Distributed Hash Table）是GlusterFS如何在多个服务器之间聚合容量和性能的真正核心。它的职责是将每个文件放在其子卷中的一个上，不像复制（将副本放在所有子卷上）或条带化（将块放在其所有子卷上）。这是一个路由功能，而不是分割或复制。

### **3.4.2 DHT如何运作**？

DHT中使用的基本方法是一致哈希。每个子卷（块）在32位散列空间内分配一个范围，整个范围没有孔或重叠。然后通过对其名称进行散列，在同一空间中为每个文件分配一个值。每一个砖将具有指定的范围，包括文件的哈希值，因此文件“应该”在相应的块上。但是，在许多情况下情况并非如此，例如，当在创建文件后块集发生更改，或当块接近满时。DHT的大部分复杂性涉及这些特殊情况，我们将在稍后讨论。

当您打开文件时，distribute translator会提供一条信息来查找您的文件，即文件名。要确定该文件的位置，translator通过散列算法运行文件名，以便将该文件名转换为数字。

**DHT哈希值赋值的一些观察**：

散列范围到brick的分配由存储在目录上的扩展属性确定，因此分布是特定于目录的。

一致性哈希通常被认为是围绕圆圈的散列，但在GlusterFS中它更线性。没有必要在零处“环绕”，因为在零处总是有一个中断（在一个块的范围和另一个块的范围之间）。

如果缺少块，则哈希空间中会有一个洞。更糟糕的是，如果在块处于脱机状态时重新分配哈希范围，则某些新范围可能会与存储在该块上的（现在已过时）范围重叠，从而对文件的位置产生一些混淆。

## 3.5 AFR Translator

GlusterFS中的自动文件复制（AFR—Automatic File Replication）Translator利用扩展属性来跟踪文件操作。它负责跨块复制数据。

### 3.5.1 AFR的作用

包括以下内容：

1. 保持复制一致性（即两块上的数据应该相同，即使在多个应用程序/挂载点并行运行同一文件/目录上的操作的情况下，只要副本集中的所有块都已启动）。
2. 只要存在至少一个具有正确数据的块，就提供一种在出现故障时恢复数据的方法。
3. 为read / stat / readdir等提供新数据。

## 3.4、地域复制

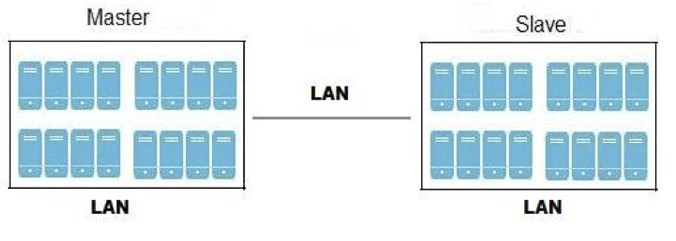
地域复制提供跨地理位置不同位置的异步数据复制，并在Glusterfs 3.2中引入。它主要在WAN上工作，用于复制整个卷，而不像AFR那样是集群内复制。这主要用于备份整个数据以进行灾难恢复。

地域复制使用主从模型，从而在**Master**（GlusterFS卷）和**Slave**（可以是本地目录或GlusterFS卷）之间进行复制。Slave使用SSH隧道访问本地目录或卷。

地域复制通过局域网（LAN），广域网（WAN）和Internet提供增量复制服务。

**1）通过LAN进行地域复制**

您可以配置地域复制以通过局域网镜像数据。



**2）通过WAN进行地域复制**

您可以配置地域复制以通过广域网复制数据。



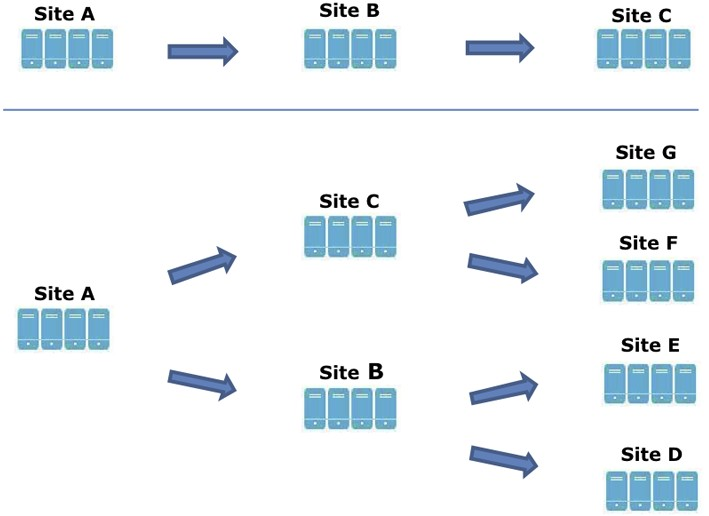
**3）通过Internet进行地域复制**

您可以配置地域复制以通过Internet镜像数据。



**4）多站点级联地域复制**

您可以配置地域复制，以跨多个站点以级联方式镜像数据。



异步复制数据主要有两个方面：

1）**更改检测** - 这些包括文件操作必需的详细信息。有两种方法可以同步检测到的更改：

a）更改日志 ： 更改日志是一个translator，它记录发生fops的必要的详细信息。这些更改可以以二进制格式或ASCII格式编写。每个类别有三个类别，由特定的更改日志格式表示。所有三种类别都记录在一个changelog文件中。

**Entry** ： create（），mkdir（），mknod（），symlink（），link（），rename（），unlink（），rmdir（）

**数据** ：write（），writev（），truncate（），ftruncate（）

**Meta** ： setattr（），fsetattr（），setxattr（），fsetxattr（），removexattr（），fremovexattr（）

为了记录操作类型和接受的实体，使用类型标识符。通常，执行操作的实体将由路径名标识，但我们选择使用GlusterFS内部文件标识符（GFID）（因为GlusterFS支持基于GFID的后端，并且路径名字段可能并不总是有效以及其他原因，超出了本文档的范围）。因此，三种操作的记录格式可归纳如下：

Entry ： GFID + FOP +模式+ UID + GID + PARGFID / BNAME [PARGFID / BNAME]

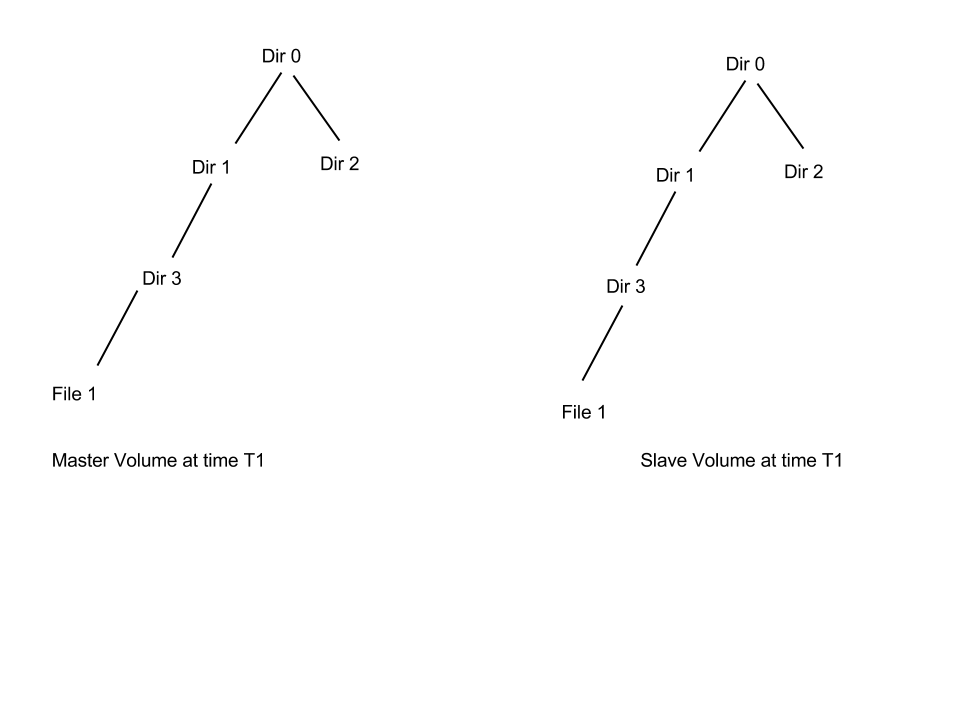
Meta ：文件的GFID

Data ：文件的GFID

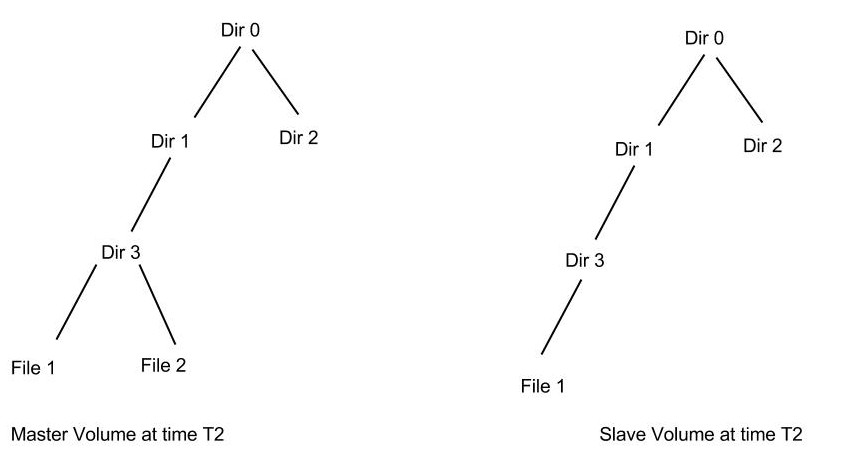
GFID类似于inodes。Data和Meta fops记录执行操作实体的GFID，从而记录inode上的数据/元数据更改。Entry fops至少记录一组六或七条记录（取决于操作类型），足以识别实体所经历的操作类型。通常，此记录包括实体的GFID，文件操作的类型（它是一个整数[在Glusterfs中使用的枚举值]）以及父GFID和basename（类似于父inode和basename）。

更改日志文件在特定时间间隔后翻转。然后，我们对文件执行处理操作，例如将其转换为可理解/可读的格式，保留更改日志的私有副本等。然后，库将使用这些日志并提供应用程序请求。

b）Xsync ：Marker Translator为每个文件和目录维护扩展属性“xtime”。每当发生任何更新时，它都会更新该文件及其所有祖先的xtime属性。因此，更改会从节点（发生更改的位置）一直传播到根节点。



考虑上面的目录树结构。在时间T1，主设备和从设备彼此同步。



在时间T2，创建了新文件File2。这将触发从File2到root的xtime标记（其中xtime是当前时间戳），即File2，Dir3，Dir1和最后Dir0的xtime都将被更新。

地域复制守护程序根据xtime（master）> xtime（slave）的条件对文件系统进行爬取。因此，在我们的示例中，它将仅爬取目录结构的左侧部分，因为目录结构的右侧部分仍具有相同的时间戳。虽然爬取算法很快，但我们仍然需要抓取目录结构的很大一部分。

2）**复制** - 我们使用rsync进行数据复制。Rsync是一个外部实用程序，它将计算两个文件的差异，并将此差异从源发送到同步。

## 3.5、GlusterFS的整体工作

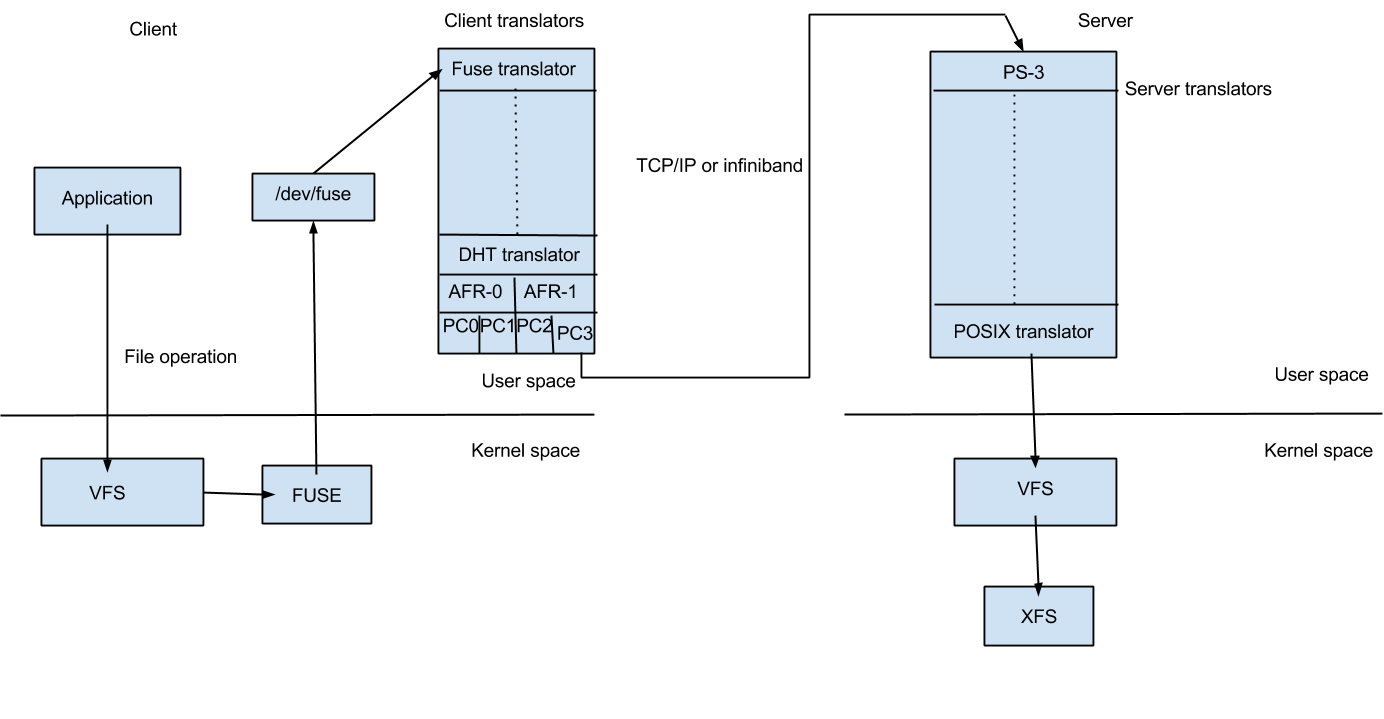
只要GlusterFS安装在服务器节点中，就会创建一个gluster管理守护程序（glusterd）二进制文件。此守护程序应该在群集中的所有参与节点中运行。启动glusterd后，可以创建一个包含所有存储服务器节点的可信服务器池（TSP）（TSP甚至可以包含单个节点）。现在可以将作为基本存储单元的块创建为这些服务器中的导出目录。来自这个TSP的任何数量的块都可以组合在一起形成一个卷。

创建卷后，glusterfsd进程将在每个参与的块中开始运行。除此之外，还将在/ var / lib / glusterd / vols /中生成称为vol文件的配置文件。卷中的每个块都有对应的配置文件。这将包含有关该特定块的所有详细信息。还将包含创建客户端进程所需的配置文件。现在我们的文件系统已经可以使用了。我们可以非常轻松地在客户端计算机上安装此卷，如下所示，并像使用本地存储一样使用它：

mount.glusterfs `<IP or hostname>`:`<volume\_name>` `<mount\_point>

IP或主机名可以是可信服务器池中创建所需卷的任何节点的IP或主机名。

当我们在客户端中安装卷时，客户端glusterfs进程与服务器的glusterd进程通信。服务器glusterd进程发送包含客户端转换器列表的配置文件（vol文件），另一个包含卷中每个块的信息，客户端glusterfs进程现在可以直接与每个块的glusterfsd进程通信。设置现已完成，现在该卷已准备好供客户端服务。



当客户端在挂载的文件系统中发出系统调用（文件操作或Fop）时，VFS（标识文件系统的类型为glusterfs）将把请求发送到FUSE内核模块。FUSE内核模块将通过/ dev / fuse将其发送到客户机节点的用户空间中的GlusterFS（这已在FUSE部分中描述）。客户端上的GlusterFS进程由一组称为client translator的translator组成，这些translator在存储服务器glusterd进程发送的配置文件（vol文件）中定义。这些translators中的第一个是FUSE translator，它由FUSE库（libfuse）组成。每个translator都具有与glusterfs支持的每个文件操作或fop相对应的功能。该请求将在每个translator中达到相应的功能。

1. FUSE translator
2. DHT translator – DHT translator将请求映射到包含所需文件或目录的正确块。
3. AFR translator - 它接收来自前一个translator的请求，如果卷类型是复制的，它会复制请求并将其传递给副本的protocol client translator。
4. Protocol Client translator - Protocol Client translator是client translator堆栈中的最后一个。该translator分为多个线程，每个线程对应一个卷。这将直接与每块的glusterfsd通信。

在包含需要的块的存储服务器节点中，请求再次通过一系列称为服务器转换器的转换器，主要是：

1）协议服务器转换器

2）POSIX翻译

请求最终将到达VFS，然后将与底层本机文件系统进行通信。响应将回溯相同的路径。

# 四、进阶学习

## 4.1 管理集群

### 4.1.1 管理Gluster服务

安装GlusterFS后，必须启动glusterd服务。glusterd服务充当Gluster弹性卷管理

器，监督glusterfs进程，协调动态卷操作，例如无中断地跨多个存储服务器添加和删除卷。

手动启动和停止glusterd

要手动启动glusterd，请输入以下命令：

# /etc/init.d/glusterd start

要手动停止glusterd，请输入以下命令：

# /etc/init.d/glusterd stop

自动启动glusterd

Red Hat和Fedora发行版

# chkconfig glusterd on

Debian和Ubuntu等衍生品

# update-rc.d glusterd defaults

Red Hat和Debian以外的系统

# echo "glusterd" >> /etc/rc.local

### 4.1.2 管理受信任的存储池

可信存储池（TSP）是存储服务器的可信网络。在配置GlusterFS卷之前，必须创建存储服务器的可信存储池，该存储服务器将通过对等方探测服务器为卷提供块。TSP中的服务器是彼此的对等体。

在服务器上安装Gluster之后，在创建可信存储池之前，每个服务器都属于仅包含该服务器的存储池。

用于创建存储池的服务器必须可通过主机名解析。

glusterd守护程序必须在要添加到存储池的所有存储服务器上运行。

必须将服务器上的防火墙配置为允许访问端口24007。

以下命令在由3个服务器组成的TSP上运行 - server1，server2和server3。

#### 4.1.2.1 添加服务器

要将服务器添加到TSP，请从池中已有的服务器进行对等探测。

# gluster peer probe <server>

例如，要将新服务器4添加到上述群集，请从其他服务器之一进行探测：

server1# gluster peer probe server4

Probe successful

验证第一台服务器（server1）的对等体状态：

server1# gluster peer status

Number of Peers: 3

Hostname: server2

Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5

State: Peer in Cluster (Connected)

Hostname: server3

Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7

State: Peer in Cluster (Connected)

Hostname: server4

Uuid: 3e0cabaa-9df7-4f66-8e5d-cbc348f29ff7

State: Peer in Cluster (Connected)

#### 4.1.2.2 列表服务器

列出TSP中的所有节点：

server1# gluster pool list

UUID Hostname State

d18d36c5-533a-4541-ac92-c471241d5418 localhost Connected

5e987bda-16dd-43c2-835b-08b7d55e94e5 server2 Connected

1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7 server3 Connected

3e0cabaa-9df7-4f66-8e5d-cbc348f29ff7 server4 Connected

#### 4.1.2.3 查看对等方状态

要查看TSP中对等方的状态：

server1# gluster peer status

Number of Peers: 3

Hostname: server2

Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5

State: Peer in Cluster (Connected)

Hostname: server3

Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7

State: Peer in Cluster (Connected)

Hostname: server4

Uuid: 3e0cabaa-9df7-4f66-8e5d-cbc348f29ff7

State: Peer in Cluster (Connected)

#### 4.1.2.4 删除服务器

要从TSP中删除服务器，请从池中的另一台服务器运行以下命令：

# gluster peer detach <server>

例如，要从受信任存储池中删除server4：

server1# gluster peer detach server4

Detach successful

验证对等体状态：

server1# gluster peer status

Number of Peers: 2

Hostname: server2

Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5

State: Peer in Cluster (Connected)

Hostname: server3

Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7

State: Peer in Cluster (Connected)

## 4.2设置存储

卷是块的逻辑集合，其中每个块是可信存储池中服务器上的导出目录。在创建卷之前，您需要设置将形成卷的块。

### [块命名约定](https://docs.gluster.org/en/latest/Administrator%20Guide/Brick%20Naming%20Conventions/)

FHS-2.3并不完全清楚服务器共享的数据应驻留在何处。它确实声明“ / srv包含由此系统提供的特定于站点的数据 ”，但GlusterFS数据是否特定于站点？

共识似乎倾向于使用/data。放置块的良好分层方法是：

/data/glusterfs/<volume>/<brick>/brick

在此示例中，<brick>是已挂载的文件系统。

示例：每个服务器一个块

物理磁盘/ dev / sdb将用作您要创建名为myvol1的卷的块存储。您已经在4台服务器的每一台上使用XFS对/ dev / sdb1进行了分区和格式化。

在所有4台服务器上：

mkdir -p /data/glusterfs/myvol1/brick1

mount /dev/sdb1 /data/glusterfs/myvol1/brick1

我们将brick在该文件系统的目录中定义实际的块。如果未安装XFS文件系统，则会导致块无法启动。

仅在一台服务器上：

gluster volume create myvol1 replica 2 server{1..4}:/data/glusterfs/myvol1/brick1/brick

这将创建卷myvol1，它使用/data/glusterfs/myvol1/brick1/brick所有4台服务器上的目录。

示例：每个服务器两个砖

两个物理磁盘/ dev / sdb和/ dev / sdc将用作您要创建的名为myvol2的卷的块存储。您已经在4台服务器上分别格式化了/ dev / sdb1和/ dev / sdc1以及XFS。

在所有4台服务器上：

mkdir -p /data/glusterfs/myvol2/brick{1,2}

mount /dev/sdb1 /data/glusterfs/myvol2/brick1

mount /dev/sdc1 /data/glusterfs/myvol2/brick2

我们将再次brick在这些文件系统的目录中定义实际的块。

仅在一台服务器上：

gluster volume create myvol2 replica 2 \

server{1..4}:/data/glusterfs/myvol2/brick1/brick \

server{1..4}:/data/glusterfs/myvol2/brick2/brick

注意：gluster volume create myvol2 replica 2 server{1..4}:/data/glusterfs/myvol2/brick{1,2}/brick， Bash会扩展最后{}一个，所以最终会在每个服务器上的两块之间复制，而不是跨服务器复制。

### 4.2.2格式化和安装块

要创建精简配置的逻辑卷，请继续执行以下步骤：

使用pvcreate命令创建物理卷（PV）。例如：

# pvcreate --dataalignment 1280K /dev/sdb

这里，/ dev / sdb是一个存储设备。根据您的设备使用正确的数据对齐选项。

注意：设备名称和对齐值将根据您使用的设备而有所不同。

使用vgcreate命令从PV创建卷组（VG）：例如：

# vgcreate --physicalextentsize 128K gfs\_vg /dev/sdb

建议只能从一个存储设备创建一个VG。

使用以下命令创建精简池：

使用以下命令创建LV以充当元数据设备：

# lvcreate -L metadev\_sz --name metadata\_device\_name VOLGROUP

例如：

# lvcreate -L 16776960K --name gfs\_pool\_meta gfs\_vg

使用以下命令创建LV作为数据设备：

# lvcreate -L datadev\_sz --name thin\_pool VOLGROUP

例如：

# lvcreate -L 536870400K --name gfs\_pool gfs\_vg

使用以下命令从数据LV和元数据LV创建精简池：

# lvconvert --chunksize STRIPE\_WIDTH --thinpool VOLGROUP/thin\_pool --poolmetadata VOLGROUP/metadata\_device\_name

例如：

# lvconvert --chunksize 1280K --thinpool gfs\_vg/gfs\_pool --poolmetadata gfs\_vg/gfs\_pool\_meta

注意：默认情况下，精简池中新配置的块被清零，以防止不同块设备之间的数据泄漏。

# lvchange --zero n VOLGROUP/thin\_pool

例如：

# lvchange --zero n gfs\_vg/gfs\_pool

使用lvcreate命令从先前创建的池创建精简配置卷：

例如：

# lvcreate -V 1G -T gfs\_vg/gfs\_pool -n gfs\_lv

建议在精简池中只创建一个LV。

使用支持的XFS配置格式化brick，安装块，并验证brick是否正确安装。

运行# mkfs.xfs -f -i size=512 -n size=8192 -d su=128k,sw=10 DEVICE以将块格式化为支持的XFS文件系统格式。这里，DEVICE是瘦LV。inode大小设置为512字节，以适应GlusterFS使用的扩展属性。

运行# mkdir /mountpoint以创建将块链接到的目录。

在/ etc / fstab中添加一个条目：

/dev/gfs\_vg/gfs\_lv /mountpoint xfs rw,inode64,noatime,nouuid 1 2

运行# mount /mountpoint以安装砖。

运行该df -h命令以验证是否已成功装入块：

# df -h

/dev/gfs\_vg/gfs\_lv 16G 1.2G 15G 7% /exp1

### [POSIX访问控制列表](https://docs.gluster.org/en/latest/Administrator%20Guide/Access%20Control%20Lists/)

POSIX访问控制列表（ACL）允许您为不同的用户或组分配不同的权限，即使它们与原始所有者或拥有组不对应。

例如：用户john创建一个文件，但不希望任何人对此文件执行任何操作，但另一个用户antony除外（即使有其他用户属于该组john）。

这意味着，除文件所有者，文件组和其他人外，还可以使用POSIX ACL授予或拒绝其他用户和组访问权限。

1）激活POSIX ACL支持

要将POSIX ACL用于文件或目录，必须使用POSIX ACL支持挂载文件或目录的分区。

2）在服务器上激活POSIX ACL支持

要为POSIX ACL支持安装后端导出目录，请使用以下命令：

# mount -o acl

例如： # mount -o acl /dev/sda1 /export1

或者，如果分区在/ etc / fstab文件中列出，请为分区添加以下条目以包括POSIX ACLs选项：

LABEL=/work /export1 ext3 rw, acl 14

3）在客户端上激活POSIX ACL支持

要为POSIX ACL支持安装glusterfs卷，请使用以下命令：

# mount –t glusterfs -o acl

例如：

# mount -t glusterfs -o acl 198.192.198.234:glustervolume /mnt/gluster

4）设置POSIX ACL

您可以设置两种类型的POSIX ACL，即访问ACL和默认ACL。您可以使用访问ACL为特定文件或目录授予权限。您只能在目录上使用默认ACL，但如果该目录中的文件没有ACL，则它将继承该目录的默认ACL的权限。

您可以为每个用户，每个组，不在文件的用户组中的用户以及有效的右掩码设置ACL。

5）设置Access ACL

您可以应用Access ACL以授予文件和目录的权限。

设置或修改访问ACL

您可以使用以下命令设置或修改访问ACL：

# setfacl –m file

ACL条目类型是所有者，组和其他的POSIX ACL表示。

权限必须是字符r（读取），w （写入）和x（执行）的组合。您必须以下列格式指定ACL条目，并可以指定以逗号分隔的多个条目类型。

ACL输入 描述：

u：UID：\ < permission> 设置用户的访问ACL。您可以指定用户名或UID

g：GID：\ < permission> 设置组的访问ACL。您可以指定组名称或GID。

m：\ < permission> 设置有效权限掩码。掩码是拥有组的所有访问权限与所有用户和组条目的组合。

o：\ < permission> 为文件组中的用户以外的用户设置访问ACL。

如果文件或目录已具有POSIX ACL，并且使用了setfacl命令，则会将其他权限添加到现有POSIX ACL，或者修改现有规则。

例如，要为用户antony提供读写权限：

# setfacl -m u:antony:rw /mnt/gluster/data/testfile

6）设置默认ACL

您只能将默认ACL应用于目录。它们确定在创建时从其父目录继承的文件系统对象的权限。

您可以使用以下命令为文件和目录设置默认ACL：

# setfacl –m –-set

权限必须是字符r（读取），w（写入）和x（执行）的组合。如下所述指定ACL entry\_type，用逗号分隔多个条目类型。

u：user\_name：permissions 设置用户的访问ACL。指定用户名或UID。

g：group\_name：permissions 设置组的访问ACL。指定组名称或GID。

m：permission 设置有效权限掩码。掩码是拥有组的所有访问权限以及所有用户和组条目的组合。

o：permissions 为文件组中的用户以外的用户设置访问ACL。

例如，要为不在用户组中的用户设置/ data目录的默认ACL：

# setfacl –m --set o::r /mnt/gluster/data

注意：为单个文件设置的访问ACL可以覆盖默认ACL权限。

默认ACL的影响

以下是将目录的默认ACL的权限传递给其中的文件和子目录的方式：

a）子目录继承父目录的默认ACL作为其默认ACL和访问ACL。文件继承默认ACL作为其访问ACL。

b）检索POSIX ACL

您可以查看文件或目录的现有POSIX ACL。

7）查看现有POSIX ACL

使用以下命令查看文件的现有访问ACL：

# getfacl

例如，要查看sample.jpg的现有POSIX ACL

# getfacl /mnt/gluster/data/test/sample.jpg

# owner: antony

# group: antony

user::rw-

group::rw-

other::r--

使用以下命令查看目录的默认ACL：

# getfacl

例如，要查看/ data / doc的现有ACL

# getfacl /mnt/gluster/data/doc

# owner: antony

# group: antony

user::rw-

user:john:r--

group::r--

mask::r--

other::r--

default:user::rwx

default:user:antony:rwx

default:group::r-x

default:mask::rwx

default:other::r-x

8）删除POSIX ACL

要删除用户，组或其他人的所有权限，请使用以下命令：

# setfacl -x

setfaclentry\_type选项

ACL entry\_type转换为所有者，组和其他的POSIX ACL表示。

权限必须是字符r（读取），w（写入）和x（执行）的组合。如下所述指定ACL entry\_type，用逗号分隔多个条目类型。

u：user\_name 设置用户的访问ACL。指定用户名或UID。

g：group\_name 设置组的访问ACL。指定组名称或GID。

m：permission 设置有效权限掩码。掩码是拥有组的所有访问权限以及所有用户和组条目的组合。

o：permissions 为文件组中的用户以外的用户设置访问ACL。

例如，要从用户antony中删除所有权限：

# setfacl -x u:antony /mnt/gluster/data/test-file

9）Samba和ACL

如果您使用Samba访问GlusterFS FUSE安装，则默认情况下会启用POSIX ACL。Samba已使用该--with-acl-support选项进行编译 ，因此在访问或装载Samba共享时不需要特殊标志。

10）NFS和ACL

目前GlusterFS通过NFS挂载支持POSIX ACL配置，即setfacl和getfacl命令通过NFS挂载工作。

## 4.3[设置客户端](https://docs.gluster.org/en/latest/Administrator%20Guide/Setting%20Up%20Clients/)

* 1. [处理属于多个组的用户](https://docs.gluster.org/en/latest/Administrator%20Guide/Handling-of-users-with-many-groups/)

## 4.4卷

### 4.4.1 调整卷选项

您可以根据需要在群集联机且可用时调整卷选项。

注意：如果每个卷中有太多块，或者有太多服务已经使用了系统中的所有特权端口，建议您将server.allow-insecure选项设置为ON。打开此选项允许端口接受/拒绝来自不安全端口的消息。因此，仅在部署需要时才使用此选项。

使用以下命令调整卷选项：

# gluster volume set <VOLNAME>

例如，要为test-volume指定性能缓存大小：

# gluster volume set test-volume performance.cache-size 256MB

Set volume successful

### 4.4.2 配置卷的传输类型

卷可以支持一种或多种传输类型，以便在客户端和块进程之间进行通信。支持的传输有三种类型，分别是tcp，rdma和tcp，rdma。

要更改卷的支持传输类型，请按照以下过程操作：

1）使用以下命令卸载所有客户端上的卷：

# umount mount-point

2）使用以下命令停止卷：

# gluster volume stop <VOLNAME>

3）更改运输类型。例如，要启用tcp和rdma，请执行followimg命令：

# gluster volume set test-volume config.transport tcp,rdma OR tcp OR rdma

4）在所有客户端上安装卷。例如，要使用rdma transport进行安装，请使用以下命令：

# mount -t glusterfs -o transport=rdma server1:/test-volume /mnt/glusterfs

### 4.4.3 扩展卷

您可以根据需要在群集联机且可用时扩展卷。例如，您可能希望向分布式卷添加块，从而增加分布并增加GlusterFS卷的容量。

同样，您可能希望将一组块添加到分布式复制卷，从而增加GlusterFS卷的容量。

注意：扩展分布式复制和分布式分散卷时，需要添加多个块，这些块是副本或分散计数的倍数。例如，要展开副本计数为2的分布式复制卷，您需要以2的倍数（例如4,6,8等）添加brick。

扩展一个卷

1、如果它们还不是TSP的一部分，请使用以下命令探测包含要添加到卷的块的服务器：

# gluster peer probe <SERVERNAME>

例如：

# gluster peer probe server4

Probe successful

2、使用以下命令添加块：

# gluster volume add-brick <VOLNAME> <NEW-BRICK>

例如：

# gluster volume add-brick test-volume server4:/exp4

Add Brick successful

3、使用以下命令检查卷信息：

# gluster volume info <VOLNAME>

该命令显示类似于以下内容的信息：

Volume Name: test-volume

Type: Distribute

Status: Started

Number of Bricks: 4

Bricks:

Brick1: server1:/exp1

Brick2: server2:/exp2

Brick3: server3:/exp3

Brick4: server4:/exp4

4、重新平衡卷以确保将文件分发到新块。

您可以使用重新平衡卷中所述的rebalance命令

收缩量

您可以根据需要在群集联机且可用时缩小卷。例如，您可能需要删除由于硬件或网络故障而在分布式卷中无法访问的块。

注意

驻留在您要移除的brick上的数据将无法再在Gluster安装点访问。但请注意，仅删除配置信息 - 您可以根据需要继续直接从块中访问数据。

缩小分布式复制和分布式分散卷时，需要删除多个副本或条带计数的块。例如，要缩小副本计数为2的分布式复制卷，您需要删除2的倍数的砖（例如4,6,8等）。此外，您要删除的brick必须来自相同的子卷（相同的副本或分散集）。

使用start选项运行remove-brick 将自动触发重新平衡操作，以将数据从已移除的块迁移到卷的其余部分。

### 4.4.4 缩小卷

您可以根据需要在群集联机且可用时缩小卷。例如，您可能需要删除由于硬件或网络故障而在分布式卷中无法访问的块。

注意：驻留在您要移除的brick上的数据将无法再在Gluster安装点访问。但请注意，仅删除配置信息 - 您可以根据需要继续直接从块中访问数据。

缩小分布式复制和分布式分散卷时，需要删除多个副本或条带计数的块。例如，要缩小副本计数为2的分布式复制卷，您需要删除2的倍数的砖（例如4,6,8等）。此外，您要删除的brick必须来自相同的子卷（相同的副本或分散集）。

使用start选项运行remove-brick 将自动触发重新平衡操作，以将数据从已移除的块迁移到卷的其余部分。

1、使用以下命令删除块：

# gluster volume remove-brick <VOLNAME> <BRICKNAME> start

例如，要删除server2：/ exp2：

# gluster volume remove-brick test-volume server2:/exp2 start

volume remove-brick start: success

2、使用以下命令查看remove brick操作的状态：

# gluster volume remove-brick <VOLNAME> <BRICKNAME> status

例如，要查看server2上的remove brick操作的状态：/ exp2 brick：

# gluster volume remove-brick test-volume server2:/exp2 status

Node Rebalanced-files size scanned status

--------- ---------------- ---- ------- -----------

617c923e-6450-4065-8e33-865e28d9428f 34 340 162 in progress

3、状态显示“已完成”后，提交remove-brick操作

# gluster volume remove-brick <VOLNAME> <BRICKNAME> commit

在这个例子中：

# gluster volume remove-brick test-volume server2:/exp2 commit

Removing brick(s) can result in data loss. Do you want to Continue? (y/n) y

volume remove-brick commit: success

Check the removed bricks to ensure all files are migrated.

If files with data are found on the brick path, copy them via a gluster mount point before re-purposing the removed brick

4、使用以下命令检查卷信息：

# gluster volume info

该命令显示类似于以下内容的信息：

# gluster volume info

Volume Name: test-volume

Type: Distribute

Status: Started

Number of Bricks: 3

Bricks:

Brick1: server1:/exp1

Brick3: server3:/exp3

Brick4: server4:/exp4

### 4.4.5 更换有缺陷的brick

在纯分布卷中替换块

要替换仅分发卷上的块，请添加新块，然后删除要替换的块。这将触发重新平衡操作，该操作将从移除的brick移动数据。

注意：仅对分布式复制卷或纯复制卷支持在gluster中使用'replace-brick'命令替换块。

删除brick Server1：/ home / gfs / r2\_1并添加Server1：/ home / gfs / r2\_2的步骤：

1、这是初始卷配置：

Volume Name: r2

Type: Distribute

Volume ID: 25b4e313-7b36-445d-b524-c3daebb91188

Status: Started

Number of Bricks: 2

Transport-type: tcp

Bricks:

Brick1: Server1:/home/gfs/r2\_0

Brick2: Server1:/home/gfs/r2\_1

2、以下是mount上存在的文件：

# ls

1. 10 2 3 4 5 6 7 8 9

3、现在添加新砖 - Server1：/ home / gfs / r2\_2：

# gluster volume add-brick r2 Server1:/home/gfs/r2\_2

volume add-brick: success

4、使用以下命令启动remove-brick：

# gluster volume remove-brick r2 Server1:/home/gfs/r2\_1 start

volume remove-brick start: success

ID: fba0a488-21a4-42b7-8a41-b27ebaa8e5f4

5、等到remove-brick状态表明它已完成。

# gluster volume remove-brick r2 Server1:/home/gfs/r2\_1 status

Node Rebalanced-files size scanned failures skipped status run time in secs

--------- ----------- ----------- ----------- ----------- ----------- ------------ --------------

localhost 5 20Bytes 15 0 0 completed 0.00

6、现在我们可以安全地删除旧砖，因此提交更改：

# gluster volume remove-brick r2 Server1:/home/gfs/r2\_1 commit

Removing brick(s) can result in data loss. Do you want to Continue? (y/n) y

volume remove-brick commit: success

7、这是新的卷配置。

Volume Name: r2

Type: Distribute

Volume ID: 25b4e313-7b36-445d-b524-c3daebb91188

Status: Started

Number of Bricks: 2

Transport-type: tcp

Bricks:

Brick1: Server1:/home/gfs/r2\_0

Brick2: Server1:/home/gfs/r2\_2

8、检查mount的内容：

# ls

1. 10 2 3 4 5 6 7 8 9

替换复制/分布式复制卷中的块

本文档的这一部分描述了brick：如何Server1:/home/gfs/r2\_0用砖代替：具有副本计数的Server1:/home/gfs/r2\_5卷。r22

Volume Name: r2

Type: Distributed-Replicate

Volume ID: 24a0437a-daa0-4044-8acf-7aa82efd76fd

Status: Started

Number of Bricks: 2 x 2 = 4

Transport-type: tcp

Bricks:

Brick1: Server1:/home/gfs/r2\_0

Brick2: Server2:/home/gfs/r2\_1

Brick3: Server1:/home/gfs/r2\_2

Brick4: Server2:/home/gfs/r2\_3

步骤：

1、确保新砖Server1中没有数据：/ home / gfs / r2\_5

2、检查所有brick是否正在运行。如果要更换的brick已经关闭，那也没关系。

3、如果还没有，那么将要更换的brick。

a）通过执行'gluster volume'获取brick的pid 状态'

# gluster volume status

Status of volume: r2

Gluster process Port Online Pid

------------------------------------------------------------------------------

Brick Server1:/home/gfs/r2\_0 49152 Y 5342

Brick Server2:/home/gfs/r2\_1 49153 Y 5354

Brick Server1:/home/gfs/r2\_2 49154 Y 5365

Brick Server2:/home/gfs/r2\_3 49155 Y 5376

b）登录到运行brick的机器并杀死brick。

# kill -15 5342

c）确认brick不再运行，其他brick运行正常。

# gluster volume status

Status of volume: r2

Gluster process Port Online Pid

------------------------------------------------------------------------------

Brick Server1:/home/gfs/r2\_0 N/A N 5342 <<---- brick is not running, others are running fine.

Brick Server2:/home/gfs/r2\_1 49153 Y 5354

Brick Server1:/home/gfs/r2\_2 49154 Y 5365

Brick Server2:/home/gfs/r2\_3 49155 Y 5376

4、使用该Gluster volume fuse mount（在这个例子中：/mnt/r2）设置元数据，以便数据将被同步到新块（在这种情况下，它是从Server1:/home/gfs/r2\_1至Server1:/home/gfs/r2\_5）

在挂载点上创建一个尚不存在的目录。然后删除该目录，通过执行setfattr对元数据更改日志执行相同操作。此操作标志着未决的changelog它会告诉自愈达蒙/坐骑执行从自我修复/home/gfs/r2\_1来/home/gfs/r2\_5。

mkdir /mnt/r2/<name-of-nonexistent-dir>

rmdir /mnt/r2/<name-of-nonexistent-dir>

setfattr -n trusted.non-existent-key -v abc /mnt/r2

setfattr -x trusted.non-existent-key /mnt/r2

检查正在替换的砖的副本上是否有待处理的xattrs：

getfattr -d -m. -e hex /home/gfs/r2\_1

# file: home/gfs/r2\_1

security.selinux=0x756e636f6e66696e65645f753a6f626a6563745f723a66696c655f743a733000

trusted.afr.r2-client-0=0x000000000000000300000002 <<---- xattrs are marked from source brick Server2:/home/gfs/r2\_1

trusted.afr.r2-client-1=0x000000000000000000000000

trusted.gfid=0x00000000000000000000000000000001

trusted.glusterfs.dht=0x0000000100000000000000007ffffffe

trusted.glusterfs.volume-id=0xde822e25ebd049ea83bfaa3c4be2b440

5、卷修复信息将显示'/'。（可能有更多的条目基于工作负荷。但'/'必须存在）

# gluster volume heal r2 info

Brick Server1:/home/gfs/r2\_0

Status: Transport endpoint is not connected

Brick Server2:/home/gfs/r2\_1

/

Number of entries: 1

Brick Server1:/home/gfs/r2\_2

Number of entries: 0

Brick Server2:/home/gfs/r2\_3

Number of entries: 0

6、用'commit force'选项替换砖。请注意，不支持replace-brick命令的其他变体。

执行replace-brick命令

# gluster volume replace-brick r2 Server1:/home/gfs/r2\_0 Server1:/home/gfs/r2\_5 commit force

volume replace-brick: success: replace-brick commit successful

检查新块现在是否在线

# gluster volume status

Status of volume: r2

Gluster process Port Online Pid

------------------------------------------------------------------------------

Brick Server1:/home/gfs/r2\_5 49156 Y 5731 <<<---- new brick is online

Brick Server2:/home/gfs/r2\_1 49153 Y 5354

Brick Server1:/home/gfs/r2\_2 49154 Y 5365

Brick Server2:/home/gfs/r2\_3 49155 Y 5376

用户可以使用以下方式跟踪修复的进度：gluster volume heal [volname] info。一旦自我修复完成，将删除更改日志。

# getfattr -d -m. -e hex /home/gfs/r2\_1

getfattr: Removing leading '/' from absolute path names

# file: home/gfs/r2\_1

security.selinux=0x756e636f6e66696e65645f753a6f626a6563745f723a66696c655f743a733000

trusted.afr.r2-client-0=0x000000000000000000000000 <<---- Pending changelogs are cleared.

trusted.afr.r2-client-1=0x000000000000000000000000

trusted.gfid=0x00000000000000000000000000000001

trusted.glusterfs.dht=0x0000000100000000000000007ffffffe

trusted.glusterfs.volume-id=0xde822e25ebd049ea83bfaa3c4be2b440

# gluster volume heal <VOLNAME> info 将表明不需要修复。

# gluster volume heal r2 info

Brick Server1:/home/gfs/r2\_5

Number of entries: 0

Brick Server2:/home/gfs/r2\_1

Number of entries: 0

Brick Server1:/home/gfs/r2\_2

Number of entries: 0

Brick Server2:/home/gfs/r2\_3

Number of entries: 0

### 4.4.6 重新平衡卷

使用add-brick命令扩展卷后，您可能需要在服务器之间重新平衡数据。扩展或缩小卷后创建的新目录将自动均匀分布。对于所有现有目录，可以通过重新平衡布局和/或数据来修复分发。

本节介绍如何使用以下常见方案在存储环境中重新平衡GlusterFS卷：

修复布局 - 修复布局以使用新的卷拓扑，以便可以将文件分发到新添加的节点。

修复布局和迁移数据 - 通过修复布局以使用新卷拓扑并迁移现有数据来重新平衡卷。

**重新平衡卷以修复布局更改**

修复布局是必要的，因为布局结构对于给定目录是静态的。即使在将新砖添加到卷中之后，现有目录中新创建的文件仍将仅在原始块中分发。该命令gluster volume rebalance <volname> fix-layout start将修复布局信息，以便可以在新添加的brick上创建文件。发出此命令后，将重新验证已缓存的所有文件统计信息。

从GlusterFS 3.6开始，将文件分配给brick将考虑brick的大小。例如，20TB砖的分配数量是10TB砖的两倍。在3.6之前的版本中，无论大小如何，两块砖被视为相等，并且将被分配相等份额的文件。

修复布局重新平衡仅修复布局更改，不会迁移数据。如果要迁移现有数据，请使用gluster volume rebalance start命令在服务器之间重新平衡数据。

重新平衡卷以修复布局

使用以下命令在任何Gluster服务器上启动重新平衡操作：

# gluster volume rebalance <VOLNAME> fix-layout start

例如：

# gluster volume rebalance test-volume fix-layout start

Starting rebalance on volume test-volume has been successful

**重新平衡卷以修复布局和迁移数据**

分别使用add-brick扩展卷后，需要在服务器之间重新平衡数据。remove-brick命令将自动触发重新平衡。

重新平衡卷以修复布局并迁移现有数据

1、使用以下命令在任一服务器上启动重新平衡操作：

# gluster volume rebalance <VOLNAME> start

例如：

# gluster volume rebalance test-volume start

Starting rebalancing on volume test-volume has been successful

2、使用以下命令在任何一台服务器上强制启动迁移操作：

# gluster volume rebalance <VOLNAME> start force

例如：

# gluster volume rebalance test-volume start force

Starting rebalancing on volume test-volume has been successful

**显示再平衡操作的状态**

您可以根据需要显示有关重新平衡卷操作的状态信息。

1、使用以下命令检查重新平衡操作的状态：

# gluster volume rebalance <VOLNAME> status

例如：

# gluster volume rebalance test-volume status

Node Rebalanced-files size scanned status

--------- ---------------- ---- ------- -----------

617c923e-6450-4065-8e33-865e28d9428f 416 1463 312 in progress

完成重新平衡操作的时间取决于卷上的文件数以及相应的文件大小。继续检查重新平衡状态，验证重新平衡的文件数或扫描的总文件数不断增加。

例如，再次运行status命令可能会显示类似于以下内容的结果：

# gluster volume rebalance test-volume status

Node Rebalanced-files size scanned status

--------- ---------------- ---- ------- -----------

617c923e-6450-4065-8e33-865e28d9428f 498 1783 378 in progress

重新平衡完成后，重新平衡状态显示以下内容：

# gluster volume rebalance test-volume status

Node Rebalanced-files size scanned status

--------- ---------------- ---- ------- -----------

617c923e-6450-4065-8e33-865e28d9428f 502 1873 334 completed

**停止正在进行的再平衡操作**

如果需要，您可以停止重新平衡操作。

1、使用以下命令停止重新平衡操作：

# gluster volume rebalance <VOLNAME> stop

例如：

# gluster volume rebalance test-volume stop

Node Rebalanced-files size scanned status

--------- ---------------- ---- ------- -----------

617c923e-6450-4065-8e33-865e28d9428f 59 590 244 stopped

Stopped rebalance process on volume test-volume

### 4.4.7 停止卷

使用以下命令停止卷：

# gluster volume stop <VOLNAME>

例如，要停止测试卷：

# gluster volume stop test-volume

Stopping volume will make its data inaccessible. Do you want to continue? (y/n)

输入y以确认操作。该命令的输出显示以下内容：

Stopping volume test-volume has been successful

### 4.4.8 删除卷

使用以下命令删除卷：

# gluster volume delete <VOLNAME>

例如，要删除测试卷：

# gluster volume delete test-volume

Deleting volume will erase all information about the volume. Do you want to continue? (y/n)

输入y以确认操作。该命令显示以下内容：

Deleting volume test-volume has been successful

### 4.4.9 在复制时触发自我修复

在复制模块中，以前您必须在brick脱机并重新联机时手动触发自我修复，以使所有副本同步。现在，主动式自我修复守护程序在后台运行，诊断问题并自动启动每10分钟自我修复需要修复的文件。

您可以查看需要的文件列表愈合，这是目前/先前的文件列表中医治这是在脑裂状态，文件的列表，你可以手动触发整个卷上或仅在文件自我修复哪个需要治愈。

1、仅在需要修复的文件上触发自我修复：

# gluster volume heal <VOLNAME>

例如，要触发需要修复 测试卷的文件的自我修复：

# gluster volume heal test-volume

Heal operation on volume test-volume has been successful

2、触发卷上所有文件的自我修复：

# gluster volume heal <VOLNAME> full

例如，要在test-volume的所有文件上触发自我修复：

# gluster volume heal test-volume full

Heal operation on volume test-volume has been successful

3、查看需要修复的文件列表：

# gluster volume heal <VOLNAME> info

例如，要查看需要修复的测试卷上的文件列表 ：

# gluster volume heal test-volume info

Brick server1:/gfs/test-volume\_0

Number of entries: 0

Brick server2:/gfs/test-volume\_1

Number of entries: 101

/95.txt

/32.txt

/66.txt

/35.txt

/18.txt

/26.txt

/47.txt

/55.txt

/85.txt

...

4、查看自我修复的文件列表：

# gluster volume heal <VOLNAME> info healed

例如，要查看自我修复的测试卷上的文件列表：

# gluster volume heal test-volume info healed

Brick Server1:/gfs/test-volume\_0

Number of entries: 0

Brick Server2:/gfs/test-volume\_1

Number of entries: 69

/99.txt

/93.txt

/76.txt

/11.txt

/27.txt

/64.txt

/80.txt

/19.txt

/41.txt

/29.txt

/37.txt

/46.txt

...

5、查看自愈失败的特定卷的文件列表：

# gluster volume heal <VOLNAME> info failed

例如，要查看未自我修复的测试卷文件列表：

# gluster volume heal test-volume info failed

Brick Server1:/gfs/test-volume\_0

Number of entries: 0

Brick Server2:/gfs/test-volume\_3

Number of entries: 72

/90.txt

/95.txt

/77.txt

/71.txt

/87.txt

/24.txt

...

6、查看处于裂脑状态的特定卷的文件列表：

# gluster volume heal <VOLNAME> info split-brain

例如，要查看处于裂脑状态的测试卷文件列表：

# gluster volume heal test-volume info split-brain

Brick Server1:/gfs/test-volume\_2

Number of entries: 12

/83.txt

/28.txt

/69.txt

...

Brick Server2:/gfs/test-volume\_3

Number of entries: 12

/83.txt

/28.txt

/69.txt

...

### 4.4.10 非统一文件分配

NUFA转换器或非统一文件访问转换器设计用于在HPC类型的环境中使用时更优先考虑本地驱动器。它可以应用于Distribute和Replica转换器; 在后一种情况下，如果空间允许，它确保一个副本是本地的。

当服务器上的客户端创建文件时，文件将根据文件名分配给卷中的块。这种分配可能并不理想，因为对非本地块或导出目录的读/写操作存在较高的延迟和不必要的网络流量。NUFA确保在服务器的本地导出目录中创建文件，从而减少延迟并节省访问该文件的服务器的带宽。这对于在存储服务器上的挂载点上运行的应用程序也很有用。

如果本地块空间不足或达到最小磁盘空闲限制，如果这些块上有可用空间，NUFA会将文件分发到同一卷中的其他块。

在创建卷中的任何数据之前，应启用NUFA。

使用以下命令启用NUFA：

# gluster volume set <VOLNAME> cluster.nufa enable on

NUFA在以下条件下受到支持：

1）每个服务器卷只有一个brick。

2）对于FUSE客户端。NUFA不支持NFS或SMB。

3）正在挂载已启用NUFA的卷的客户端必须存在于受信任的存储池中。

NUFA调度程序也存在，用于Unify translator; 见下文。

volume bricks

type cluster/nufa

option local-volume-name brick1

subvolumes brick1 brick2 brick3 brick4 brick5 brick6 brick7

end-volume

**NUFA附加选项**

1）lookup-unhashed

这是一个高级选项，如果在与文件名的哈希值匹配的子卷上缺少文件，则会在所有子卷中查找文件。默认为打开。

2）local-volume-name

要考虑本地和首选文件创建的卷名称。默认设置是搜索与系统主机名匹配的卷。

3）subvolumes

此选项列出属于此“cluster / nufa”卷的子卷。这个translator需要不止一个子卷。

**BitRot检测**

通过Gluster中的BitRot检测，可以识别“隐蔽”类型的磁盘错误，其中数据被静默损坏，没有从磁盘到存储软件层的指示，而不是发生错误。这也有助于捕获brick的“后端”修复（其中数据直接在brick上操纵，而无需通过FUSE，NFS或任何其他访问协议）。

BitRot检测默认是禁用的，需要启用才能使用其他子命令。

1、为给定的卷启用bitrot检测 ：

# gluster volume bitrot <VOLNAME> enable

禁用bitrot：

# gluster volume bitrot <VOLNAME> disable

注意：启用bitrot会扫描每个节点的Signer和Scrubber守护程序。Signer负责签署（计算每个文件的校验和）对象，并且清洗由对象数据验证计算的校验和。

2、Scrubber守护程序具有三（3）个限制模式，可调整验证对象的速率。

# volume bitrot <VOLNAME> scrub-throttle lazy

# volume bitrot <VOLNAME> scrub-throttle normal

# volume bitrot <VOLNAME> scrub-throttle aggressive

默认情况下，scrubber每两周清洗一次文件系统。可以根据预定义的频率（例如每月等）将其调整为擦洗。这可以如下所示完成：

# volume bitrot <VOLNAME> scrub-frequency daily

# volume bitrot <VOLNAME> scrub-frequency weekly

# volume bitrot <VOLNAME> scrub-frequency biweekly

# volume bitrot <VOLNAME> scrub-frequency monthly

注意：GA版本不提供每日清洗。

3、Scrubber守护程序可以暂停，并在需要时稍后恢复。这可以如下所示完成：

# volume bitrot <VOLNAME> scrub pause

恢复清洗：

# volume bitrot <VOLNAME> scrub resume`

注意：只要为该特定卷启用了bitrot，就不能暂停（并恢复）签名并始终处于活动状态。

### 4.4.11 使用过滤器修改.vol文件

如果您需要手动更改.vol文件，建议通过客户端界面（'gluster foo'）进行更改。不建议直接更改.vol文件，因为无法预测.vol文件将在磁盘上重置时，例如使用'gluster set foo'命令。命令行界面并不读取.vol文件，而是保持状态并重建它们（来自'/ var / lib / glusterd / vols / \ $ vol / info'）。

有另一种方法可以做到这一点。您可以在目录'/ usr / lib \* / glusterfs / \ $ VERSION / filter'中创建shell脚本。每次将.vol文件写回磁盘时，都会执行所有脚本。传递给位于其中的所有脚本的第一个也是唯一的参数是.vol文件的名称。

所以你可以在那里创建一个如下所示的脚本：

#!/bin/sh`\

sed -i 'some-sed-magic' "$1"

这将运行脚本，然后脚本将在.vol文件上运行sed命令（传递 \ $ 1）。

重要的是，脚本需要设置为可执行文件（例如通过chmod），否则它将不会运行

## 4.5 通过GlusterFS配置NFS-Ganesha

NFS-Ganesha是NFS协议的用户空间文件服务器，支持NFSv3，v4，v4.1和pNFS。它提供了与FUSE兼容的文件系统抽象层（FSAL），允许文件系统开发人员插入自己的存储机制并从任何NFS客户端访问它。NFS-Ganesha可以直接通过其FSAL访问FUSE文件系统，而无需向内核复制任何数据，从而可能缩短响应时间。

### 4.5.1 安装nfs-ganesha

Gluster RPMs（> = 3.10）

Glusterfs-server

glusterfs-API

glusterfs-ganesha

Ganesha RPMs（> = 2.5）

NFS- ganesha

NFS- ganesha -gluster

### 4.5.2 手动启动NFS-Ganesha

1、要手动启动NFS-Ganesha，请使用以下命令：

# server nfs-ganesha start

where:

/var/log/ganesha.log is the default log file for the ganesha process.

/etc/ganesha/ganesha.conf is the default configuration file

NIV\_EVENT is the default log level.

2、如果用户想要在首选模式下运行ganesha，请执行以下命令：

#ganesha.nfsd -f \ <location\_of\_nfs-ganesha.conf\_file> -L \ <location\_of\_log\_file> -N \ <log\_level>

For example:

#ganesha.nfsd -f nfs-ganesha.conf -L nfs-ganesha.log -N NIV\_DEBUG

where:

nfs-ganesha.log is the log file for the ganesha.nfsd process.

nfs-ganesha.conf is the configuration file

NIV\_DEBUG is the log level.

3、默认情况下，服务器的exportlist将为Null

Note : include following parameters in ganesha configuration file for exporting gluster volumes

NFS\_Core\_Param {

#Use supplied name other tha IP In NSM operations

NSM\_Use\_Caller\_Name = true;

#Copy lock states into "/var/lib/nfs/ganesha" dir

Clustered = false;

#Use a non-privileged port for RQuota

Rquota\_Port = 875;

}

### 4.5.3 通过NFS-Ganesha导出GlusterFS卷的分步过程

第一步 ：要在卷中导出任何GlusterFS卷或目录，请为导出配置文件中的每个条目创建EXPORT块。导出任何条目都需要以下参数。- #cat export.conf

EXPORT{

Export\_Id = 1 ; # Export ID unique to each export

Path = "volume\_path"; # Path of the volume to be exported. Eg: "/test\_volume"

FSAL {

name = GLUSTER;

hostname = "10.xx.xx.xx"; # IP of one of the nodes in the trusted pool

volume = "volume\_name"; # Volume name. Eg: "test\_volume"

}

Access\_type = RW; # Access permissions

Squash = No\_root\_squash; # To enable/disable root squashing

Disable\_ACL = TRUE; # To enable/disable ACL

Pseudo = "pseudo\_path"; # NFSv4 pseudo path for this export. Eg: "/test\_volume\_pseudo"

Protocols = "3","4" ; # NFS protocols supported

Transports = "UDP","TCP" ; # Transport protocols supported

SecType = "sys"; # Security flavors supported

}

第2步 ：现在在ganesha配置文件中包含导出配置文件（默认情况下）。这可以通过在文件末尾添加以下行来完成 - ％include“\ <导出配置路径>”

Note :

The above two steps can be done with following script

#/usr/libexec/ganesha/create-export-ganesha.sh <ganesha directory> on <volume name>

By default ganesha directory is "/etc/ganesha"

This will create export configuration file in <ganesha directory>/exports/export.<volume name>.conf

Also it will add above entry to ganesha.conf

第3步：打开该卷的features.cache-invalidation - gluster volume set \ <volume name> features.cache-invalidation on

第4步 ：dbus命令用于export/unexport volume-

export - #dbus-send --system --print-reply --dest = org.ganesha.nfsd / org / ganesha / nfsd / ExportMgr org.ganesha.nfsd.exportmgr.AddExport串：/exports/export.conf string：“EXPORT（path= / \ <volume name>）”

unexprot - #dbus-send --system --dest = org.ganesha.nfsd / org / ganesha / nfsd / ExportMgr org.ganesha.nfsd.exportmgr.RemoveExport uint16：\ <export id>

注意 :

Step 4 can be performed via following script

#/usr/libexec/ganesha/dbus-send.sh <ganesha directory> [on|off] <volume name>

上面的脚本（在第三步和第四步中提到）在glusterfs 3.10 rpms中可用。

第5步：

1）要检查卷是否已导出，请运行

#showmount -e localhost

2）或者使用以下dbus命令

#dbus-send --type = method\_call --print-reply --system --dest = org.ganesha.nfsd / org / ganesha / nfsd / ExportMgr org.ganesha.nfsd.exportmgr.ShowExports

3）查看客户端

#dbus-send --type = method\_call --print-reply --system --dest = org.ganesha.nfsd / org / ganesha / nfsd / ClientMgr org.ganesha.nfsd.clientmgr.ShowClients

### 4.5.4 使用高度可用的Active-Active NFS-Ganesha和GlusterFS cli

请注意目前nfs-ganesha的HA解决方案可在3.10中找到。从3.12开始，HA将由正在开发的称为storhaug的不同项目处理。

在高度可用的Active-Active环境中，如果连接到运行特定应用程序的NFS客户端的NFS-Ganesha服务器崩溃，则应用程序/ NFS客户端将无缝连接到另一个NFS-Ganesha服务器，而无需任何管理干预。使用Pacemaker和Corosync维护群集。Pacemaker充当资源管理器，Corosync提供集群的通信层。使用UPCALL基础架构实现群集中多头NFS-Ganesha服务器的数据一致性。UPCALL基础结构是一个通用且可扩展的框架，可在后端文件系统中检测到任何更改的情况下向各自的glusterfs客户端（在本例中为NFS-Ganesha服务器）发送通知。

高可用性群集在以下三个阶段中配置：

**创建ganesha-ha.conf文件**

安装Gluster Storage时，会在以下位置/ etc / ganesha中创建ganesha-ha.conf.example。将文件重命名为ganesha-ha.conf并按照以下示例中的建议进行更改：sample ganesha-ha.conf file：

＃已创建的HA集群的名称。在子网内必须是唯一的

HA\_NAME = “象头-HA-360”

#Gluster可信池的节点子集，形成ganesha HA集群。

#Hostname已指定。

HA\_CLUSTER\_NODES = “服务器1，服务器，...”

＃HA\_CLUSTER\_NODES = “server1.lab.redhat.com，server2.lab.redhat.com，...”

＃上面指定的每个节点的虚拟IP。

VIP\_server1 = “10.0.2.1”

VIP\_server2 = “10.0.2.2”

**使用gluster CLI配置NFS-Ganesha**

可以使用gluster CLI设置或清除HA群集。此外，它还可以导出和导入特定卷。有关更多信息，请参阅使用gluster CLI配置NFS-Ganesha一节。

**使用ganesha-ha.sh脚本修改HA群集**

发布集群后，可以使用该ganesha-ha.sh脚本进行任何进一步的修改。有关更多信息，请参阅使用ganesha-ha.sh脚本修改HA群集一节。

**使用Gluster CLI配置NFS-Ganesha**

在您的环境中运行NFS-Ganesha之前，请确保考虑以下先决条件：

1、必须可以导出Gluster存储卷，并且所有节点上都安装了NFS-Ganesha rpm。

2、必须在NFS-Ganesha守护程序使用的主机接口上启用IPv6。要启用IPv6支持，请执行以下步骤：

在/etc/modprobe.d/ipv6.conf文件中注释或删除行选项ipv6 disable = 1。

重新启动系统。

3、确保群集中的所有节点都是DNS可解析的。例如，您可以使用群集中所有节点的详细信息填充/ etc / hosts。

4、禁用并停止NetworkManager服务。

5、在所有计算机上启用并启动网络服务。

6、创建并安装gluster共享卷。

gluster volume set all cluster.enable-shared-storage enable

7、在所有机器上安装Pacemaker和Corosync。

8、在所有计算机上设置集群验证密码。

9、需要在所有HA节点上启用无密码ssh。跟着这些步骤，

在群集中的一个（主）节点上，运行：

ssh-keygen -f /var/lib/glusterd/nfs/secret.pem

在所有节点上部署pubkey~root / .ssh / authorized键，运行：

ssh-copy-id -i /var/lib/glusterd/nfs/secret.pem.pub root @ $ node

将密钥复制到集群中的所有节点，运行：

scp /var/lib/glusterd/nfs/secret.\* $ node：/ var / lib / glusterd / nfs /

在共享存储路径中创建一个名为“nfs-ganesha”的目录，并在其中创建ganesha.conf和ganesha-ha.conf（从glusterfs 3.9开始）

**配置HA Cluster**

要设置HA Cluster，请通过执行以下命令启用NFS-Ganesha：

#gluster nfs-ganesha enable

要禁用HA群集，请执行以下命令：

#gluster nfs-ganesha disable

Note :

Enable command performs the following

\* create a symlink ganesha.conf in /etc/ganesha using ganesha.conf in shared storage

\* start nfs-ganesha process on nodes part of ganesha cluster

\* set up ha cluster

and disable does the reversal of enable

Also if gluster nfs-ganesha [enable/disable] fails of please check following logs

\* /var/log/glusterfs/glusterd.log

\* /var/log/messages (and grep for pcs commands)

\* /var/log/pcsd/pcsd.log

**使用cli通过NFS-Ganesha导出卷**

要export Red Hat Gluster存储卷，请执行以下命令：

#gluster volume set <volname> ganesha.enable on

要unexport Red Hat Gluster存储卷，请执行以下命令：

#gluster volume set <volname> ganesha.enable off

此命令unexport Red Hat Gluster存储卷而不影响其他导出。

要验证卷集选项的状态，请遵循以下指导原则：

通过执行以下命令检查NFS-Ganesha是否已启动：

ps aux | grep ganesha.nfsd

检查卷是否已导出。

showmount -e localhost

ganesha.nfsd守护程序的日志将写入/var/log/ganesha.log。检查日志文件，注意任何意外行为。

**使用ganesha-ha.sh脚本修改HA cluster**

要修改现有HA群集并更改导出的默认值，请使用位于/ usr / libexec / ganesha /的ganesha-ha.sh脚本。

**将节点添加到群集**

在将节点添加到群集之前，请确保Pre-requisites to run NFS-Ganesha满足部分中提到的所有先决条件。将节点添加到cluster。在现有NFS-Ganesha集群中的任何节点上执行以下命令：

#./ganesha-ha.sh --add <HA\_CONF\_DIR> <HOSTNAME> <NODE-VIP>

where,

HA\_CONF\_DIR: The directory path containing the ganesha-ha.conf file.

HOSTNAME: Hostname of the new node to be added

NODE-VIP: Virtual IP of the new node to be added.

**删除群集中的节点**

要从群集中删除节点，请在现有NFS-Ganesha群集中的任何节点上执行以下命令：

#./ganesha-ha.sh --delete <HA\_CONF\_DIR> <HOSTNAME>

where,

HA\_CONF\_DIR: The directory path containing the ganesha-ha.conf file.

HOSTNAME: Hostname of the new node to be added

**修改默认导出配置**

要修改默认导出配置，请在现有ganesha集群中的任何节点上执行以下步骤：

在位于的相应导出文件中编辑/添加必填字段/etc/ganesha/exports。

执行以下命令：

#./ganesha-ha.sh --refresh-config <HA\_CONFDIR> <volname>

where,

HA\_CONF\_DIR: The directory path containing the ganesha-ha.conf file.

volname: The name of the volume whose export configuration has to be changed.

注意不得更改导出ID。⁠

**在gluster节点之外配置ganesha ha cluster**

目前ganesha HA集群创建与glusterd紧密集成。所以这里用户需要使用ganesha节点创建另一个TSP。然后使用上述步骤创建ganesha HA集群，直到执行“gluster nfs-ganesha enable”导出/取消导出应该在不使用glusterd cli的情况下执行（按照手动步骤，在执行步骤4之前将localhost替换为所需的主机名/ ip“hostname = localhost; “在导出配置文件中

**为pNFS配置Gluster卷**

并行网络文件系统（pNFS）是NFS v4.1协议的一部分，允许计算客户端直接和并行访问存储设备。pNFS集群由MDS（元数据服务器）和DS（数据服务器）组成。客户端将所有读/写请求直接发送到DS，所有其他操作由MDS处理。

步骤：

打开卷的feature.cache-invalidation。

gluster v set \ <volname> features.cache-invalidation on

选择群集中的一个节点作为MDS，并将其配置为将以下块添加到ganesha配置文件中

GLUSTER

{

PNFS\_MDS = true;

}

在群集中的每个节点中手动启动NFS-Ganesha。

检查是否在所有节点中通过nfs-ganesha导出卷。

#showmount -e localhost

使用NFS版本4.1协议和MDS的ip挂载卷

#mount -t nfs4 -o minorversion = 1 \ <MDS的ip>：/ \ <卷名> \ <mount path>

需要注意的要点：

当前架构仅支持单个MDS和多个DS。客户端安装的服务器将充当MDS，包括MDS在内的所有服务器都可以充当DS。

目前，pNFS（更具体地说是MDS）不支持HA。虽然它是可配置的，但整个群集的一致性得到保证。

如果任何DS发生故障，那么MDS将处理这些I / O.

此后，所有后续NFS客户端都需要使用相同的服务器通过pNFS安装该卷。即不优先考虑卷的多个MDS

仅对分布式，复制或分发复制卷pNFS支持。它使用RHEL 6.5，fedora 20，fedora 21 nfs客户端进行测试和验证。使用最新的nfs-clients总是更好

## 4.6 特征

[地域复制](https://docs.gluster.org/en/latest/Administrator%20Guide/Geo%20Replication/)

[配额](https://docs.gluster.org/en/latest/Administrator%20Guide/Directory%20Quota/)

[快照](https://docs.gluster.org/en/latest/Administrator%20Guide/Managing%20Snapshots/)

[垃圾](https://docs.gluster.org/en/latest/Administrator%20Guide/Trash/)

## 4.7 使用其他接口进行数据访问

### 4.7.1 对象存储

SwiftOnFile项目使GlusterFS卷可以用作Openstack Swift的后端 - 一个分布式对象存储。这允许通过Swift的RESTful API将对象PUT作为文件系统接口上的文件进行访问，反之亦然，即通过文件系统接口（NFS / FUSE / native）创建的文件可以作为Swift的RESTful API上的对象进行访问。

SwiftOnFile项目以前gluster-swift也是UFO (Unified File and Object)如此。有关SwiftOnFile的更多信息，请访问[此处](https://github.com/swiftonfile/swiftonfile/blob/master/doc/markdown/quick_start_guide.md)。gluster-swift（现已过时）和swiftonfile项目的工作存在差异。较旧的gluster-swift代码和相关文档可以在 swiftonfile repo的icehouse [分支](https://github.com/swiftonfile/swiftonfile/tree/icehouse)中找到。

**SwiftOnFile vs gluster-swift**

| **Gluster-Swift** | **SwiftOnFile** |
| --- | --- |
| 一个GlusterFS卷只映射到并存储一个Swift帐户。Mountpoint层次结构：container/object | 一个GlusterFS卷或XFS分区可以有多个帐户。Mountpoint层次结构：acc/container/object |
| 覆盖帐户服务器，容器服务器和对象服务器。我们需要与上游Swift保持同步，并且通常可能需要更改代码或变通方法来支持新的Swift功能 | 仅实现对象服务器。因为代理，容器和帐户级别的新功能很可能与SwiftOnFile兼容，因为它只是一个存储策略，因此很少需要赶上Swift。 |
| 不对帐户和容器使用DB。容器列表涉及文件系统爬网。帐户/容器上的HEAD在没有FS爬行的情况下提供不准确或过时的结果。 | 使用Swift的DB来存储帐户和容器信息。帐户或容器列表不涉及FS爬网。HEAD到帐户/容器的准确信息 - 支持帐户配额的能力。 |
| 在容器和帐户上GET列出文件系统中的实际文件。 | 在容器和帐户上GET只列出Swift上的对象PUT。通过文件系统接口创建的文件不会出现在容器和对象列表中。 |
| 需要独立部署，不与现有Swift集群集成。 | 与任何现有的Swift部署集成为存储策略。 |

### 4.7.2 [使用Cinder主机访问GlusterFS](https://docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20Cinder/)

注意：自Ocata以来，GlusterFS驱动程序已从Openstack中删除。本指南仅适用于较旧的Openstack版本。

1.简介

GlusterFS和Cinder集成提供了一个数据存储系统，使用户能够以对象和文件的形式访问相同的数据，从而简化管理并控制存储成本。

GlusterFS - GlusterFS是一个开源的分布式文件系统，能够扩展到几PB并处理数千个客户端。GlusterFS通过Infiniband RDMA或TCP / IP互连将存储构建块集群在一起，聚合磁盘和内存资源，并在单个全局命名空间中管理数据。GlusterFS基于可堆叠的用户空间设计，可为各种工作负载提供卓越的性能。

Cinder - Cinder是OpenStack服务，负责处理虚拟机的持久存储。这是在Nova中运行的实例的持久块存储。可以使用快照进行备份和数据，以恢复数据或用于创建新的块存储卷。

使用Enterprise Linux 6，配置OpenStack Grizzly以将GlusterFS用于其Cinder（块）存储非常简单。

这些已经过GlusterFS 3.3和GlusterFS 3.4的测试。其他版本也可以使用，但尚未经过测试。

2.先决条件

ClusterFS:有关安装GlusterFS的先决条件和说明的信息，请参阅<http://www.gluster.org/community/documentation/index.php>。

Cinder:有关安装Cinder的先决条件和说明的信息，请参阅http://docs.openstack.org/。

在开始之前，您必须确保Cinder 中没有现有卷。使用“cinder delete”删除任何和“cinder list”以验证它们是否被删除。如果您不删除现有的cinder卷，则会在稍后的过程中导致错误，从而破坏Cinder安装。

注 - 与其他软件不同，“openstack-config”和“cinder”命令通常要求您以root用户身份运行它们。没有先前的配置，通过sudo运行它们通常是行不通的。（这可以改变，但超出了本HOW-TO的范围。）

3在Cinder主机上安装GlusterFS客户端

在每个Cinder主机上，安装GlusterFS客户端软件包：

$ sudo yum -y install glusterfs-fuse

4.配置Cinder以添加GlusterFS

在每个Cinder主机上，运行以下命令将GlusterFS添加到Cinder配置：

# openstack-config --set /etc/cinder/cinder.conf DEFAULT volume\_driver cinder.volume.drivers.glusterfs.GlusterfsDriver

# openstack-config --set /etc/cinder/cinder.conf DEFAULT glusterfs\_shares\_config /etc/cinder/shares.conf

# openstack-config --set /etc/cinder/cinder.conf DEFAULT glusterfs\_mount\_point\_base /var/lib/cinder/volumes

5.创建GlusterFS卷列表

在每个Cinder节点上，创建一个简单的文本文件/etc/cinder/shares.conf。

此文件是要使用的GlusterFS卷的简单列表，每行一个，使用以下格式：

GLUSTERHOST:VOLUME

GLUSTERHOST:NEXTVOLUME

GLUSTERHOST2:SOMEOTHERVOLUME

例如：

myglusterbox.example.org:myglustervol

6.更新GlusterFS的防火墙

您必须更新每个Cinder节点上的防火墙规则以与GlusterFS节点通信。

要打开的端口在步骤3中说明：

http://gluster.org/community/documentation/index.php/Gluster\_3.2:\_Installing\_GlusterFS\_on\_Red\_Hat\_Package\_Manager\_(RPM)\_Distributions

如果您使用iptables作为防火墙，可以在“\* filter”部分的OUTPUT ACCEPT下添加这些行。您可能应该根据您的环境调整它们（例如，只接受来自GlusterFS服务器的连接）。

-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 24007 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 24008 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 24009 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 24010 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 24011 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 38465:38469 -j ACCEPT

重启防火墙服务：

$ sudo service iptables restart

7.重新启动Cinder服务

配置已完成，现在您必须重新启动Cinder服务才能使其处于活动状态。

$ for i in api scheduler volume; do sudo service openstack-cinder-${i} start; done

检查Cinder卷日志以确保没有错误：

$ sudo tail -50 /var/log/cinder/volume.log

8.验证GlusterFS与Cinder

要验证安装和配置是否成功，请创建Cinder卷，然后使用GlusterFS进行检查。

创建一个Cinder卷：

# cinder create --display\_name myvol 10

创建卷需要几秒钟。创建后，运行以下命令：

# cinder list

卷应处于“可用”状态。现在，在GlusterFS卷目录中查找新文件：

$ sudo ls -lah /var/lib/cinder/volumes/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX/

（XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX将是特定于您的安装的编号）

新创建的文件应位于该目录中，该目录是您刚刚创建的新卷。每次创建卷时都会出现一个新文件。

例如：

$ sudo ls -lah /var/lib/cinder/volumes/29e55f0f3d56494ef1b1073ab927d425/

total 4.0K

drwxr-xr-x. 3 root root 73 Apr 4 15:46 .

drwxr-xr-x. 3 cinder cinder 4.0K Apr 3 09:31 ..

-rw-rw-rw-. 1 root root 10G Apr 4 15:46 volume-a4b97d2e-0f8e-45b2-9b94-b8fa36bd51b9

[带Keystone的GlusterFS](https://docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20Keystone%20Quickstart/)

[在ZFS之上安装Gluster](https://docs.gluster.org/en/latest/Administrator%20Guide/Gluster%20On%20ZFS/)

### 4.7.3 [配置Bareos以在Gluster上存储备份](https://docs.gluster.org/en/latest/Administrator%20Guide/Bareos/)

此描述假定您已准备好并配置了Gluster环境。这些示例storage.example.org用作循环DNS名称，可用于联系任何可用的GlusterD进程。调用使用的Gluster卷backups。客户端系统可以通过安装FUSE来访问卷，如下所示：

# mount -t glusterfs storage.example.org:/backups /mnt

Bareos包含一个使用的存储守护进程的插件 libgfapi。这使得Bareos可以访问Gluster卷而无需提供FUSE安装。

在这里，我们将使用一个专门用于备份的服务器。这个系统叫做backup.example.org。Bareos Director与Bareos Storage Daemon一起在这台主机上运行。在该示例中，在同一服务器上运行文件守护程序。这样就可以备份Bareos Director，它可以作为Bareos数据库的备份，并且可以保持配置。

**Bareos安装**

Bareos安装需要Bareos Director和Storage Daemon。为了备份文件系统，文件守护进程也需要可用。对于本文档中的描述，使用了CentOS-7，其中包含以下软件包和版本：

glusterfs-3.7.4

bareos-14.2 with bareos-storage-glusterfs

Gluster存储服务器不需要安装任何Bareos软件包。将应用程序（Bareos）和存储服务器保留在不同的系统上通常会更好。因此，在配置Bareos存储库后，在backup.example.org服务器上安装软件包：

# yum install bareos-director bareos-database-sqlite3 \

bareos-storage-glusterfs bareos-filedaemon \

bareos-bconsole

为了让事情尽可能简单，它使用了SQlite。对于生产部署，建议使用MySQL或PostgrSQL。需要创建初始数据库：

# sqlite3 /var/lib/bareos/bareos.db < /usr/lib/bareos/scripts/ddl/creates/sqlite3.sql

# chown bareos:bareos /var/lib/bareos/bareos.db

# chmod 0660 /var/lib/bareos/bareos.db

该bareos-bconsole软件包是可选的。bconsole是一个终端应用程序，可用于启动备份，检查不同Bareos组件的状态等。测试配置bconsole相对简单。

安装软件包后，您需要启动并启用守护程序：

# systemctl start bareos­sd

# systemctl start bareos­fd

# systemctl start bareos­dir

# systemctl enable bareos­sd

# systemctl enable bareos­fd

# systemctl enable bareos­dir

**Gluster卷准备**

允许Bareos访问Gluster卷需要几个步骤。默认情况下，Gluster不允许客户端从非特权端口进行连接。由于Bareos Storage Daemon不以root用户身份运行，因此需要打开连接权限。

客户端访问Gluster卷时涉及两个进程。在初始阶段，联系GlusterD，当客户端收到卷的布局时，客户端将直接连接到brick。因此，允许非特权进程连接的更改有两个方面：

1、在/etc/glusterfs/glusterd.vol该选项中rpc-auth-allow-insecure on 需要在所有存储服务器上添加。修改配置文件后，需要重新启动GlusterD进程 systemctl restart glusterd。

2、卷的brick进程通过卷选项配置。通过执行gluster volume set backups server.allow-insecure on所需的选项得到设置。某些版本的Gluster需要在考虑选项之前停止/启动音量，对于这些版本，您需要执行gluster volume stop backups和gluster volume start backups。

除网络权限外，还需要允许Bareos Storage Daemon写入Gluster Volume提供的文件系统。这是通过设置正常的UNIX权限/所有权来实现的，以便正确的用户/组可以写入卷：

# mount -t glusterfs storage.example.org:/backups /mnt

# mkdir /mnt/bareos

# chown bareos:bareos /mnt/bareos

# chmod ug=rwx /mnt/bareos

# umount /mnt

根据用户/组在环境中的维护方式，bareos 用户和组可能在存储服务器上不可用。如果是这样的情况下，chown上述命令可以适于使用uid与gid所述的bareos从用户和组backup.example.org。在Bareos服务器上，输出看起来类似于：

# id bareos

uid=998(bareos) gid=997(bareos) groups=997(bareos),6(disk),30(tape)

这使得chown命令看起来像这样：

# chown 998:997 /mnt/bareos

**Bareos配置**

当bareos-storage-glusterfs得到了安装，一个示例配置文件已经被添加了。该/etc/bareos/bareos-sd.d/device-gluster.conf包含Archive Device指令，这是该Gluster的音量和路径备份应该得到保存的URL。在我们的示例中，条目应设置为：

Device {

Name = GlusterStorage

Archive Device = gluster://storage.example.org/backups/bareos

Device Type = gfapi

Media Type = GlusterFile

...

}

Bareos提供的作业的默认配置是将备份写入 /var/lib/bareos/storage。为了将所有备份写入Gluster卷，需要修改Bareos Director的配置。在/etc/bareos/bareos-dir.conf配置中，可以更改所有作业的默认值以使用GlusterFile存储：

JobDefs {

Name = "DefaultJob"

...

# Storage = File

Storage = GlusterFile

...

}

更改配置文件后，Bareos守护程序需要应用它们。最简单的通知流程更改的配置文件是指示他们reload的配置：

# bconsole

Connecting to Director backup:9101

1000 OK: backup-dir Version: 14.2.2 (12 December 2014)

Enter a period to cancel a command.

\*reload

使用bconsole它还可以检查配置是否已应用。该status命令可用于显示已配置的存储的URL。如果所有设置都正确，结果如下所示：

\*status storage=GlusterFile

Connecting to Storage daemon GlusterFile at backup:9103

...

Device "GlusterStorage" (gluster://storage.example.org/backups/bareos) is not open.

...

**创建您的第一个备份**

Bareos Director中配置了多个默认作业。其中之一是DefaultJob在之前的步骤中进行了修改。此作业使用 SelfTestFileSet进行备份/usr/sbin。运行此作业将验证配置是否正常工作。可以在以后添加其他作业，其他文件集和更多文件守护程序（备份的客户端）。

\*run

A job name must be specified.

The defined Job resources are:

1: BackupClient1

2: BackupCatalog

3: RestoreFiles

Select Job resource (1-3): 1

Run Backup job

JobName: BackupClient1

Level: Incremental

Client: backup-fd

...

OK to run? (yes/mod/no): yes

Job queued. JobId=1

该作业需要几秒钟才能完成，该status命令可用于显示进度。完成后，该messages命令将显示结果：

\*messages

...

JobId: 1

Job: BackupClient1.2015-09-30\_21.17.56\_12

...

Termination: Backup OK

包含备份的存档将位于Gluster卷上。要检查文件是否可用，请将卷装入存储服务器：

# mount -t glusterfs storage.example.org:/backups /mnt

# ls /mnt/bareos

## 4.8 [GlusterFS服务日志和位置](https://docs.gluster.org/en/latest/Administrator%20Guide/Logging/)

下面列出了GlusterFS服务器中基于组件，服务和功能的日志。根据文件系统层次结构标准（FHS），所有日志文件都放在/var/log目录中。⁠

**glusterd：**

glusterd日志位于/var/log/glusterfs/glusterd.log。每个服务器一个glusterd日志文件。此日志文件还包含快照和用户日志。

**gluster cli命令：**

gluster cli日志位于/var/log/glusterfs/cli.log。在GlusterFS可信存储池中的节点上执行的Gluster命令已登录/var/log/glusterfs/cmd\_history.log。

**bricks：**

brick日志位于/var/log/glusterfs/bricks/<path extraction of brick path>.log。服务器上每块砖一个日志文件

**rebalance：**

重新平衡日志位于/var/log/glusterfs/VOLNAME-rebalance.log。服务器上每卷一个日志文件。

**Self heal deamon：**

Self heal deamon位于 /var/log/glusterfs/glustershd.log。每台服务器一个日志文件

**quota：**

/var/log/glusterfs/quotad.log是每个节点上运行的配额守护程序的日志。 /var/log/glusterfs/quota-crawl.log每当启用配额时，都会执行文件系统爬网，并且相应的日志将存储在此文件中。 /var/log/glusterfs/quota-mount- VOLNAME.log安装了辅助FUSE客户端glusterFS的/ VOLNAME以及此文件中找到的相应客户端日志。

One log file per server (and per volume from quota-mount.

**Gluster NFS：**

/var/log/glusterfs/nfs.log 每台服务器一个日志文件

**SAMBA Gluster：**

/var/log/samba/glusterfs-VOLNAME-<ClientIP>.log。如果客户端在glusterFS服务器节点上安装它，则可能找不到实际的日志文件或挂载点。在这种情况下，需要考虑所有glusterFS类型安装操作的安装输出。

**Ganesha NFS：**

/var/log/nfs-ganesha.log

**FUSE Mount：**

/var/log/glusterfs/<mountpoint path extraction>.log

**Geo-replication：**

/var/log/glusterfs/geo-replication/<master> /var/log/glusterfs/geo-replication-slaves

**gluster volume heal VOLNAME info命令：**

/var/log/glusterfs/glfsheal-VOLNAME.log。每个执行命令的服务器的一个日志文件。

**gluster-swift：**

/var/log/messages

**SwiftKrbAuth：**

/var/log/httpd/error\_log

## 4.9 [监控Gluster工作负载](https://docs.gluster.org/en/latest/Administrator%20Guide/Monitoring%20Workload/)

您可以在不同参数上监视GlusterFS卷。监视卷有助于GlusterFS卷的容量规划和性能调整任务。使用这些信息，您可以识别问题并进行故障排除。

您可以使用Volume Top和Profile命令查看性能并识别卷的每个brick的瓶颈/热点。这有助于系统管理员在需要探测性能时获取重要的性能信息。

您还可以执行卷进程的声明卷和卷的nfs服务器进程，还可以查看卷状态和卷信息。

### 4.9.1 运行GlusterFS卷配置文件命令

GlusterFS Volume Profile命令提供了一个接口，用于获取卷的每个文件操作（FOP）的每块砖I / O信息。每砖信息有助于识别存储系统中的瓶颈。

本节介绍如何通过执行以下操作来运行GlusterFS Volume Profile命令：

#### 4.9.1.1 开始分析

您必须启动性能分析以查看每个块的文件操作信息。

要开始分析，请使用以下命令：

# gluster volume profile start

例如，要在测试卷上开始分析：

# gluster volume profile test-volume start

Profiling started on test-volume

启动卷上的分析时，“卷信息”中将显示以下附加选项：

diagnostics.count-fop-hits: on

diagnostics.latency-measurement: on

#### 4.9.1.2 显示I / 0信息

您可以使用以下命令查看每个brick的I / O信息：

# gluster volume profile info

例如，要查看测试卷上的I / O信息：

# gluster volume profile test-volume info

#### 4.9.1.3 停止分析

如果您不再需要分析信息，则可以停止分析卷。

使用以下命令停止分析：

# gluster volume profile stop

例如，要停止对测试卷进行分析：

# gluster volume profile stop

### 4.9.2 运行GlusterFS卷TOP命令

GlusterFS Volume Top命令允许您查看glusterfs brick的性能指标，如读取，写入，文件打开调用，文件读取调用，文件写入调用，目录打开调用和目录实际调用。top命令最多显示100个结果。

本节介绍如何运行和查看以下GlusterFS Top命令的结果

#### 4.9.2.1 查看打开fd计数和最大fd计数

您可以查看当前打开的fd计数（当前打开的文件列表和计数）以及最大打开fd计数（当前打开的文件数和打开的最大文件数）任何给定的时间点，因为服务器已启动并运行）。如果未指定砖名称，则将显示属于该卷的所有brick的打开fd度量标准。

使用以下命令查看open fd count和maximum fd count：

# gluster volume top open [brick ] [list-cnt ]

例如，要查看brick服务器上的open fd count和maximum fd count：/ test test-volume和list top 10 open call：

# gluster volume top open brick list-cnt

#### 4.9.2.2 查看最高文件读取呼叫

您可以在每个brick上查看最高读取呼叫。如果未指定brick name，则默认情况下将显示100个文件的列表。

查看最高文件使用以下命令读取调用：

# gluster volume top read [brick ] [list-cnt ]

例如，要在brick服务器上查看最高的Read调用：/ test-volume的export：

# gluster volume top read brick list-cnt

#### 4.9.2.3 查看最高文件写入调用

您可以查看每个块上具有最高文件写入调用的文件列表。如果未指定brick name，则默认情况下将显示100个文件的列表。

查看最高文件使用以下命令写入调用：

# gluster volume top write [brick ] [list-cnt ]

例如，要在brick服务器上查看最高的Write调用：/ test-volume的export：

# gluster volume top write brick list-cnt

#### 4.9.2.4 查看目录上最高的公开调用

您可以查看每个块的目录上具有最高打开调用的文件列表。如果未指定brick名称，则将显示属于该卷的所有brick的度量标准。

使用以下命令查看每个目录上的打开调用列表：

# gluster volume top opendir [brick ] [list-cnt ]

例如，要查看brick服务器上的打开调用：/ export / of test-volume：

# gluster volume top opendir brick list-cnt

#### 4.9.2.5 查看目录上的最高读取调用

您可以查看每个块上具有最高目录读取调用的文件列表。如果未指定brick名称，则将显示属于该卷的所有brick的度量标准。

使用以下命令查看每个块上最高目录读取调用的列表：

# gluster volume top readdir [brick ] [list-cnt ]

例如，要在brick服务器上查看最高目录读取调用：/ test-volume：

# gluster volume top readdir brick list-cnt

#### 4.9.2.6 查看每个Brick上的读取性能列表

您可以查看每块brick上文件的读取吞吐量。如果未指定砖名称，则将显示属于该卷的所有brick的度量标准。输出将是读取吞吐量。

此命令将为指定的计数和块大小启动dd并测量相应的吞吐量。

使用以下命令查看每块砖上的读取性能列表：

# gluster volume top read-perf [bs count ] [brick ] [list-cnt ]

例如，要查看brick服务器上的读取性能：/ export / of test-volume，256块大小的计数1和列表计数10：

# gluster volume top read-perf bs 256 count 1 brick list-cnt

#### 4.9.2.7 查看每个Brick的写性能列表

您可以在每个brick上查看文件的写入吞吐量列表。如果未指定砖名称，则将显示属于该卷的所有brick的度量标准。输出将是写吞吐量。

此命令将为指定的计数和块大小启动dd并测量相应的吞吐量。要查看每个brick上的写入性能列表：

使用以下命令查看每块砖上的写入性能列表：

# gluster volume top write-perf [bs count ] [brick ] [list-cnt ]

例如，要查看brick服务器上的写入性能：/ export / of test-volume，256块大小的计数1和列表计数10：

# gluster volume top write-perf bs 256 count 1 brick list-cnt

#### 4.9.2.8 显示卷信息

您可以根据需要显示有关特定卷或所有卷的信息。

使用以下命令显示有关特定卷的信息：

# gluster volume info ``VOLNAME

例如，要显示有关测试卷的信息：

# gluster volume info test-volume

Volume Name: test-volume

Type: Distribute

Status: Created

Number of Bricks: 4

Bricks:

Brick1: server1:/exp1

Brick2: server2:/exp2

Brick3: server3:/exp3

Brick4: server4:/exp4

使用以下命令显示有关所有卷的信息：

# gluster volume info all

# gluster volume info all

Volume Name: test-volume

Type: Distribute

Status: Created

Number of Bricks: 4

Bricks:

Brick1: server1:/exp1

Brick2: server2:/exp2

Brick3: server3:/exp3

Brick4: server4:/exp4

Volume Name: mirror

Type: Distributed-Replicate

Status: Started

Number of Bricks: 2 X 2 = 4

Bricks:

Brick1: server1:/brick1

Brick2: server2:/brick2

Brick3: server3:/brick3

Brick4: server4:/brick4

Volume Name: Vol

Type: Distribute

Status: Started

Number of Bricks: 1

Bricks:

Brick: server:/brick6

#### 4.9.2.9 在卷上执行Statedump

Statedump是一种机制，通过该机制，您可以在发出命令时获取所有内部变量的详细信息以及glusterfs进程的状态。您可以使用statedump命令执行卷进程和nfs服务器进程的声明。以下选项可用于确定要转储的信息：

mem - 转储brick的内存使用情况和内存池详细信息。

iobuf - 倾倒brick的iobuf细节。

priv - 转储已加载翻译的私人信息。

callpool - 转储卷的挂起调用。

fd - 转储卷的打开fd表。

inode - 转储卷的inode表。

**显示volume statedump**

1、使用以下命令显示卷或NFS服务器的声明项：

# gluster volume statedump [nfs] [all|mem|iobuf|callpool|priv|fd|inode]

例如，要显示测试卷的声明：

# gluster volume statedump test-volume

Volume statedump successful

statedump文件是在/tmp 目录中的brick服务器上或使用server.statedump-path volume选项设置的目录中创建的。转储文件的命名约定是 <brick-path>.<brick-pid>.dump。

2、通过defult，陈述的输出存储在/tmp/<brickname.PID.dump>该特定服务器上的 文件中。使用以下命令更改statedump文件的目录：

# gluster volume set server.statedump-path

例如，要更改test-volume的statedump文件的位置：

# gluster volume set test-volume server.statedump-path /usr/local/var/log/glusterfs/dumps/

Set volume successful

您可以使用以下命令查看statedump文件的已更改路径：

# gluster volume info

#### 4.9.2.10 显示卷状态

您可以根据需要显示有关特定卷，砖或所有卷的状态信息。状态信息可用于了解brick，nfs进程和整个文件系统的当前状态。状态信息还可用于监视和调试卷信息。您可以查看卷的状态以及以下详细信息：

detail - 显示有关brick的其他信息。

clients - 显示连接到卷的客户端列表。

mem - 显示brick的内存使用情况和内存池详细信息。

inode - 显示卷的inode表。

fd - 显示卷的open fd（文件描述符）表。

callpool - 显示卷的挂起调用。

**显示卷状态**

1、使用以下命令显示有关特定卷的信息：

# gluster volume status [all| []] [detail|clients|mem|inode|fd|callpool]

例如，要显示有关测试卷的信息：

# gluster volume status test-volume

STATUS OF VOLUME: test-volume

BRICK PORT ONLINE PID

--------------------------------------------------------

arch:/export/1 24009 Y 22445

--------------------------------------------------------

arch:/export/2 24010 Y 22450

2、使用以下命令显示有关所有卷的信息：

# gluster volume status all

# gluster volume status all

STATUS OF VOLUME: volume-test

BRICK PORT ONLINE PID

--------------------------------------------------------

arch:/export/4 24010 Y 22455

STATUS OF VOLUME: test-volume

BRICK PORT ONLINE PID

--------------------------------------------------------

arch:/export/1 24009 Y 22445

--------------------------------------------------------

arch:/export/2 24010 Y 22450

3、使用以下命令显示有关brick的其他信息：

# gluster volume status detail

例如，要显示有关测试卷砖的其他信息：

# gluster volume status test-volume details

STATUS OF VOLUME: test-volume

-------------------------------------------

Brick : arch:/export/1

Port : 24009

Online : Y

Pid : 16977

File System : rootfs

Device : rootfs

Mount Options : rw

Disk Space Free : 13.8GB

Total Disk Space : 46.5GB

Inode Size : N/A

Inode Count : N/A

Free Inodes : N/A

Number of Bricks: 1

Bricks:

Brick: server:/brick6

4、使用以下命令显示访问卷的客户端列表：

# gluster volume status clients

例如，要显示连接到test-volume的客户端列表：

# gluster volume status test-volume clients

Brick : arch:/export/1

Clients connected : 2

Hostname Bytes Read BytesWritten

-------- --------- ------------

127.0.0.1:1013 776 676

127.0.0.1:1012 50440 51200

5、使用以下命令显示brick的内存使用情况和内存池详细信息：

# gluster volume status mem

例如，要显示测试卷砖的内存使用情况和内存池详细信息：

Memory status for volume : test-volume

----------------------------------------------

Brick : arch:/export/1

Mallinfo

--------

Arena : 434176

Ordblks : 2

Smblks : 0

Hblks : 12

Hblkhd : 40861696

Usmblks : 0

Fsmblks : 0

Uordblks : 332416

Fordblks : 101760

Keepcost : 100400

Mempool Stats

-------------

Name HotCount ColdCount PaddedSizeof AllocCount MaxAlloc

---- -------- --------- ------------ ---------- --------

test-volume-server:fd\_t 0 16384 92 57 5

test-volume-server:dentry\_t 59 965 84 59 59

test-volume-server:inode\_t 60 964 148 60 60

test-volume-server:rpcsvc\_request\_t 0 525 6372 351 2

glusterfs:struct saved\_frame 0 4096 124 2 2

glusterfs:struct rpc\_req 0 4096 2236 2 2

glusterfs:rpcsvc\_request\_t 1 524 6372 2 1

glusterfs:call\_stub\_t 0 1024 1220 288 1

glusterfs:call\_stack\_t 0 8192 2084 290 2

glusterfs:call\_frame\_t 0 16384 172 1728 6

6、使用以下命令显示卷的inode表：

# gluster volume status inode

例如，要显示测试卷的inode表：

# gluster volume status test-volume inode

inode tables for volume test-volume

----------------------------------------------

Brick : arch:/export/1

Active inodes:

GFID Lookups Ref IA type

---- ------- --- -------

6f3fe173-e07a-4209-abb6-484091d75499 1 9 2

370d35d7-657e-44dc-bac4-d6dd800ec3d3 1 1 2

LRU inodes:

GFID Lookups Ref IA type

---- ------- --- -------

80f98abe-cdcf-4c1d-b917-ae564cf55763 1 0 1

3a58973d-d549-4ea6-9977-9aa218f233de 1 0 1

2ce0197d-87a9-451b-9094-9baa38121155 1 0 2

7、使用以下命令显示卷的打开fd表：

# gluster volume status fd

例如，要显示测试卷的open fd表：

# gluster volume status test-volume fd

FD tables for volume test-volume

----------------------------------------------

Brick : arch:/export/1

Connection 1:

RefCount = 0 MaxFDs = 128 FirstFree = 4

FD Entry PID RefCount Flags

-------- --- -------- -----

0 26311 1 2

1 26310 3 2

2 26310 1 2

3 26311 3 2

Connection 2:

RefCount = 0 MaxFDs = 128 FirstFree = 0

No open fds

Connection 3:

RefCount = 0 MaxFDs = 128 FirstFree = 0

No open fds

8、使用以下命令显示卷的挂起调用：

# gluster volume status callpool

每个调用都有一个包含调用帧的调用堆栈。

例如，要显示测试卷的挂起调用：

# gluster volume status test-volume

Pending calls for volume test-volume

----------------------------------------------

Brick : arch:/export/1

Pending calls: 2

Call Stack1

UID : 0

GID : 0

PID : 26338

Unique : 192138

Frames : 7

Frame 1

Ref Count = 1

Translator = test-volume-server

Completed = No

Frame 2

Ref Count = 0

Translator = test-volume-posix

Completed = No

Parent = test-volume-access-control

Wind From = default\_fsync

Wind To = FIRST\_CHILD(this)->fops->fsync

Frame 3

Ref Count = 1

Translator = test-volume-access-control

Completed = No

Parent = repl-locks

Wind From = default\_fsync

Wind To = FIRST\_CHILD(this)->fops->fsync

Frame 4

Ref Count = 1

Translator = test-volume-locks

Completed = No

Parent = test-volume-io-threads

Wind From = iot\_fsync\_wrapper

Wind To = FIRST\_CHILD (this)->fops->fsync

Frame 5

Ref Count = 1

Translator = test-volume-io-threads

Completed = No

Parent = test-volume-marker

Wind From = default\_fsync

Wind To = FIRST\_CHILD(this)->fops->fsync

Frame 6

Ref Count = 1

Translator = test-volume-marker

Completed = No

Parent = /export/1

Wind From = io\_stats\_fsync

Wind To = FIRST\_CHILD(this)->fops->fsync

Frame 7

Ref Count = 1

Translator = /export/1

Completed = No

Parent = test-volume-server

Wind From = server\_fsync\_resume

Wind To = bound\_xl->fops->fsync

## 4.10 使用SSL / TLS设置GlusterFS

GlusterFS允许使用OpenSSL库使用传输层安全标准（取代安全套接字层） 来保护其通信。设置此项需要具备一些SSL / TLS概念的基本工作知识，这里只能简要概述。

1）“认证”是向第二实体证明其身份的一个实体（例如，机器，过程或人）的过程。

2）“授权”是检查实体是否具有执行操作的权限的过程。

3）TLS提供身份验证和加密。虽然GlusterFS可以使用经过TLS身份验证的身份来授权客户端与brick/卷的连接，但它不提供授权。

4）必须向第二实体Y进行认证的实体X通过与Y共享证书来实现，该证书包含足以证明X的身份的信息。X的身份证明还要求拥有 与其证书相匹配的私钥，但Y或其他任何人都不会看到此密钥。由于证书已公开，因此拥有密钥的任何人都可以声明该身份。

5）每个证书都包含其主体（所有者）的身份以及可以验证证书内容完整性的证书颁发机构或CA 的身份。委托人和CA可以是相同的（“自签名证书”）。如果它们不同，则CA必须通过附加从证书内容和CA自己的私钥派生的信息来签署证书。

6）证书签名关系可以扩展到多个级别。例如，公司X可以签署另一家公司Y的证书，然后可以使用该证书为特定用户或目的签署第三证书Z. 任何信任X（并且愿意通过两个或更多证书深度扩展该信任 ）的人都能够对Y和Z进行身份验证。

7）任何愿意接受其他实体的身份验证尝试的实体都必须拥有某种已接受证书的数据库。

在GlusterFS的情况下，客户端或服务器X使用以下文件来包含与TLS相关的信息：

1）/etc/ssl/glusterfs.pem X自己的证书

2）/etc/ssl/glusterfs.key X的私钥

3）/etc/ssl/glusterfs.ca串联其他人的证书

GlusterFS始终执行相互身份验证，但客户端当前不对经过身份验证的服务器标识执行任何操作。因此，如果客户端X想要与服务器Y通信，则X的证书（或签名者的证书）必须在Y的CA文件中，反之亦然。

对于GlusterFS中TLS的所有使用，如果连接的一端配置为使用TLS，则另一端也必须使用它。没有自动回退到非TLS通信，或允许并发TLS和非TLS访问同一资源，因为任何一个都是不安全的。相反，任何此类“混合模式”连接将被TLS使用方拒绝，牺牲可用性来维护安全性。

### 4.10.1 在I / O路径上启用TLS

要在客户端和砖服务器之间启用身份验证和加密，必须设置两个选项：

gluster volume set MYVOLUME client.ssl on

gluster volume set MYVOLUME server.ssl on

请注意，上述选项仅影响GlusterFS本机协议。NFS，SMB或Swift等外部协议不会受到影响。

### 4.10.2 使用TLS身份进行授权

在I / O路径上启用TLS后，可以使用TLS标识代替IP地址或普通用户名来控制对特定卷的访问。例如：

gluster volume set MYVOLUME auth.ssl-allow Zaphod

在这里，我们允许经过TLS身份验证的身份“Zaphod”访问MYVOLUME。这有意与现有的“auth.allow”选项相同，只是名称取自TLS证书而不是命令行字符串。请注意，gluster CLI中的不足会阻止使用包含空格的名称，否则这些名称将被允许。

### 4.10.3 在管理路径上启用TLS

管理守护程序流量不受选项控制。相反，它由每台机器上存在的文件控制：

/var/lib/glusterd/secure-access

创建此文件将导致从该计算机建立的glusterd连接使用TLS。请注意，即使是客户端也必须在安装时与远程glusterd进行通信，但此后不能这样做。

### 4.10.4 其他选项

GlusterFS TLS实现支持与TLS内部相关的两个附加选项。

第一个选项允许用户设置证书深度，如上所述。

gluster volume set MYVOLUME ssl.certificate-depth 2

在这里，我们将证书深度设置为两个，如介绍性示例中所示。默认情况下，此值为零，这意味着只接受在本地CA文件中直接指定的证书（即根本没有签名证书）。

第二个选项允许用户指定允许的TLS密码集。

gluster volume set MYVOLUME ssl.cipher-list 'HIGH:!SSLv2'

密码列表由TLS连接的双方协商，以满足双方的安全需求。在此示例中，我们将初始密码列表设置为HIGH，表示密码学社区仍然认为不会中断的密码。我们还明确禁止特定于SSL版本2的密码。默认值基于此示例，但也不包括基于CBC的密码模式，以提供针对POODLE攻击的额外缓解 。

## 4.11 [Puppet Gluster](https://docs.gluster.org/en/latest/Administrator%20Guide/Puppet/)

请参考：<https://docs.gluster.org/en/latest/Administrator%20Guide/Puppet/>

下载puppet-gluster：<https://github.com/purpleidea/puppet-gluster/>

## 4.12 RDMA Transport

GlusterFS支持使用RDMA协议在glusterfs客户端和glusterfs砖之间进行通信。GlusterFS客户端包括FUSE客户端，libgfapi客户端（包括Samba和NFS-Ganesha），gNFS服务器以及与自我修复守护程序，引用，重新平衡过程等brick进行通信的其他glusterfs进程。

注意：截至目前，只有FUSE客户端和gNFS服务器支持RDMA传输。

**注意：**

NFS客户端到gNFS服务器/ NFS Ganesha服务器通信仍将通过tcp发生。

CIFS客户端/ Windows客户端到Samba服务器通信仍然会通过tcp发生。

### 4.12.1 建立

请参阅这些外部文档，在您的机器上设置RDMA

http://pkg-ofed.alioth.debian.org/howto/infiniband-howto.html

http://people.redhat.com/dledford/infiniband\_get\_started.html

### 4.12.2 创建可信存储池

如果在存储池中创建了RDMA或TCP，RDMA卷，则受信任存储池中的所有服务器都必须具有RDMA设备。

必须使用分配给RDMA设备的IP /主机名执行对等探测。

### 4.12.3 端口和防火墙

如果找到rdma设备，进程glusterd将侦听tcp和rdma。用于rdma的端口是24008.类似地，brick进程还将在两个端口上侦听使用传输“tcp，rdma”创建的卷。

确保更新防火墙以接受这些端口上的数据包。

### 4.12.4 Gluster Volume Create

卷可以支持一种或多种传输类型，以便在客户端和brick进程之间进行通信。支持的传输有三种类型，即tcp，rdma和tcp，rdma。

示例：通过InfiniBand创建包含四个存储服务器的分布式卷：

# gluster volume create test-volume transport rdma server1:/exp1 server2:/exp2 server3:/exp3 server4:/exp4

创建测试卷已成功

请启动卷以访问数据。

### 4.12.5 Changing Transport of volume

要更改现有卷的受支持传输类型，请按照以下过程操作：

注意：仅当使用分配给RDMA设备的IP /主机名创建卷时，才可以执行此操作。

1、使用以下命令卸载所有客户端上的卷：

# umount mount-point

2、使用以下命令停止卷：

# gluster volume stop volname

3、更改运输类型。

例如，要启用tcp和rdma，请执行followimg命令：

# gluster volume set volname config.transport tcp,rdma

4、在所有客户端上安装卷。

例如，要使用rdma transport进行安装，请使用以下命令：

# mount -t glusterfs -o transport=rdma server1:/test-volume /mnt/glusterfs

注意：

config.transport选项在gluster cli的帮助下没有条目。

#gluster vol set help | grep config.transport

但是，密钥是有效的。

### 4.12.6 使用RDMA安装卷

您可以使用mount选项“transport”来指定FUSE客户端必须用于与砖进行通信的传输类型。如果仅使用一种传输类型创建卷，则在未指定任何值时，该值将成为默认值。在tcp，rdma卷的情况下，tcp是默认值。

例如，要使用rdma transport进行安装，请使用以下命令：

# mount -t glusterfs -o transport=rdma server1:/test-volume /mnt/glusterfs

### 4.12.7 Transport used by auxillary processes

所有辅助进程如自我修复守护进程，重新平衡进程等使用默认传输。如果你有一个tcp，rdma卷它将使用tcp。

在rdma音量的情况下，将使用rdma。

当卷为tcp时，选择这些进程使用的传输的配置选项，rdma尚不可用，将在以后的版本中提供。

## 4.13 [GlusterFS iSCSI](https://docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20iSCSI/)

### 4.13.1 介绍

可以使用Linux Target驱动程序设置Gluster上的iSCSI。这是一个接受iSCSI（以及i​​SER和FCoE）的用户空间守护程序。它根据用户配置解释iSCSI CDB并将其转换为其他I / O操作。在我们的例子中，我们可以将CDB转换为针对gluster文件运行的文件操作。该文件表示LUN和LBA文件中的偏移量。

已编写Linux目标驱动程序的插件以使用libgfapi。它是Linux目标驱动程序（bs\_glfs.c）的一部分。使用它，数据路径跳过FUSE。本文档将更新以描述如何使用它。您可以在Linux目标驱动程序的documentation子目录中看到README.glfs。

LIO是RHEL7中包含的Linux目标驱动程序的替代品。用户空间插件机制正在开发中。一旦存在该代码，就可以像为Linux目标驱动程序那样为gluster构建类似的机制。

下面是一个使用服务器上的Linux Target Driver进行设置的食谱。这已经在RHEL6，RHEL7和Fedora 19实例中的XEN和KVM实例上进行了测试。在此设置中，单个路径会导致gluster，这表示性能瓶颈和单点故障。对于HA和负载平衡，可以使用mpio设置两个或多个到不同gluster服务器的路径; 如果目标名称在每条路径上是等效的，那么mpio将使两条路径都无法进入单个设备。

### 4.13.2 Setup

在gluster服务器上本地挂载gluster。请注意，您也可以在gluster客户端上运行它。这些配置有利有弊，如下所述。

# mount -t glusterfs 127.0.0.1:gserver /mnt

在gluster fs中创建一个代表块设备的大文件。在这种情况下，伦是2G。（您也可以为此目的创建一个gluster“块设备”，它将跳过文件系统）。

# dd if=/dev/zero of=disk3 bs=2G count=25

使用该文件作为后端存储创建目标。

如有必要，请下载Linux SCSI目标。然后启动服务。

# yum install scsi-target-utils

# service tgtd start

您必须提供iSCSI限定名称（IQN），格式为：iqn.yyyy-mm.reversed.domain.name:OptionalIdentifierText

如下：

yyyy-mm表示设备启动时的4位数年份和2位数月份（例如：2011-07）

# tgtadm --lld iscsi --op new --mode target --tid 1 -T iqn.20013-10.com.redhat

你可以查看目标：

# tgtadm --lld iscsi --op show --mode conn --tid 1

Session: 11 Connection: 0 Initiator iqn.1994-05.com.redhat:cf75c8d4274d

接下来，向目标添加逻辑单元

# tgtadm --lld iscsi --op new --mode logicalunit --tid 1 --lun 1 -b /mnt/disk3

允许任何启动器访问目标。

# tgtadm --lld iscsi --op bind --mode target --tid 1 -I ALL

现在是时候设置你的客户了。

发现你的目标。请注意，在此示例中，目标IP地址为192.168.1.2

# iscsiadm --mode discovery --type sendtargets --portal 192.168.1.2

登录您的目标会话。

# iscsiadm --mode node --targetname iqn.2001-04.com.example:storage.disk1.amiens.sys1.xyz --portal 192.168.1.2:3260 --login

你应该有一个新的SCSI磁盘。您将在/ var / log / messages中看到它。你会在lsblk中看到它。

您可以向其发送I / O：

# dd if=/dev/zero of=/dev/sda bs=4K count=100

要拆除iSCSI连接：

# iscsiadm -m node -T iqn.2001-04.com.redhat -p 172.17.40.21 -u

### 4.13.2 在gluster客户端上运行iSCSI目标

您可以在gluster客户端上运行Linux目标守护程序。这种设置的优点是客户可以运行gluster并享受gluster的所有好处。例如，gluster可以“扇出”I / O到不同的gluster服务器。缺点是客户端需要加载和配置gluster。如果可能的话，最好在客户端上运行gluster。

**参考**

[1]http://www.linuxjournal.com/content/creating-software-backed-iscsi-targets-red-hat-enterprise-linux-6

[2] http://www.cyberciti.biz/tips/howto-setup-linux-iscsi-target-sanwith-tgt.html

## 4.14 Linux kernel tuning for GlusterFS

### 4.14.1 GlusterFS的Linux内核调优

不时有问题出现在内部和许多发烧友关于内核调优的Gluster所说的内容，如果有的话。

内核调优的罕见性是由于Linux内核在大多数工作负载上做得相当不错。但这种设计还有另一面。Linux内核历史上曾急切地占用了大量的RAM，只要有一些，或者将缓存作为提高性能的主要方式。

对于大多数情况来说，这很好，但随着工作负载量的增加和服务器上的集群负载增加，这变得很麻烦，导致工作的灾难性故障等。

我们有很多经验，无论是CAD，EDA还是类似的工具，我们都会看到大量内存系统（包括CAD，EDA或类似工具），我们有时会遇到Gluster的稳定性问题。我们必须仔细分析内存占用量和磁盘等待时间的数量。这给了我们一个非常了不起的故事，包括磁盘垃圾，巨大的iowaits，内核oops，磁盘挂起等。

本文是我在许多网站上执行调优选项的许多经验的结果。调整不仅有助于整体响应，而且还极大地稳定了整个集群。

当涉及内存调整时，旅程从“VM”子系统开始，该子系统具有奇怪的选项数量，这可能会导致很多混乱。

### 4.14.2 vm.swappiness

vm.swappiness是一个可调内核参数，用于控制内核优先于RAM交换的内容。在源代码级别，它也被定义为窃取映射内存的趋势。高swappiness值意味着内核更容易取消映射映射页面。低swappiness值意味着相反，内核将不太容易取消映射页面。换句话说，vm.swappiness值越高，系统交换的越多。

当有大量数据交换进RAM时，高系统交换会产生非常不良影响。许多人认为价值被设置得很高，但根据我的经验，将值设置为“0”会导致性能提升。

符合这里的细节 - http://lwn.net/Articles/100978/

但同样，这些变化应该由用户针对自己的应用程序进行的测试和尽职调查来推动。重载，流媒体应用程序应将此值设置为“0”。通过将此值更改为“0”，系统的响应性得到改善。

### 4.14.3 vm.vfs\_cache\_pressure

此选项控制内核回收用于缓存目录和inode对象的内存的趋势。

在默认值vfs\_cache\_pressure = 100时，内核将尝试以相对于pagecache和swapcache reclaim的“合理”速率回收dentries和inode。减少vfs\_cache\_pressure会导致内核更喜欢保留dentry和inode缓存。当vfs\_cache\_pressure = 0时，内核将永远不会因内存压力而回收dentries和inode，这很容易导致内存不足的情况。将vfs\_cache\_pressure增加到100会导致内核更喜欢回收dentries和inode。

使用GlusterFS，许多具有大量存储空间和许多小文件的用户由于“inode / dentry”缓存而很容易在服务器端使用大量RAM，从而导致内核不断爬行数据结构时性能下降40GB RAM系统。将此值更改为高于100可帮助许多用户实现内核的公平缓存和更高响应性。

### 4.14.4 vm.dirty\_background\_ratio

**vm.dirty\_ratio**

这两个中的第一个（vm.dirty\_background\_ratio）定义在将页面的背景刷新到磁盘之前可能变脏的内存百分比。在达到此百分比之前，没有页面刷新到磁盘。但是，当刷新开始时，它在后台完成，而不会破坏前台中的任何正在运行的进程。

现在，两个参数中的第二个（vm.dirty\_ratio）定义了强制刷新开始之前脏页可占用的内存百分比。如果脏页的百分比达到此阈值，则所有进程都变为同步，并且在它们请求的io操作实际执行且数据在磁盘上之前不允许它们继续。在高性能I / O机器的情况下，这会导致问题，因为数据缓存被切除并且所有进行I / O的进程都被阻塞以等待I / O. 这将导致大量悬挂过程，从而导致高负载，这导致系统不稳定和性能蹩脚。

将它们从标准值降低会导致所有内容都刷新到磁盘而不是存储在RAM中。它有助于大型内存系统，通常会将45G-90G页面缓存刷新到磁盘，从而导致前端应用程序的等待时间过长，从而降低整体响应能力和交互性。

**“1”> / proc / sys / vm / pagecache**

页面缓存是一个磁盘缓存，它保存来自文件和可执行程序的数据，即包含文件或块设备实际内容的页面。页面缓存（磁盘缓存）用于减少磁盘读取次数。值“1”表示1％的RAM用于此目的，因此大多数RAM都是从磁盘而不是RAM中获取的。在上述值更改后，此值有些腥。更改此选项不是必需的，但如果您仍然对控制pagecache感到偏执，则此值应该有所帮助。

**“deadline”> / sys / block / sdc / queue / scheduler**

I / O调度程序是Linux内核的一个组件，它决定如何为底层设备排队读取和写入缓冲区。使用智能RAID控制器理论上“noop”更好，因为Linux对（物理）磁盘几何结构一无所知，因此让控制器充分了解磁盘几何结构可以高效地处理请求。但“截止日期”似乎可以提升表现。您可以在Linux内核源文档中阅读有关它们的更多信息：linux / Documentation / block / \* iosched.txt。我还看到混合操作期间“读取”吞吐量增加（许多写入）。

**“256”> / sys / block / sdc / queue / nr\_requests**

这是I / O请求的大小，它们在由调度程序传送到磁盘之前被缓冲。某些控制器的内部队列大小（queue\_depth）大于I / O调度程序的nr\_requests，因此I / O调度程序没有太多机会正确排序和合并请求。截止日期或CFQ调度程序喜欢将nr\_requests设置为queue\_depth值的2倍，这是给定控制器的默认值。合并订单和请求有助于调度程序在巨大负载期间更具响应性。

**echo“16”> / proc / sys / vm / page-cluster**

page-cluster控制一次尝试写入交换的页数。它定义了交换I / O大小，在上面的示例中，根据64k的RAID条带大小添加“16”。使用swappiness = 0后这没有意义，但是如果你定义了swappiness = 10或20，那么当你的RAID条带大小为64k时，使用这个值会有所帮助。

**blockdev --setra 4096 / dev /** （例如： - sdb，hdc或dev\_mapper）

默认的块设备设置通常会导致许多RAID控制器的性能下降。添加上述选项，将预读设置为4096 \* 512字节扇区，至少对于流式拷贝，增加了速度，通过在内核用于准备I / O的时间段内提前读取来使HD的集成缓存饱和。它可能会放入下一次读取请求的缓存数据。如果使用潜在有用的驱动时间或加载高速缓存之外的数据，那么预读太多可能会导致大型文件上的随机I / O无法访问。

在文件系统级别推荐但尚未测试的一些其他杂项更改如下。确保您的文件系统知道数组中的条带大小和磁盘数。例如，条带大小为64K的raid5数组和6个磁盘（实际上是5个，因为在每个条带集中都有一个磁盘在执行奇偶校验）。这些建立在理论假设的基础上，并从RAID专家提供的各种其他博客/文章中收集。

- > ext4 fs，5个磁盘，64K条带，4K块中的单位

mkfs -text4 -E stride = \ $（（64/4））

- > xfs，5个磁盘，64K条带，512字节扇区的单位

mkfs -txfs -d sunit = \ $（（64 \* 2）） - d swidth = \ $（（5 \* 64 \* 2））

您可能需要考虑为流式大文件增加上述条带大小。

警告：对于某些类型的应用程序，上述更改非常主观。未经用户对其各自申请进行事先尽职调查，本文不保证任何任何好处。它应该只在预期的整体系统响应能力增加的情况下应用，或者如果它解决了持续的问题。

更多信息和有趣的文章/电子邮件/博客阅读

http://dom.as/2008/02/05/linux-io-schedulers/

http://www.nextre.it/oracledocs/oraclemyths.html

http://article.gmane.org/gmane.linux.raid/17546

https://lkml.org/lkml/2006/11/15/40

http://misterd77.blogspot.com/2007/11/3ware-hardware-raid-vs-linux-software.html

http://www.jejik.com/articles/2008/04/benchmarking\_linux\_filesystems\_on\_software\_raid\_1/

## 4.15[导出和Netgroup身份验证](https://docs.gluster.org/en/latest/Administrator%20Guide/Export%20And%20Netgroup%20Authentication/)

### 4.15.1 NFS的导出和Netgroups身份验证

此功能为Gluster的NFS服务器添加了Linux样式的导出和netgroups身份验证。更具体地说，此功能允许用户限制Gluster卷和Gluster卷中子目录的访问特定IP（导出身份验证）或网络组（网络组身份验证）或两者的组合。Netgroups在Unix环境中用于控制NFS导出，远程登录和远程shell的访问。每个网络组都有一个唯一的名称，并定义了一组主机，用户，组和其他网络组。此信息存储在文件中，gluster NFS服务器根据这些文件管理客户端的权限

### 4.15.2 含义和用法

目前，gluster可以通过简单的IP列表限制对卷的访问。但是，通过允许通过网络组管理大型IP列表，此功能使该功能更具可扩展性。此外，它还提供了更加精细的权限处理，如通配符支持，对某些客户端的只读权限等。

该文件/var/lib/glusterd/nfs/export包含可用作该服务器的客户端的计算机的详细信息。典型的导出条目使用以下格式：

/<export path> <host/netgroup> (options),..

这里导出名称可以是该卷内的gluster卷或子目录路径。接下来它包含主机/网络组的列表，后跟适用于该条目的选项。以“@”开头的字符串被视为网络组，而以@开头的字符串是主机。选项包括安装相关参数，现在的选项，如“sec”，“ro / rw”，“anonuid”有效参数。

如果\*被提及为host / netgroup字段，那么任何客户端都可以挂载该导出路径。

该文件/var/lib/glusterd/nfs/netgroup应提及导出文件中提到的每个网络组的扩展。典型的网络组条目如下所示：

<netgroup name> ng1000\nng1000 ng999\nng999 ng1\nng1 ng2\nng2 (ip1,ip2,..)

gluster NFS服务器将在特定时间间隔后检查这些文件的内容

### 4.15.3 卷选项

1、启用导出/网络组功能

gluster volume set <volname> nfs.exports-auth-enable on

2、更改gluster NFS服务器的刷新间隔

gluster volume set <volname> nfs.auth-refresh-interval-sec <time in seconds>

3、更改导出条目的缓存间隔

gluster volume set <volname> nfs.auth-cache-ttl-sec <time in seconds>

### 4.15.4 测试export / netgroup文件

用户应该能够在应用配置之前检查文件的有效性。“glusterfsd”命令现在具有以下可用于检查配置的附加参数： - --print-netgroups：验证netgroups文件并将其打印出来。例如， - glusterfsd --print-netgroups <name of the file>

--print-exports：验证导出文件并将其打印出来。

例如，

glusterfsd --print-export <name of the file>

### 4.15.5 需要注意的点

此功能目前不支持导出手册页中的所有选项，但我们可以轻松添加它们。

这些文件/var/lib/glusterd/nfs/export和/var/lib/glusterd/nfs/netgroup应该在设置之前创建nfs.exports-auth-enable的每个节点选择可信存储池。

这些文件由用户手动处理。因此，它们的内容可以在受信任的存储池中的gluster nfs服务器之间有所不同。即，可以为同一群集中的gluster NFS服务器提供不同的身份验证机制。

请勿混淆此功能和使用身份验证nfs.rpc-auth-allow，nfs.export-dir这可能会导致不一致。

### 4.15.6 故障排除

更改文件内容后，如果在认证机制中没有正确反映，只需使用volume stop重新启动服务器并启动，这样gluster NFS服务器将再次强制读取这些文件的内容。

## 4.16 Thin Arbiter volumes in gluster

### 4.16.1 Thin Arbiter volumes in gluster

Thin Arbiter是一种新型的quorum节点，与传统的arbiter brick相比，其中优质和坏数据的粒度较小。在这种类型的卷中，quorum被考虑在brick级而不是每个文件的基础上。如果在数据块上甚至有一个标记为坏（即需要修复）的文件，则该brick被认为对整个文件都不好。所以，即使是不同的文件，如果写入在其他数据砖上失败但是在这个“坏”brick上成功，我们将返回写入失败。

### 4.16.2 Why thin arbiter？

这是一种处理拉伸集群类型工作负载的解决方案，但它也可用于常规工作负载，以防用户对arbiter / 3路复制相比满足此类仲裁。精简arbiter节点可以放置在可信存储池之外，即thin arbiter是集群中的“拉伸”节点。即使该连接具有高延迟，该节点也可以放置在云端或任何位置。由于此节点仅在出现故障（或brick已关闭）的情况下才会参与并决定法定人数，因此在正常情况下不会影响性能。如果一切正常，执行任何文件操作的成本将低于arbiter。只有在第一次失败的情况下，I / O才会进入数据块并进入thin arbiter，直到修复完成。

### 4.16.3 Setting up thin arbiter volume

在节点上运行thin arbiter进程的命令：

#/usr/local/sbin/glusterfsd -N --volfile-id ta-vol -f /var/lib/glusterd/vols/thin-arbiter.vol --brick-port 24007 --xlator-option ta-vol-server.transport.socket.listen-port=24007

创建thin arbiter副本2卷：

#glustercli volume create <volname> --replica 2 <host1>:<brick1> <host2>:<brick2> --thin-arbiter <quorum-host>:<path-to-store-replica-id-file>

例如：

glustercli volume create testvol --replica 2 server{1..2}:/bricks/brick-{1..2} --thin-arbiter server-3:/bricks/brick\_ta --force

volume create: testvol: success: please start the volume to access data

### 4.16.4 thin Arbiter的工作原理

Thin arbiter节点上只会运行一个进程，该进程将用于更新所有卷上所有副本对的副本标识文件。副本ID文件以xattrs的形式包含好的和坏的数据块的信息。副本对将使用将在安装期间创建的相应副本标识文件。

1）读取事务：满足仲裁时允许读取。即

当所有数据块和瘦仲裁器都启动时：在数据砖上执行查找以找出好/坏块并从好块中提供内容。

当一块brick出现时：使用EIO进行FOP失败。

两块brick正在堆积：如果有两个数据砖，则在数据砖上进行查找以找出好/坏块，并且将从好砖中提供内容。一次查找足以找出该文件的好/坏副本并将其保留在inode上下文中。如果一个数据砖和thin arbiter brick都已启动，则在瘦仲裁器上完成xattrop以获取源（好）砖的信息。如果数据块（UP）也在thin arbiter上标记为源块，则在数据块上查找此文件以检查文件是否真正健康。如果文件良好，将从此块提供数据，否则将向用户返回EIO错误。

2）写入事务：thin arbiter不参与I / O，事务将选择在thin arbiter brick上进行风压操作，以确保在出现故障时保持最新的必要元数据。操作失败将导致使用xattrs中的源/接收器信息更新thin arbiter上的replica-id文件，就像它在AFR中发生的那样。

## 4.17 Trash for GlusterFS

### 4.17.1 Trash Translator

Trash translator将允许用户访问已删除或截断的文件。每个brick都将维护一个隐藏的.trashcan目录，该目录将用于存储从相应brick中删除或截断的文件。可以从挂载点访问所有这些.trashcan目录的聚合。为了避免名称冲突，在将原始文件名移动到废纸篓目录时会将时间戳附加到原始文件名。

### 4.17.2 含义和用法

除了访问由用户删除或截断的文件的主要用例外，垃圾翻译器还可以帮助内部操作，例如自我修复和重新平衡。在自我修复和重新平衡期间，可能会丢失关键数据。在这些情况下，trash translator可以帮助恢复丢失的数据。trash translator旨在拦截unlink，truncate和ftruncate fops，将当前文件的副本存储在trash目录中，然后在原始文件上执行fop。对于内部操作，文件存储在垃圾目录内的“internal\_op”文件夹下。

### 4.17.3 卷选项

1、glustervolume set<VOLNAME> features.trash <on/off>

此命令可用于在卷中启用垃圾箱翻译器。如果设置为on，则在volume start命令期间将在卷内的每个块中创建trash目录。默认情况下，转换器在卷启动期间加载但仍保持不起作用。在此选项的帮助下禁用垃圾箱不会从卷中删除垃圾目录甚至其内容。

2、gluster volume set <VOLNAME> features.trash-dir <name>

此命令用于将垃圾邮件目录重新配置为用户指定的名称。参数是有效的目录名称。将在此名称下的每个砖内创建目录。如果用户未指定，则垃圾箱翻译器将使用默认名称“.trashcan”创建垃圾箱目录。只有在打开垃圾翻译器时才能使用此功能。

3、gluster volume set<VOLNAME> features.trash-max-filesize <size>

此命令可用于根据文件大小过滤进入垃圾箱目录的文件。trash\_max\_filesize之上的文件将被直接删除/截断。大小的值后面可以跟随乘法后缀为KB（= 1024字节），MB（= 1024 \* 1024字节）和GB（= 1024 \* 1024 \* 1024字节）。默认大小设置为5MB。考虑到垃圾目录正在消耗glusterfs卷空间这一事实，垃圾邮件功能的实现方式是它直接删除/截断大小> 1GB的文件，即使此选项设置为大于1GB的某个值。

4、gluster volume set <VOLNAME> features.trash-elimination-path <path1> [，<path2> ,. 。。]

此命令可用于设置垃圾翻译器的消除模式。在删除/截断期间，驻留在此模式下的文件不会移动到垃圾箱目录。路径必须是卷中存在的有效路径。

5、gluster volume set <VOLNAME> features.trash-internal-op <on / off>

此命令可用于为自我修复和重新平衡等内部操作启用垃圾箱。默认情况下设置为关闭。

### 4.17.4 用例

以下步骤说明了从目录中删除文件的简单方案

1、创建一个简单的分发卷并启动它。

# gluster volume create test rhs:/home/brick

# gluster volume start test

2、启用trash translator

# gluster volume set test features.trash on

3、通过本机客户端挂载glusterfs卷，如下所示。

# mount -t glusterfs rhs:test /mnt

4、在mount中创建目录和文件。

# mkdir mnt/dir

# echo abc > mnt/dir/file

5、从装载中删除文件。

# rm mnt/dir/file -rf

在垃圾目录中签出。

# ls mnt/.trashcan

我们可以在垃圾邮件目录中找到已删除的文件，并在其文件名后附加时间戳。

例如，

[root@rh-host ~]# mount -t glusterfs rh-host:/test /mnt/test

[root@rh-host ~]# mkdir /mnt/test/abc

[root@rh-host ~]# touch /mnt/test/abc/file

[root@rh-host ~]# rm /mnt/test/abc/file

remove regular empty file ‘/mnt/test/abc/file’? y

[root@rh-host ~]# ls /mnt/test/abc

[root@rh-host ~]#

[root@rh-host ~]# ls /mnt/test/.trashcan/abc/

file2014-08-21\_123400

需要记住的要点

卷启动后，将在卷内创建垃圾箱目录，并通过挂载显示。禁用垃圾箱不会对其从山上的可见性产生任何影响。

即使不允许删除trash-directory，当前在其上发布删除时也会删除当前存放的垃圾邮件内容，并且只存在空的垃圾邮件目录。

已知问题

由于trash translator驻留在服务器端更高的translator（如AFR），DHT不知道重命名和截断操作由此translator完成，最终将文件移动到trash目录。除非直到基于完整路径的查找出现在已删除的文件上，否则这些文件可能无法从装载中看到。

## 4.18 裂脑和解决方法

### 4.18.1 裂脑-split brain

裂脑是两个或多个文件的复制副本变得不同的情况。当一个文件处于分裂的大脑中时，文件的数据或元数据在副本的brick之间存在不一致，并且没有足够的信息来权威地选择副本作为原始并且修复坏副本，尽管所有brick都是在线和在线。对于目录，还有一个条目分裂大脑，其中的文件可以在副本的块中具有不同的gfid /文件类型。分裂脑可能主要是由于两个原因：1。由于网络断开：客户端暂时失去与brick的连接。 - 有2块砖的副本对，server1上的brick1和server2上的brick2。 - Client1失去与brick2的连接，client2由于网络分裂而失去与brick1的连接。 - 来自client1的写入转到brick1，而来自client2的写入转到brick2，这只是裂脑。2. Gluster brick进程发生故障或返回错误： - Server1已关闭且server2已启动：写入发生在服务器2上。 - Server1出现，server2出现故障（Heal未发生/服务器2上的数据未在server1上复制）：写入发生在server1上。 - Server2出现：server1和server2都具有彼此独立的数据。

如果我们使用副本2卷，则不可能在不失去可用性的情况下防止裂脑。

### 4.18.2 处理裂脑的方法

在glusterfs中有解决分裂大脑的方法。您可以在此处查看有关如何解决裂脑的详细说明。此外，还有一些方法可以减少最终陷入裂脑情况的可能性。它们是：1、副本3卷；2、Arbiter卷

这两个都使用glusterfs的client-quorum选项来避免裂脑情况。

**Client quorum：**

这是在自动文件复制（此处为AFR）模块中实现的功能，用于防止复制/分布式复制卷的I / O路径中的分割脑。默认情况下，如果特定副本子文件不满足client-quorum，则它将变为只读。其他子系统（在dist-rep卷中）仍将具有R / W访问权限。在这里，您可以看到有关客户端仲裁的更多详细信息。

**副本2卷中的client quorum：**

在副本2卷中，不可能同时实现高可用性和一致性，而不会牺牲对分区的容忍度。如果我们将client-quorum选项设置为auto，那么第一块砖必须始终为up，而不管第二块砖的状态如何。如果只有第二块砖，则子体积变为只读。如果将quorum-type设置为fixed，并且quorum-count设置为1，那么我们最终可能会分裂脑。 - Brick1已启动且brick2已关闭。满足Quorum并在brick1上进行写入。 - Brick1下降，砖2出现（没有发生痊愈）。达到法定人数，写在brick2上。 - Brick1出现了。满足法定人数，但两块砖都有独立的写作 - 裂脑。为了避免这种情况，我们必须将仲裁计数设置为2，这将花费可用性。

**1、副本3卷：**

当我们使用副本计数3创建复制或分布式复制卷时，默认情况下cluster.quorum-type选项设置为auto。这意味着至少应该运行2个brick以满足法定人数并允许写入。这是副本3卷的建议设置，不应更改此设置。以下是它如何防止文件在裂脑中结束：

B1，B2和B3是复制品3卷的3块砖。1. B1和B2上升，B3下降。满足法定人数并在B1和B2上进行写入。2. B3上升，B2下降。满足法定人数并在B1和B3上进行写入。3. B2出现，B1下降。满足法定人数。但是当写入请求到来时，AFR发现B2和B3互相指责（B2表示某些写入在B3上待处理，而B3表示某些写入在B2上待处理），因此写入是不允许的并且因EIO而失败。

用于创建副本3卷的命令：

$gluster volume create <volname> replica 3 host1:brick1 host2:brick2 host3:brick3

**2.Arbiter volume：**

Arbiter提供副本2和副本3之间的最佳位置，用户希望副本3提供裂脑保护但不想投资3倍存储空间。仲裁器也是副本3卷，其中副本的第三块砖自动配置为仲裁节点。这意味着第三个砖只存储文件名和元数据，但不存储任何数据。这将有助于避免裂脑，同时提供与正常复制品3体积相同的一致性水平。

用于创建仲裁卷的命令：

$gluster volume create <volname> replica 3 arbiter 1 host1:brick1 host2:brick2 host3:brick3

命令的唯一区别是，我们需要arbiter 1在副本计数后再添加一个关键字。由于它也是副本3卷，因此默认情况下cluster.quorum-type选项设置为auto，并且至少需要2个砖才能满足仲裁并允许写入。由于仲裁块只有文件的名称和元数据，因此还有一些检查可以保证一致性。Arbiter的工作原理如下：

1、客户端在写入时采用完整文件锁（副本3占用范围锁）。

2、如果2个brick已经启动，并且如果其中一块是仲裁器，并且它将另一块brick归咎于brick，则所有FOP将因ENOTCONN而失败（传输端点未连接）。如果仲裁者没有责怪其他brick，则允许FOP继续进行。

3、如果2个brick已启动且仲裁器已关闭，则允许FOP。

4、如果只有一个brick，则不满足client-quorum，并且卷变为EROFS。

5、在所有情况下，如果在启动FOP之前只有一个源，并且如果FOP在该源上失败，则应用程序将收到ENOTCONN。

**副本3和Arbiter卷之间的差异：**

1、如果是副本3卷，我们将整个文件存储在所有brick中，建议使用相同大小的brick。但是在仲裁器的情况下，由于我们不存储数据，仲裁器砖的尺寸比其他brick的尺寸要小。

2、Arbiter是副本2和副本3卷之间的状态。如果我们只有Arbiter而另一个brick已经用完并且Arbiter砖将其他brick归咎于其他brick，那么我们就无法继续使用FOP。

3、与Arbiter相比，副本3提供高可用性，因为与Arbiter不同，副本3在所有3个块中具有完整的数据副本。

## 4.19 Arbiter volumes and quorum option

### 4.19.1 gluster中的Arbiter volumes and quorum option

arbiter卷是复制卷的特殊子集，旨在防止分裂脑并提供与普通副本3卷相同的一致性保证，而不消耗3倍的空间。

### 4.19.2 Arbiter configuration

创建卷的语法是：

# gluster volume create <VOLNAME> replica 3 arbiter 1 <NEW-BRICK> ...

注意：使用仲裁功能的卷只能是replica 3 arbiter 1

例如：

# gluster volume create testvol replica 3 arbiter 1 server{1..6}:/bricks/brick

volume create: testvol: success: please start the volume to access data

这意味着对于列出的每3块砖，其中1块是仲裁器。我们创造了6块砖。副本数量为3，系列中的每个第3块砖将成为副本子卷。由于我们有两组3，因此创建了一个由两个副本子卷组成的分布子卷。

每个副本子卷被定义为在3块砖中有1个仲裁器。仲裁砖取自每个副本子卷的末尾。

# gluster volume info

Volume Name: testvol

Type: Distributed-Replicate

Volume ID: ae6c4162-38c2-4368-ae5d-6bad141a4119

Status: Created

Number of Bricks: 2 x (2 + 1) = 6

Transport-type: tcp

Bricks:

Brick1: server1:/bricks/brick

Brick2: server2:/bricks/brick

Brick3: server3:/bricks/brick (arbiter)

Brick4: server4:/bricks/brick

Brick5: server5:/bricks/brick

Brick6: server6:/bricks/brick (arbiter)

Options Reconfigured :

transport.address-family: inet

performance.readdir-ahead: on `

Arbiter块将仅存储文件/目录名称（即树结构）和扩展属性（元数据），但不存储任何数据。即文件大小（如图所示ls -l）将为零字节。它还将存储其他gluster元数据，如.glusterfs文件夹及其内容。

注意：启用仲裁功能会自动将 客户端仲裁配置为“自动”。此设置不可更改。

**Arbiter brick（s）sizing**

由于Arbiter块不存储文件数据，因此其磁盘使用量将远远小于副本的其他块。brick的大小取决于您计划在卷中存储的文件数量。一个好的估计将是副本中文件数量的4kb倍。

### 4.19.3 Why Arbiter？

**复制卷中的裂脑**

当文件处于裂脑状态时，文件中的数据或元数据（权限，uid / gid，扩展属性等）之间存在不一致，并且我们没有足够的信息来授权选择虽然所有brick都在线并且在线，但复制为原始并且可以修复坏的副本。对于目录，还有一个入口分裂大脑，其中的文件在副本的块中具有不同的gfids /文件类型（比如一个是文件，另一个是同名目录）。

本文档 介绍如何使用gluster cli或挂载点解析裂脑中的文件。几乎总是，由于网络断开（客户端暂时失去与brick的连接）而发生分裂，并且由于glusterbrick进程停止或返回错误而很少发生。

**Server-quorum and some pitfalls**

本文档 提供了此功能的详细说明。server-quorum的卷选项是：

1）cluster.server-quorum-ratio值描述：0到100

2）cluster.server-quorum-type值描述：none | server如果设置为server，则此选项使指定的卷能够参与服务器端仲裁。如果设置为none，则不考虑单独的卷进行卷检查。

cluster.server-quorum-ratio是一个百分比数字，并且是群集范围的 - 即，您在同一个受信任池中的不同卷的比率不同。

对于双节点可信存储池，将此值设置为大于50％非常重要，这样两个彼此分离的节点就不会认为它们同时具有仲裁。对于双节点普通副本卷，这意味着两个节点都需要启动并运行。因此，没有HA /故障转移的概念。

有些用户从2个节点创建副本2卷，并且在没有brick的情况下同步探测“虚拟”节点，并启用服务器仲裁，比例为51％。这并不妨碍文件进入裂脑。例如，如果B1和B2是副本的brick/节点而B3是虚拟节点，我们仍然可以最终处于裂脑中：

1）B1下降，B2和B3上升。服务器 - 仲裁仍然是。文件由客户端修改。

2）B2下降，B1回升。满足服务器仲裁。客户端修改了相同的文件。

3）我们现在在B1和B2 ==> split-brain中的文件有不同的内容。

作者认为，如果您希望避免跨节点而不是I / O路径中的卷配置进行卷分配，则server-quorum非常有用。与在客户端仲裁中不同的是，当仲裁丢失时，卷变为只读，特定节点中服务器 - 仲裁的丢失会导致glusterd终止该节点上的brick进程（对于参与的卷），甚至无法读取。

**Client Quorum**

Client-quorum是AFR中实现的一项功能，用于防止复制/分布式复制卷的I / O路径中的分割大脑。默认情况下，如果特定副本子文件不满足client-quorum，则它将变为只读。其他子系统（在dist-rep卷中）仍将具有R / W访问权限。

以下卷集选项用于配置它：

1）cluster.quorum-type默认值：无值说明：none | auto | fixed如果设置为“fixed”，则此选项仅允许在该副本集中的活动块数（文件所属的块）中写入文件）大于或等于'quorum-count'选项中指定的计数。如果设置为“auto”，则此选项仅允许在文件数> = ceil（构成该副本的brick总数/ 2）时写入文件。如果副本的数量是偶数，那么还有一个检查：如果向上砖的数量正好等于n / 2，那么第一块砖必须是其中一块砖。如果它超过n / 2那么第一块砖就不一定是brick中的一块。

2）cluster.quorum-count值描述：允许写入的副本集中必须处于活动状态的块数。此选项与cluster.quorum-type = fixed选项一起使用，以指定要参与仲裁的活动块数。如果仲裁类型为auto，则此选项无意义。

3）cluster.quorum-reads缺省值：no值描述：yes | no如果quorum-reads设置为'yes'（或'true'或'on'），只有满足仲裁时才允许读取，否则read（和write）将返回ENOTCONN。如果设置为“no”（或“false”或“off”），则即使未满足仲裁也会提供读取，但写入将因EROFS而失败。

**副本2卷和副本3卷**

从上面的描述中可以清楚地看出，client-quorum无法真正应用于副本2卷:(无需花费HA）。如果quorum类型设置为auto，则根据前面给出的描述，第一块砖必须始终为up，而不管第二块砖的状态如何。IOW，如果只有第二块砖，则子程序变为EROFS，即没有HA。如果quorum-type设置为fixed，则quorum-count 必须为2才能防止split-brains（否则write1可以在brick1中成功，另一个在brick2 => split-brain中）。因此，出于所有实际目的，如果要在副本2卷中获得高可用性，建议不要启用client-quorum。

在副本3卷中，默认情况下启用client-quorum并设置为“auto”。这意味着需要2块砖才能使写入成功。以下是此配置如何防止文件以裂脑结束：

说B1，B2和B3是brick：1。B3下降，满足法定数量，写入发生在文件B1和B2上2. B3上升，B2下降，法定数量再次满足，写入发生在B1和B3上。3. B2出现，B1下降，达到法定人数。现在，当发出写入时，AFR发现B2和B3的挂起xattrs互相指责，因此不允许写入，并且EIO失败。

即使是副本3卷也有一个角落案例，文件最终可能会出现裂脑。AFR通常对写入的{offset，length}进行范围锁定。如果在非重叠{offset，length}的同一个文件上发生3次写入，并且每次写入都失败（仅）一个不同的brick，那么我们将文件的AFR xattrs相互指责。

### 4.19.4 Arbiter如何运作

Arbiter卷有2个组件。一个是在每个第三个（即Arbiter）砖的brick过程中加载的仲裁器xlator。另一个是仲裁逻辑本身，它存在于客户端上加载的AFR（复制xlator）中。

前者作为FOPS的一种“过滤器”转换器 - 即它允许入口操作命中POSIX，阻止某些inode操作，如read（用ENOTCONN解除调用），并解除其他inode操作，如write，truncate等成功没有把它下到POSIX。

后者。即AFR中存在的仲裁逻辑执行以下操作：

写入文件时采用完整文件锁定，而不是正常复制卷中的范围锁定。这可以防止前面描述的3种方式的角落裂脑。

允许/失败写入FOPS以及client-quorum的Arbiter卷的行为可以在以下步骤中进行总结：

1）如果所有3块brick都已启动（快乐案例），则没有问题，并且允许FOP。

2）如果2块brick已经启动，并且其中一块是Arbiter（即第3块brick），并且 它会将另一块brick归咎于给定文件，那么所有写入FOPS都会因ENOTCONN而失败。这是因为，在这种情况下，唯一真正的副本位于关闭的brick上。因此，我们不能允许写入，直到该砖也开始。如果仲裁者没有责备其他brick，则允许FOPS继续进行。这里'责备'是AFR changelog扩展属性的值。

3）如果2个brick已启动并且Arbiter已关闭，则允许使用FOPS。当Arbiter出现时，条目/元数据会发生。当然不需要数据治愈。

4）如果只有一个brick，则不满足client-quorum，并且卷变为EROFS。

5）在所有情况下，如果在启动FOP之前只有一个源（即使所有brick都已启动），并且如果该源上的FOP失败，则应用程序将收到ENOTCONN。例如，假设在B2和B3上写入失败，即B1是唯一的源。现在如果由于某种原因，第二次写入在B1上失败（尽管所有brick都已启动之前有可能完成自我修复），应用程序将收到该写入的失败（ENOTCONN）。

上面描述的brick向上或向下并不一定意味着brick处理离线。这也可能意味着由于网络断开等原因，安装失去了与砖的连接。

## 4.20 强制锁

### 4.20.1 强制锁

对GlusterFS内部的强制锁定的支持本身并不会集中到Linux内核为用户空间文件系统提供的内容。在这里，我们使用和不使用文件模式位来强制执行核心强制锁定语义。请仔细阅读设计规范，该规范解释了为GlusterFS完成的强制锁实现背后的整个概念。

**含义和用法**

默认情况下，将禁用卷的强制锁定，并且可以使用卷集选项将卷配置为在3种不同的强制锁定模式下运行。

### 4.20.2 卷选项

gluster volume set <VOLNAME> locks.mandatory-locking <off / file / forced / optimal>

off - 禁用指定卷的强制锁定。

file - 借助模式位启用Linux内核样式强制锁定语义（未经过充分测试）

forced - 检查卷中每个数据修改操作的字节范围锁是否存在冲突

optimal - 组合模式，其中POSIX客户端可以使用其咨询锁定语义，这仍将遵循其他客户端（如SMB）获取的强制锁定。

注意： - 有关这些关键值的更多信息，请参阅设计文档。

**需要记住的要点：**

1）强制锁定卷集选项可用的有效键值仅在后续启动/重新启动卷后生效。

2）由于一些突出问题，建议关闭性能转换器，以便在上述任何一种强制锁定模式中配置卷时具有强制锁定的完整功能。有关详细信息，请参阅下面的“已知问题”部分。

**已知的问题**

由于强制锁的整个逻辑是在服务器端加载的锁转换器中实现的，因此早期成功返回到fops，如通过驻留在客户端的性能转换器打开，读取，写入上层/应用层将影响预期的功能。强制锁。在以下bugzilla报告中跟踪了一个这样的问题：

https://bugzilla.redhat.com/show\_bug.cgi?id=1194546

对于强制锁定和正在进行的读/写操作，可能存在未触发的竞赛窗口。有关详细信息，请参阅以下错误报告：

https://bugzilla.redhat.com/show\_bug.cgi?id=1287099

## 4.21 [GlusterFS coreutilities](https://docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20Coreutils/)

### 4.21.1 Coreutils for GlusterFS volumes

GlusterFS Coreutils是一套旨在模仿标准Linux coreutils的实用程序，除了它使用gluster C API才能完成工作。它提供了类似于ftp程序的界面。操作包括将文件从服务器提取到本地计算机，将文件从本地计算机放到服务器，从服务器检索目录信息等操作。

**安装GlusterFS**

有关GlusterFS的先决条件，说明和配置的信息，请参阅http://docs.gluster.org/en/latest/上的安装指南。

**安装glusterfs-coreutils**

目前glusterfs-coreutils将仅作为rpm打包。很快就会支持其他包格式。

**对于fedora**

使用dnf / yum安装glusterfs-coreutils：

dnf install glusterfs-coreutils

或者

yum install glusterfs-coreutils

### 4.21.2 用法

glusterfs-coreutils提供了一组基本实用程序，如cat，cp，flock，ls，mkdir，rm，stat和tail，它们是使用通常称为libgfapi的GlusterFS API专门实现的。这些实用程序既可以在gluster远程shell中使用，也可以作为独立命令使用，并将'gf'添加到各自的基本名称之前。例如，glusterfs cat实用程序被命名为gfcat，依此类推，但flock核心实用程序除外，因为没有提供独立的gfflock命令（请参阅有关为什么以这种方式设计flock的注释部分）。

**在远程gluster-shell中使用coreutils**

调用一个新shell

要进入gluster客户端shell，请键入gfcli并按Enter键。现在，您将看到类似的提示，如下所示：

gfcli>

有关更多选项，请参见gfcli的手册页。

**连接到gluster卷**

现在我们需要将客户端连接到已经启动的某些glusterfs卷。使用connect命令执行如下操作：

gfcli> connect glfs://<SERVER-IP or HOSTNAME>/<VOLNAME>

例如，如果在具有hostname localhost的服务器上有一个名为vol的卷，则上述命令将采用以下格式：

gfcli> connect glfs://localhost/vol

通过验证新提示，确保您已成功连接到远程gluster卷，该提示应如下所示：

gfcli (<SERVER IP or HOSTNAME/<VOLNAME>)

**尝试你最喜欢的实用程序**

请仔细阅读手册页，了解每个命令的不同实用程序和可用选项。例如，man gfcp将显示有关在gluster-shell之外或之内使用cp命令的详细信息。运行不同的命令如下：

gfcli (localhost/vol) ls .

gfcli (localhost/vol) stat .trashcan

**终止卷的客户端连接**

使用disconnect命令关闭连接：

gfcli (localhost/vol) disconnect

gfcli>

**退出shell**

gfcli> quit

**使用独立的glusterfs coreutil命令**

如上所述，glusterfs coreutils还提供独立命令来执行基本的GNU coreutil功能。所有这些命令都以'gf'为前缀。您可以直接利用这些来一次性建立和执行操作，而不是调用gluster客户端shell。例如，请参阅gfstat命令的以下示例用法：

gfstat glfs://localhost/vol/foo

有关flock coreutility的豁免，由于“注释”部分中描述的原因而无法作为独立命令提供。

有关每个命令和相应选项的更多信息，请参阅相关的手册页。

**Notes**

1）在gluster client-shell的特定会话中，保留了命令历史记录，即，您可以使用向上/向下箭头键来搜索先前执行的命令或使用Ctrl + R搜索反向历史搜索技术。

2）flock不能作为独立的'gfflock'使用。因为锁始终与文件描述符相关联。与其他所有命令不同，flock在获取锁之后无法立即清理文件描述符。对于flock，我们需要维护一个有效的连接作为glusterfs客户端。

## 4.22 [事件API](https://docs.gluster.org/en/latest/Administrator%20Guide/Events%20APIs/)

### 4.22.1 事件API

*3.9版中的新功能*

设置PYTHONPATH（仅在源安装的情况下）

如果使用源安装安装Gluster，cliutils/usr/local/lib/python.2.7/site-packages通过添加来安装在Set PYTHONPATH下~/.bashrc

export PYTHONPATH=/usr/local/lib/python2.7/site-packages:$PYTHONPATH

### 4.22.2 启用和启动事件API

在所有对等节点中启用并启动glustereventsd

在使用Systemd的系统中，

systemctl enable glustereventsd

systemctl start glustereventsd

FreeBSD或其他，添加以下内容 /etc/rc.conf

glustereventsd\_enable="YES"

并启动glustereventsd使用，

service glustereventsd start

SysVInit（CentOS 6）:

chkconfig glustereventsd on

service glustereventsd start

### 4.22.3 状态

状态可以查看可用以下命令：

gluster-eventsapi status

输出示例：

Webhooks:

None

+-----------+-------------+-----------------------+

| NODE | NODE STATUS | GLUSTEREVENTSD STATUS |

+-----------+-------------+-----------------------+

| localhost | UP | UP |

| node2 | UP | UP |

+-----------+-------------+-----------------------+

### 4.22.4 Webhooks

**Webhooks**类似于回调（通过HTTP），在事件Gluster将调用配置的Webhook URL（通过POST）。Webhook是一个侦听URL的Web服务器，可以在集群外部署。Gluster节点应该能够在配置的端口上访问此Webhook服务器。

用python编写的Webhook示例，

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route("/listen", methods=["POST"])

def events\_listener():

gluster\_event = request.json

if gluster\_event is None:

# No event to process, may be test call

return "OK"

# Process gluster\_event

# {

# "nodeid": NODEID,

# "ts": EVENT\_TIMESTAMP,

# "event": EVENT\_TYPE,

# "message": EVENT\_DATA

# }

print (gluster\_event)

return "OK"

app.run(host="0.0.0.0", port=9000)

使用以下命令测试和注册webhook，

usage: gluster-eventsapi webhook-test [-h] [--bearer\_token BEARER\_TOKEN] url

positional arguments:

url URL of Webhook

optional arguments:

-h, --help show this help message and exit

--bearer\_token BEARER\_TOKEN, -t BEARER\_TOKEN

Bearer Token

示例（Webhook服务器正在运行192.168.122.188:9000），

gluster-eventsapi webhook-test http://192.168.122.188:9000/listen

+-----------+-------------+----------------+

| NODE | NODE STATUS | WEBHOOK STATUS |

+-----------+-------------+----------------+

| localhost | UP | OK |

| node2 | UP | OK |

+-----------+-------------+----------------+

如果所有对等节点的Webhook状态都正常，则使用注册Webhook，

usage: gluster-eventsapi webhook-add [-h] [--bearer\_token BEARER\_TOKEN] url

positional arguments:

url URL of Webhook

optional arguments:

-h, --help show this help message and exit

--bearer\_token BEARER\_TOKEN, -t BEARER\_TOKEN

Bearer Token

例，

gluster-eventsapi webhook-add http://192.168.122.188:9000/listen

+-----------+-------------+-------------+

| NODE | NODE STATUS | SYNC STATUS |

+-----------+-------------+-------------+

| localhost | UP | OK |

| node2 | UP | OK |

+-----------+-------------+-------------+

**注意**：如果任何节点的“同步”状态为“不正常”，请确保在该节点启动时从对等节点运行以下命令

gluster-eventsapi sync

要取消订阅事件，请使用以下命令删除webhook

usage: gluster-eventsapi webhook-del [-h] url

positional arguments:

url URL of Webhook

optional arguments:

-h, --help show this help message and exit

例，

gluster-eventsapi webhook-del <http://192.168.122.188:9000/listen>

### 4.22.5 Configuration

查看所有配置，

usage: gluster-eventsapi config-get [-h] [--name NAME]

optional arguments:

-h, --help show this help message and exit

--name NAME Config Name

输出示例：

+-----------+-------+

| NAME | VALUE |

+-----------+-------+

| log\_level | INFO |

| port | 24009 |

+-----------+-------+

要更改任何配置，

usage: gluster-eventsapi config-set [-h] name value

positional arguments:

name Config Name

value Config Value

optional arguments:

-h, --help show this help message and exit

示例输出，

+-----------+-------------+-------------+

| NODE | NODE STATUS | SYNC STATUS |

+-----------+-------------+-------------+

| localhost | UP | OK |

| node2 | UP | OK |

+-----------+-------------+-------------+

要重置任何配置，

usage: gluster-eventsapi config-reset [-h] name

positional arguments:

name Config Name or all

optional arguments:

-h, --help show this help message and exit

示例输出，

+-----------+-------------+-------------+

| NODE | NODE STATUS | SYNC STATUS |

+-----------+-------------+-------------+

| localhost | UP | OK |

| node2 | UP | OK |

+-----------+-------------+-------------+

**注意**：如果任何节点状态未UP或同步状态不正常，请确保gluster-eventsapi sync从对等节点运行。

### 4.22.6 将节点添加到群集

**将新节点添加到群集时**

1）使用上述步骤在新节点中启用和启动事件

2）gluster-eventsapi sync从新节点以外的对等节点运行命令。

### 4.22.7 API文档

Glustereventsd将JSON格式的事件推送到已配置的Webhooks。所有活动都将具有以下属性。

| **属性** | **描述** |
| --- | --- |
| NODEID | 节点UUID |
| TS | 事件时间戳 |
| 事件 | 事件类型 |
| 信息 | 事件特定数据 |

例：

{

"nodeid": "95cd599c-5d87-43c1-8fba-b12821fd41b6",

"ts": 1468303352,

"event": "VOLUME\_CREATE",

"message": {

"name": "gv1"

}

}

“message”可以根据事件类型具有以下属性。

### 4.22.8 Events

#### 4.22.8.1 peer事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| PEER\_ATTACH | 主办 | 添加节点的主机名或IP |
| PEER\_DETACH | 主办 | 分离节点的主机名或IP |

#### 4.22.8.2 volume事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| VOLUME\_CREATE | 名称 | 卷名 |
| VOLUME\_START | 力 | 在启动期间使用或不使用强制选项 |
|  | 名称 | 卷名 |
| VOLUME\_STOP | 力 | 在停止期间使用或不使用强制选项 |
|  | 名称 | 卷名 |
| VOLUME\_DELETE | 名称 | 卷名 |
| VOLUME\_SET | 名称 | 卷名 |
|  | 选项 | 选项列表[（key1，val1），（key2，val2），..] |
| VOLUME\_RESET | 名称 | 卷名 |
|  | 选项 | 选项名称 |

#### 4.22.8.3 brick事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BRICK\_RESET\_START | 体积 | 卷名 |
|  | 来源砖 | Source Brick详情 |
| BRICK\_RESET\_COMMIT | 体积 | 卷名 |
|  | 目的地砖 | 目的地砖 |
|  | 来源砖 | Source Brick详情 |
| BRICK\_REPLACE | 体积 | 卷名 |
|  | 目的地砖 | 目的地砖 |
|  | 来源砖 | Source Brick详情 |

#### 4.22.8.4 Georep事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| GEOREP\_CREATE | 力 | 会话创建期间使用的强制选项 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
|  | no\_verify | 不使用验证选项 |
|  | push\_pem | 使用push pem选项或不使用 |
|  | ssh\_port | 如果在会话创建期间配置了SSH端口 |
|  | 主 | 主卷名 |
| GEOREP\_START | 力 | 会话开始时使用的强制选项 |
|  | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
| GEOREP\_STOP | 力 | 会话停止期间使用的强制选项 |
|  | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
| GEOREP\_PAUSE | 力 | 会话暂停期间使用的强制选项 |
|  | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
| GEOREP\_RESUME | 力 | 会话恢复期间使用的强制选项 |
|  | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
| GEOREP\_DELETE | 力 | 在会话删除期间使用强制选项 |
|  | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
| GEOREP\_CONFIG\_SET | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
|  | 选项 | Geo-rep配置的名称 |
|  | 值 | 改变了价值 |
| GEOREP\_CONFIG\_RESET | 主 | 主卷名 |
|  | 奴隶 | 奴隶细节（Slavehost :: SlaveVolume） |
|  | 选项 | Geo-rep配置的名称 |

#### 4.22.8.5 bitrot事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BITROT\_ENABLE | 名称 | 卷名 |
| BITROT\_DISABLE | 名称 | 卷名 |
| BITROT\_SCRUB\_THROTTLE | 名称 | 卷名 |
|  | 值 | 改变了价值 |
| BITROT\_SCRUB\_FREQ | 名称 | 卷名 |
|  | 值 | 改变了价值 |
| BITROT\_SCRUB\_OPTION | 名称 | 卷名 |
|  | 值 | 改变了价值 |

#### 4.22.8.6 quota事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| QUOTA\_ENABLE | 体积 | 卷名 |
| QUOTA\_DISABLE | 体积 | 卷名 |
| QUOTA\_SET\_USAGE\_LIMIT | 体积 | 卷名 |
|  | 路径 | 卷中的路径，其中设置了配额选项 |
|  | 限制 | 改变了价值 |
| QUOTA\_SET\_OBJECTS\_LIMIT | 体积 | 卷名 |
|  | 路径 | 卷中的路径，其中设置了配额选项 |
|  | 限制 | 改变了价值 |
| QUOTA\_REMOVE\_USAGE\_LIMIT | 体积 | 卷名 |
|  | 路径 | “配额”选项重置的卷中的路径 |
| QUOTA\_REMOVE\_OBJECTS\_LIMIT | 体积 | 卷名 |
|  | 路径 | “配额”选项重置的卷中的路径 |
| QUOTA\_ALERT\_TIME | 体积 | 卷名 |
|  | 时间 | 改变了警报时间 |
| QUOTA\_SOFT\_TIMEOUT | 体积 | 卷名 |
|  | 软超时 | 改变了价值 |
| QUOTA\_HARD\_TIMEOUT | 体积 | 卷名 |
|  | 硬超时 | 改变了价值 |
| QUOTA\_DEFAULT\_SOFT\_LIMIT | 体积 | 卷名 |
|  | 默认情况下，软限制 | 改变了价值 |

#### 4.22.8.7 snapshot事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| SNAPSHOT\_CREATED | snapshot\_name | 快照名称 |
|  | 卷名 | 卷名 |
|  | snapshot\_uuid | 快照UUID |
| SNAPSHOT\_CREATE\_FAILED | snapshot\_name | 快照名称 |
|  | 卷名 | 卷名 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_ACTIVATED | snapshot\_name | 快照名称 |
|  | snapshot\_uuid | 快照UUID |
| SNAPSHOT\_ACTIVATE\_FAILED | snapshot\_name | 快照名称 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_DEACTIVATED | snapshot\_name | 快照名称 |
|  | snapshot\_uuid | 快照UUID |
| SNAPSHOT\_DEACTIVATE\_FAILED | snapshot\_name | 快照名称 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_SOFT\_LIMIT\_REACHED | 卷名 | 卷名 |
|  | volume\_id | 卷ID |
| SNAPSHOT\_HARD\_LIMIT\_REACHED | 卷名 | 卷名 |
|  | volume\_id | 卷ID |
| SNAPSHOT\_RESTORED | snapshot\_name | 快照名称 |
|  | 卷名 | 卷名 |
|  | snapshot\_uuid | 快照UUID |
| SNAPSHOT\_RESTORE\_FAILED | snapshot\_name | 快照名称 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_DELETED | snapshot\_name | 快照名称 |
|  | snapshot\_uuid | 快照UUID |
| SNAPSHOT\_DELETE\_FAILED | snapshot\_name | 快照名称 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_CLONED | clone\_uuid | 快照克隆UUID |
|  | snapshot\_name | 快照名称 |
|  | clone\_name | 快照克隆名称 |
| SNAPSHOT\_CLONE\_FAILED | snapshot\_name | 快照名称 |
|  | clone\_name | 快照克隆名称 |
|  | 错误 | 失败的细节 |
| SNAPSHOT\_CONFIG\_UPDATED | 自动删除 | 自动删除值（如果可用） |
|  | config\_type | 卷配置或系统配置 |
|  | hard\_limit | 硬限制值（如果有） |
|  | soft\_limit | 软限制值（如果有） |
|  | 卡激活 | 快照激活值（如果可用） |
| SNAPSHOT\_CONFIG\_UPDATE\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_INITIALISED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_INIT\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_ENABLED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_ENABLE\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_DISABLED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_DISABLE\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_ADDED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_ADD\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_EDITED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_EDIT\_FAILED | 错误 | 错误详情 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_DELETED | 状态 | 成功状态 |
| SNAPSHOT\_SCHEDULER\_SCHEDULE\_DELETE\_FAILED | 错误 | 错误详情 |

#### 4.22.8.8 Svc事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| SVC\_MANAGER\_FAILED | 体积 | 卷名称（如果有） |
|  | svc\_name | 服务名称 |
| SVC\_CONNECTED | 体积 | 卷名称（如果有） |
|  | svc\_name | 服务名称 |
| SVC\_DISCONNECTED | svc\_name | 服务名称 |

#### 4.22.8.9 Peer events

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| PEER\_STORE\_FAILURE | 窥视 | 主机名或IP |
| PEER\_RPC\_CREATE\_FAILED | 窥视 | 主机名或IP |
| PEER\_REJECT | 窥视 | 主机名或IP |
| PEER\_CONNECT | 主办 | 主机名或IP |
|  | UUID | 主机UUID |
| PEER\_DISCONNECT | 主办 | 主机名或IP |
|  | UUID | 主机UUID |
|  | 州 | 断开状态 |
| PEER\_NOT\_FOUND | 窥视 | 主机名或IP |
|  | UUID | 主机UUID |

#### 4.22.8.10 未知事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| UNKNOWN\_PEER | 窥视 | 主机名或IP |

#### 4.22.8.11 brick事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BRICK\_START\_FAILED | 窥视 | 主机名或IP |
|  | 体积 | 卷名 |
|  | 砖 | 砖 |
| BRICK\_STOP\_FAILED | 窥视 | 主机名或IP |
|  | 体积 | 卷名 |
|  | 砖 | 砖 |
| BRICK\_DISCONNECTED | 窥视 | 主机名或IP |
|  | 体积 | 卷名 |
|  | 砖 | 砖 |
| BRICK\_CONNECTED | 窥视 | 主机名或IP |
|  | 体积 | 卷名 |
|  | 砖 | 砖 |

**bricks事件**

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BRICKS\_START\_FAILED | 体积 | 卷名 |

**Brickpath事件**

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BRICKPATH\_RESOLVE\_FAILED | 窥视 | 主机名或IP |
|  | 体积 | 卷名 |
|  | 砖 | 砖 |

#### 4.22.8.12 Notify事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| NOTIFY\_UNKNOWN\_OP | 运 | 操作名称 |

#### 4.22.8.13 Quorum事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| QUORUM\_LOST | 体积 | 卷名 |
| QUORUM\_REGAINED | 体积 | 卷名 |

#### 4.22.8.14 rebalance事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| REBALANCE\_START\_FAILED | 体积 | 卷名 |
| REBALANCE\_STATUS\_UPDATE\_FAILED | 体积 | 卷名 |

#### 4.22.8.15 import事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| IMPORT\_QUOTA\_CONF\_FAILED | 体积 | 卷名 |
| IMPORT\_VOLUME\_FAILED | 体积 | 卷名 |
| IMPORT\_BRICK\_FAILED | 窥视 | 主机名或IP |
|  | 砖 | 砖细节 |

#### 4.22.8.16 compare事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| COMPARE\_FRIEND\_VOLUME\_FAILED | 体积 | 卷名 |

#### 4.22.8.17 Ec事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| EC\_MIN\_BRICKS\_NOT\_UP | subvol | 子卷 |
| EC\_MIN\_BRICKS\_UP | subvol | 子卷 |

#### 4.22.8.18 Georep事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| GEOREP\_FAULTY | master\_node | 主卷的主机名或IP |
|  | brick\_path | 砖路径 |
|  | slave\_host | 从属主机名或IP |
|  | 主音量 | 主卷名 |
|  | current\_slave\_host | Geo-rep工作者尝试连接的当前Slave Host |
|  | slave\_volume | 从属卷名称 |

#### 4.22.8.19 quota事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| QUOTA\_CROSSED\_SOFT\_LIMIT | 用法 | 用法 |
|  | 体积 | 卷名 |
|  | 路径 | 路径 |

#### 4.22.8.20 bitrot事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| BITROT\_BAD\_FILE | GFID | 文件的GFID |
|  | 路径 | 路径如果可用 |
|  | 砖 | 砖细节 |

#### 4.22.8.21 Client事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| CLIENT\_CONNECT | CLIENT\_IDENTIFIER | 客户标识符 |
|  | client\_uid | 客户端UID |
|  | server\_identifier | 服务器标识符 |
|  | brick\_path | 砖的路径 |
| CLIENT\_AUTH\_REJECT | CLIENT\_IDENTIFIER | 客户标识符 |
|  | client\_uid | 客户端UID |
|  | server\_identifier | 服务器标识符 |
|  | brick\_path | 砖的路径 |
| CLIENT\_DISCONNECT | CLIENT\_IDENTIFIER | 客户标识符 |
|  | client\_uid | 客户端UID |
|  | server\_identifier | 服务器标识符 |
|  | brick\_path | 砖的路径 |

#### 4.22.8.22 Posix事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| POSIX\_SAME\_GFID | GFID | 文件的GFID |
|  | 路径 | 文件的路径 |
|  | NEWPATH | 文件的新路径 |
|  | 砖 | 砖细节 |
| POSIX\_ALREADY\_PART\_OF\_VOLUME | 卷的标识 | 卷ID |
|  | 砖 | 砖细节 |
| POSIX\_BRICK\_NOT\_IN\_VOLUME | 砖 | 砖细节 |
| POSIX\_BRICK\_VERIFICATION\_FAILED | 砖 | 砖细节 |
| POSIX\_ACL\_NOT\_SUPPORTED | 砖 | 砖细节 |
| POSIX\_HEALTH\_CHECK\_FAILED | 路径 | 路径 |
|  | 砖 | 砖细节 |
|  | 运 | 错误号码 |
|  | 错误 | 错误 |

#### 4.22.8.23 Afr事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| AFR\_QUORUM\_MET | subvol | 子卷名称 |
| AFR\_QUORUM\_FAIL | subvol | 子卷名称 |
| AFR\_SUBVOL\_UP | subvol | 子卷名称 |
| AFR\_SUBVOLS\_DOWN | subvol | 子卷名称 |
| AFR\_SPLIT\_BRAIN | subvol | 子卷名称 |

#### 4.22.8.24 Tier事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| TIER\_ATTACH | 第一卷 | 卷名 |
| TIER\_ATTACH\_FORCE | 第一卷 | 卷名 |
| TIER\_DETACH\_START | 第一卷 | 卷名 |
| TIER\_DETACH\_STOP | 第一卷 | 卷名 |
| TIER\_DETACH\_COMMIT | 第一卷 | 卷名 |
| TIER\_DETACH\_FORCE | 第一卷 | 卷名 |
| TIER\_PAUSE | 第一卷 | 卷名 |
| TIER\_RESUME | 第一卷 | 卷名 |
| TIER\_WATERMARK\_HI | 第一卷 | 卷名 |
| TIER\_WATERMARK\_DROPPED\_TO\_MID | 第一卷 | 卷名 |
| TIER\_WATERMARK\_RAISED\_TO\_MID | 第一卷 | 卷名 |
| TIER\_WATERMARK\_DROPPED\_TO\_LOW | 第一卷 | 卷名 |

#### 4.22.8.25 volume事件

| **事件类型** | **属性** | **描述** |
| --- | --- | --- |
| VOLUME\_ADD\_BRICK | 体积 | 卷名 |
|  | brick | 由Space分隔的砖细节 |
| VOLUME\_REMOVE\_BRICK\_START | 体积 | 卷名 |
|  | brick | 由Space分隔的砖细节 |
| VOLUME\_REMOVE\_BRICK\_COMMIT | 体积 | 卷名 |
|  | brick | 由Space分隔的砖细节 |
| VOLUME\_REMOVE\_BRICK\_STOP | 体积 | 卷名 |
|  | brick | 由Space分隔的砖细节 |
| VOLUME\_REMOVE\_BRICK\_FORCE | 体积 | 卷名 |
|  | brick | 由Space分隔的砖细节 |
| VOLUME\_REBALANCE\_START | 体积 | 卷名 |
| VOLUME\_REBALANCE\_STOP | 体积 | 卷名 |
| VOLUME\_REBALANCE\_FAILED | 体积 | 卷名 |
| VOLUME\_REBALANCE\_COMPLETE | 体积 | 卷名 |

## 4.23 [用基于Debian的系统的gfapi构建QEMU](https://docs.gluster.org/en/latest/Administrator%20Guide/Building%20QEMU%20With%20gfapi%20For%20Debian%20Based%20Systems/)

这个方法已在Ubuntu 13.10上在干净的最新的环境中进行了测试。如果我记得正确的话，较旧的Ubuntu发行版需要一些黑客攻击。其他基于Debian的发行版应该能够遵循这种依赖性调整。如果你让它在另一个发行版上工作，请更新这个。

**满足依赖性**

首先尝试获取qemu依赖项

apt-get build-dep qemu

下一个命令从上游Debian sid中获取debian控制文件中指定的所有依赖项。您可以查看其中指定的选项并调整到品味。

# get almost all the rest and the tools to work up the Debian magic

apt-get install devscripts quilt libiscsi-dev libusbredirparser-dev libssh2-1-dev libvdeplug-dev libjpeg-dev glusterfs\*

我们需要一个更新版本的libseccomp用于Ubuntu 13.10

mkdir libseccomp

cd libseccomp

# grab it from upstream sid

wget http://ftp.de.debian.org/debian/pool/main/libs/libseccomp/libseccomp\_2.1.0+dfsg.orig.tar.gz

wget http://ftp.de.debian.org/debian/pool/main/libs/libseccomp/libseccomp\_2.1.0+dfsg-1.debian.tar.gz

# get it ready

tar xf libseccomp\_2.1.0+dfsg.orig.tar.gz

cd libseccomp-2.1.0+dfsg/

# install the debian magic

tar xf ../libseccomp\_2.1.0+dfsg-1.debian.tar.gz

# apply series files if any

while quilt push; do quilt refresh; done

# build debs, they'll appear one directory up

debuild -i -us -uc -b

cd ..

# install it

dpkg -i \*.deb

**Building QEMU**

如果符合您的依赖关系，那么下一部分就很简单。对于高级读者，一旦在安装之前解压缩debian / control，您可能想要更改QEMU构建的选项以及请求的目标。

cd ..

mkdir qemu

cd qemu

# download our sources. you'll want to check back frequently on these for changes

wget http://ftp.de.debian.org/debian/pool/main/q/qemu/qemu\_1.7.0+dfsg.orig.tar.xz

wget http://ftp.de.debian.org/debian/pool/main/q/qemu/qemu\_1.7.0+dfsg-2.debian.tar.gz

wget http://download.gluster.org/pub/gluster/glusterfs/3.4/LATEST/glusterfs-3.4.2.tar.gz

tar xf glusterfs-3.4.2.tar.gz

tar xf qemu\_1.7.0+dfsg.orig.tar.xz

cd qemu-1.7.0+dfsg/

# unpack the debian magic

tar xf ../qemu\_1.7.0+dfsg-2.debian.tar.gz

# bring glusterfs in to the buiild

cp -r ../glusterfs-3.4.2 glusterfs

# the glusterfs check in configure looks around weird. I've never asked why but moving the src stuff up one works and tests fine

cd glusterfs/api/

mv src/\* .

cd ../..

#you'll need to edit debian/control to enable glusterfs replacing

- ##--enable-glusterfs todo

+ # --enable-glusterfs

+ glusterfs-common (>= 3.4.0),

#And finally build. It'll take ages. http://xkcd.com/303/

# apply series if any

while quilt push; do quilt refresh; done

# build packages

debuild -i -us -uc -b

cd ..

您的deb现在可以安装了。由读者决定他们想要安装哪些目标。

## 4.24 附录（Appendices）

### 4.24.1 Network configuration Techniques

#### 4.24.1.1 Bonding best practices

绑定网络接口将多个物理接口合并到一个逻辑绑定接口中，并具有单个IP地址。N路绑定接口可以承受N-1物理接口的丢失，并且在某些情况下可以改善性能。

**When to bond**

1）需要高可用性的网络链接

2）工作负载：对大文件的顺序访问（大部分时间用于读/写）

3）客户端/服务器的网络吞吐量限制\ <\ <存储吞吐量限制

1 GbE（几乎总是）

10-Gbps链路或更快 - 对于写入，复制使网络上的负载加倍，并且副本通常位于客户端可以并行传输的不同对等端上。

4）限制：如果网络对等体不在同一VLAN上，则绑定模式6不会提高吞吐量。

**How to configure**

1）bonding-howto

2）Gluster客户端的最佳绑定模式是模式6（balance-alb），这允许客户端在大多数时间内在不同的NIC上并行传输写入。在具有巨型帧的2个10-GbE NIC上，使用绑定模式6观察到来自单个客户端的写入峰值吞吐量为750 MB / s。那是1.5 GB / s的网络流量。

3）另一种平衡发送和接收流量的方法是绑定模式4（802.3ad），但这需要交换机配置（中继命令）

4）另一种加载平衡的方法是绑定模式2（balance-xor）和选项“xmit\_hash\_policy = layer3 + 4”。绑定模式6和2不会改善单连接吞吐量，但会提高所有连接的聚合吞吐量。

#### 4.24.1.2 巨型帧-Jumbo frames

巨型帧是以太网（或Infiniband）帧，其大小大于默认值1500字节（Infiniband默认值约为2000字节）。增加帧大小可减少操作系统和硬件的负载，操作系统和硬件必须每帧处理中断和协议消息。

**When to configure**

1）任何网络都比1-GbE快

2）工作负载是顺序大文件读/写

3）限制：要求VLAN中的所有网络交换机必须配置为处理巨型帧，否则不进行配置。

**How to configure**

1）编辑/ etc / sysconfig / network-scripts / ifcfg-your-interface中的网络接口文件

2）以太网（在ixgbe驱动程序上）：将“MTU = 9000”（MTU表示“最大传输单元”）记录添加到网络接口文件

3）Infiniband（在mlx4驱动程序上）：将“CONNECTED\_MODE = yes”和“MTU = 65520”记录添加到网络接口文件

4）ifdownyour-interface; ifup your-interface

5）使用“ping -s 16384 other-host-on-VLAN”进行测试

6）由于协议头，交换机要求最大帧大小大于MTU，通常为9216字节

#### 4.24.1.3 配置后端网络以进行存储

此方法允许您通过在不同网络接口上隔离不同协议的流量来为多协议站点添加网络容量。此方法可以降低延迟并提高吞吐量。例如，此方法可以保持自我修复和重新平衡流量，使其不与网络接口的非Gluster客户端流量竞争，并且将更好地支持多流I / O.

何时配置

对于非Gluster服务，例如NFS，Swift（REST），Gluster服务器上提供了CIFS。它不会帮助Gluster客户端（外部节点上有Gluster挂载点）。

网络端口被过度使用。

如何配置

大多数网卡都有多个端口 - 将端口1设置为非Gluster端口，将端口2设置为Gluster端口。

将Gluster端口与非Gluster端口分离到单独的VLAN，以简化配置。

### 4.24.2 性能测试

#### 4.24.2.1 Gluster性能测试

创建Gluster卷后，您需要验证它是否具有适合您应用程序的性能，如果没有，则需要一种方法来隔离问题的根本原因。

有两种工作负载：

synthetic - 运行下面的测试程序

application - 运行现有应用程序

#### 4.24.2.2 分析工具

理想情况下，最好使用您希望在Gluster上运行的实际应用程序，但应用程序通常不会告诉系统管理员有关性能问题的位置，特别是延迟（响应时间）问题。因此，Gluster中内置了非侵入性分析工具，可以在不更改应用程序的情况下测量应用程序所看到的性能。Gluster分析方法目前基于io-stats转换器，包括：

1、客户端分析 - 检测Gluster mountpoint或libgfapi进程以对分析数据进行采样。在这种情况下，io-stats转换器位于转换器堆栈的“顶部”，因此配置文件数据真正代表应用程序（或FUSE挂载点）要求Gluster执行的操作。例如，单个应用程序写入作为WRITE FOP（文件操作）调用计数一次，并且该WRITE FOP的延迟包括AFR转换器在堆栈中较低的数据复制的延迟。

2、服务器端分析 - 这是使用“gluster volume profile”命令完成的（并且“gluster volume top”也可用于标识正在使用的特定热文件）。服务器端分析可以测量整个Gluster卷随时间的吞吐量，并可以测量服务器端延迟。但是，它不包含网络或客户端延迟。由于客户端转换器改变了I / O工作负载，因此很难推断应用程序行为（例如：擦除编码，缓存分层）。

简而言之，使用客户端分析来理解“为什么我的应用程序没有响应”？并使用服务器端分析来了解Gluster卷的繁忙程度，对它应用的工作负载类型（即它是否主要读取？是小文件吗？），以及I / O负载的传播程度如何整个卷。

**客户端分析**

要运行客户端分析，

gluster volume profile your-volume start

setfattr -n trusted.io-stats-dump -v /tmp/io-stats-pre.txt / your / mountpoint

这将在客户端上生成指定的文件。像gvp-client.sh这样的脚本 可以自动收集这些数据。

**服务器端分析**

要运行它：

gluster volume profile your-volume start

定期重复此命令：gluster volume profile your-volume info

gluster volume profile your-volume start

像gvp.sh这样的脚本可以帮助您自动执行此过程。

现在正在开发用于对这些数据进行后处理的脚本，让我们知道您需要什么以及用于呈现数据的有用格式。

#### 4.24.2.3 测试工具

在本节中，我们建议一些基本的工作负载测试，可用于以与应用程序无关的方式测量Gluster性能，适用于各种类似POSIX的操作系统和运行时环境。然后，我们提供一些术语和概念框架来解释这些结果。

我们在这里建议的工具旨在在分布式文件系统中运行。即使是现在，这仍然是文件系统基准测试的一个相对罕见的属性！可以从单个系统运行更大的基准测试集。虽然单系统结果很重要，但它们远远不能确定分布式文件系统的性能。

fio - 用于大文件I / O测试。

smallfile - 用于纯工作负载小文件测试

iozone - 用于纯工作负载大文件测试

parallel-libgfapi - 用于纯工作负载libgfapi测试

SPECsfs2014的“netmist”混合工作负载生成器在某些情况下可能是合适的，但在技术上不是开源工具。这个工具是由Don Capps编写的，他是iozone的作者。

**FIO**

fio非常强大，可以从传统的发行版中轻松安装，与iozone不同，并且具有越来越强大的分布式测试功能，在其截至2015年5月的上游--client参数中描述。要使用此模式，首先启动fio“服务器”实例每个工作负载生成器使用：

fio --server --daemonize=/var/run/fio-svr.pid

并确保您的防火墙允许端口8765通过它。您现在可以使用以下语法在主机集上运行测试：

fio --client=workload-generator.list --output-format=json my-workload.fiojob

您也可以将它用于分布式测试，方法是在不同的主机上启动fio实例，注意尽可能接近同一时间启动所有fio实例，限制每线程吞吐量，并指定运行持续时间而不是数量数据，以便所有fio实例大约在同一时间结束。然后，您可以聚合来自不同主机的fio结果，以获得有意义的聚合结果。

fio也有不同的I / O引擎，特别是Huamin Chen 为fio创建了libgfapi引擎，这样你就可以使用fio来测试Gluster性能，而无需使用FUSE。

分布式模式下fio的局限性：

1）stonewalling - fio根据最后一个线程完成测试运行的时间计算吞吐量。相反，iozone默认根据FIRST线程完成工作负载的时间计算吞吐量。这可能会导致iozone的吞吐量（欺骗性地）更高的吞吐量，因为不可避免地会有一些“落后”线程比其他线程更跛行到终点线。在某些情况下，可以通过指定测试的时间限制来克服此限制。这适用于随机I / O测试，通常您不希望读/写整个文件/设备。

2）当响应时间> 1秒时不准确 - 至少在某些情况下，当fio线程遇到响应时间远大于1秒时，fio报告的IOPS过高，当实现中存在不公平时，这可能发生在分布式存储中。

3）io引擎没有集成。

**smallfile分布式I / O基准测试**

Smallfile是一个基于python的小文件分布式POSIX工作负载生成器，可用于快速测量整个集群中各种元数据密集型工作负载的性能。它不依赖于任何特定的文件系统或实现AFAIK。它运行在Linux，Windows上，也适用于大多数Unix。它旨在补充使用iozone基准测试来测量大文件工作负载的性能，并借用iozone和Ric Wheeler的fs\_mark中的某些概念。它是由Ben England于2009年3月开发的，现在是开源的（Apache License v2）。

这是一个典型的简单测试序列，然后在后续测试中使用初始创建测试中的文件。还有许多小文件操作类型比这5个（参见doc），但这些是最常用的。

SMF="./smallfile\_cli.py --top /mnt/glusterfs/smf --host-set h1,h2,h3,h4 --threads 8 --file-size 4 --files 10000 --response-times Y "

$SMF --operation create

for s in $SERVERS ; do ssh $h 'echo 3 > /proc/sys/vm/drop\_caches' ; done

$SMF --operation read

$SMF --operation append

$SMF --operation rename

$SMF --operation delete

**IOZONE**

此工具有局限性，但使用 - + m选项（下面）进行分布式测试。

不建议使用“-a”选项自动测试所有用例，因为：

1）这不允许您在测试之前删除服务器中的读取缓存。

2）测量的大多数数据点与您正在解决的问题无关。

单线程测试是一个重要的用例，但为了充分利用可用的硬件，您通常需要进行多线程甚至多主机测试。

考虑使用“-c -e”选项来衡量数据到达持久存储所需的时间。“-C”选项可让您查看每个线程参与测试的程度。“ - + n”允许您通过跳过重新读取和重写测试来节省时间。“-w”选项告诉iozone不要删除它访问的任何文件，以便后续测试可以使用它们。为每个测试指定以下选项：

-i - 测试类型，0 =写入，1 =读取，2 =随机读取/写入

-r - 数据传输大小 - 允许您模拟应用程序使用的I / O大小

-s - 每线程文件大小 - 选择此值足以使系统达到稳定状态（通常需要多GB）

-t - 线程数 - 将同时发出I / O请求的子进程数

-F - 文件列表 - 要写入/读取的文件。如果未指定，则将在默认目录中使用文件名iozone.DUMMY。\*。

对于共享Gluster挂载点目录/ mnt / glusterfs，传输大小为64 KB，文件大小为1 GB的8线程顺序写入测试示例，包括fsync（）和close（）吞吐量计算中文件的时间：

iozone -w -c -e -i 0 -+n -C -r 64k -s 1g -t 8 -F /mnt/glusterfs/f{0,1,2,3,4,5,6,7,8}.ioz

警告：iozone中的随机I / O测试受到iozone约束的限制，它必须随机读取然后随机写入整个文件！这不是我们想要的 - 相反，它应该随机读/写一小部分文件大小或持续时间，允许我们在磁盘上分散更多，而不是等待太长时间才能完成测试。这就是为什么fio（下面）是随机I / O工作负载的首选测试工具。

分布式测试是iozone实用程序的优势，但这需要使用“ - + m”选项代替“-F”选项。使用“ - + m”选项传递的配置文件包含一系列如下所示的记录：

hostname directory iozone-pathname

其中hostname是iozone可以使用的测试驱动程序计算机的主机名或IP地址，directory是在该主机中使用的目录的路径名，而iozone-pathname是要在该主机上使用的iozone可执行文件的完整路径名。确保每个目标主机都可以解析运行iozone命令的主机的主机名。所有目标主机必须允许来自运行该命令的主机的无密码ssh访问。

例如:(这里，my-ip-address指的是运行iozone的机器）

export RSH=ssh

iozone -+m ioz.cfg -+h my-ip-address -w -c -e -i 0 -+n -C -r 64k -s 1g -t 4

并且文件ioz.cfg包含这些记录（其中/ mnt / glusterfs是每个测试机器上的Gluster挂载点，test-client-ip是客户端的IP地址）。另请注意，文件中的每条记录都是IOZone术语中的一个线程。由于我们在上面的示例中将线程数定义为4，因此我们为单个客户端提供了四个记录（线程）。

test-client-ip /mnt/glusterfs /usr/local/bin/iozone

test-client-ip /mnt/glusterfs /usr/local/bin/iozone

test-client-ip /mnt/glusterfs /usr/local/bin/iozone

test-client-ip /mnt/glusterfs /usr/local/bin/iozone

限制：由于iozone使用非特权端口，因此可能需要暂时关闭或更改部分/全部主机上的iptables。从机必须通过ssh支持来自主机的无密码访问。

请注意， - + h选项未记录，但它告诉从属主机使用哪个IP地址，以便从属设备无需解析测试驱动程序的主机名。my-ip-address是从属服务器应连接的IP地址，以便将结果报告给主机。这不必与主机的主机名相同。

通常，首先运行顺序写入测试以放置文件，在服务器上删除缓存（如果需要，还可以放置客户端），执行顺序读取测试，删除缓存，如果需要，执行随机I / O测试。使用以上示例：

export RSH=ssh

IOZ="iozone -+m ioz.cfg -+h my-ip-address -w -C -c -e -r 64k -+n "

hosts="`awk '{ print $1 }' ioz.cfg`"

$IOZ -i 0 -s 1g -t 4`\

for n in $hosts $servers ; do \

ssh $n 'sync; echo 1 > /proc/sys/vm/drop\_caches' ; done

$IOZ -i 1 -s 1g -t 4

for n in $hosts $servers ; do \

ssh $n 'sync; echo 1 > /proc/sys/vm/drop\_caches' ; done

$IOZ -i 2 -s 1g -t 4

如果您使用带有缓冲I / O的客户端（默认），请先在客户端计算机上删除缓存，然后服务器计算机也如上所示。

**Parallel-libgfapi**

此测试使用libgfapi API执行Gluster性能，绕过FUSE - 不使用挂载点。

要使用它，可以在parallel\_gfapi\_test.sh脚本中编辑脚本参数 - 所有这些都在注释“此行以下没有可编辑的参数”之上。其中包括Gluster卷名，为该卷提供服务的主机，文件数等等。然后确保将gfapi\_perf\_test可执行文件分发到指定目录中的客户端计算机，然后运行该脚本。该脚本以并行方式启动所有libgfapi工作负载生成器进程，以便它们同时启动测试。它等待它们全部完成，然后它为您收集和汇总结果。

请注意，libgfapi进程每块砖消耗一个套接字，因此在具有高砖数的Gluster卷中，可以对可以并发运行的libgfapi进程数量进行约束。具体来说，每个主机最多只能支持30000个并发TCP端口。您可能需要调整“ulimit -n”参数（请参阅/etc/security/limits.conf“nofile”参数以进行持久调整）。

**对象存储工具**

COSBench 由英特尔员工开发，对Swift和S3工作负载生成非常有用。

ssbench是OpenStack Swift工具集的一部分，是具有工作负载定义文件格式的命令行工具。

#### 4.24.2.4 工作负载-workload

应用程序可以像编写某些文件一样简单，也可以像在Gluster上运行云一样复杂。但是，所有应用程序都有性能要求，无论用户是否了解它们，如果不满足这些要求，从用户的角度来看，整个系统无法正常运行。应用程序花费大部分时间用于Gluster的活动在下面称为“工作负载”。对于Gluster文件系统，“工作负载”包括应用程序传递给Gluster的文件系统请求。有两种方法可以查看工作负载：

自上而下 - 尝试让文件系统执行的应用程序是什么？

自下而上 - 应用程序实际生成到文件系统的请求是什么？

**数据与元数据**

在此页面中，我们经常提到“大文件”或“小文件”工作负载。但是，“大文件”或“小文件”这两个术语是什么意思？“large-file”是一个故意模糊但描述性的术语，指的是大部分应用程序时间花在读取/写入文件上的工作负载。这与“小文件”工作负载形成对比，其中大部分应用程序的时间用于打开/关闭文件或访问有关文件的元数据。元数据表示“有关数据的数据”，因此它是描述文件状态的信息，而不是文件的内容。例如，文件名是一种元数据，目录和扩展属性也是如此。

**自上而下的工作负载分析**

通常这是用户可以帮助您的 - 例如，工作负载可能包括摄取十亿.mp3文件。需要回答的典型问题（大约）是：

1）什么是文件大小分布？平均值通常不够 - 文件大小分布可以是双峰的（即主要由非常大和非常小的文件大小组成）。TBS：提供指向可以收集此内容的脚本的指针。

2）文件访问的几分之一是读取与写入？

3）工作负载对缓存的友好程度如何？不同的Gluster客户端或这些客户端上的不同进程/线程是否重复读取相同的文件？

4）对于大文件工作负载，有多少次访问是顺序/随机的？顺序文件访问意味着应用程序线程以字节偏移顺序从头到尾读/写文件，随机文件访问正好相反 - 线程可以随时从任何偏移读/写。虚拟机磁盘映像通常是随机访问的，因为VM的文件系统嵌入在Gluster文件中。

为什么这些问题很重要？例如，如果您有大文件顺序读取工作负载，则网络配置+ Gluster和Linux预读很重要。如果您有小文件工作负载，则存储配置很重要，依此类推。除非您对工作负载有基本的了解，否则您不会知道哪种调优适合Gluster。

**自下而上的分析**

从文件系统为其请求提供服务的角度来看，即使是复杂的应用程序也可能具有非常简单的工作负载。如果您不知道应用程序花费的时间，可以先运行“gluster volume profile”和“gluster volume top”命令。这些非常有用的工具将帮助您了解限制该工作负载性能的工作负载和瓶颈。

TBS：链接到这些工具和脚本的文档，可将数据减少为可用的形式。

#### 4.24.2.5 配置

Gluster服务器有4个基本硬件维度，按重要性顺序列出：

network - 可能是Gluster网站最重要的硬件组件

access protocol - 使用什么样的客户端来访问文件/对象？

storage - 这对于提前做好至关重要

cpu - 在客户端上，寻找热线程（见下文）

memory - 可能会影响读取密集型，可缓存工作负载的性能

**网络测试**

网络配置对分布式存储的性能有巨大影响，但在群集生命周期的规划和安装阶段通常没有得到应有的重视。幸运的是， 网络配置 可以显着增强，通常无需额外的硬件。

要测量网络性能，请考虑使用 基于netperf的 脚本（例如 network-stream-pairs.sh） - 在主机对之间设置单向TCP流network-rpc-pairs.sh - 在主机对之间 设置请求 - 响应流

这两个工具的目的是表征整个网络基础架构的容量，以支持并行使用多个网络连接的分布式存储引起的所需流量级别。后一个脚本可能是分布式存储最现实的网络工作负载。

影响分布式存储的两个最常见的硬件问题毫不奇怪地是磁盘驱动器故障和网络故障。其中一些故障不会导致硬错误，反而会导致性能下降。例如，对于包含两个物理网络接口的绑定网络接口，如果其中一个物理接口发生故障（NIC /交换机上的端口或电缆），则绑定接口将保持运行，但性能会降低（少得多）取决于粘合模式）。另一个错误是10-GbE以太网接口无法自动协商速度达到10-Gbps - 有时网络接口自动协商为1-Gbps。如果TCP连接遇到高丢包率或未正确调整，则可能无法达到硬件支持的完整网络速度。

那么为什么要运行并行netperf会话而不只是一个呢？存在与网络拓扑（主机互连方式）相关的各种网络性能问题，特别是网络交换机和路由器拓扑，只有在几对主机试图通过同一共享资源传输流量时才会显示，这可能是例如，是连接架顶式交换机或基于刀片的交换机的中继线，其带宽不足以切换背板。单独的netperf / iperf会话不会发现这些问题，但是这个脚本会。

例如，该测试可用于模拟通过分布式文件系统的数据流。如果要模拟4个Gluster客户端，请将它们称为c1到c4，将大文件写入一组2个服务器，将它们称为s1和s2，您可以指定这些（发送方，接收方）对：

(c1,s1), (c2, s2), (c3, s1), (c4, s2)

另一方面，如果你想模拟读取，你可以使用这些（发送者，接收者）对：

(s1, c1), (s2, c2), (s1, c3), (s2, c4)

要模拟混合读写工作负载，请使用两组对：

(c1,s1), (c2, s2), (c3, s1), (c4, s2), (s1, c1), (s2, c2), (s1, c3), (s2, c4)

更复杂的流可以模拟非本机协议的行为，其中集群节点充当代理服务器 - 它是服务器（用于非本机协议）和客户端（用于本机协议）。例如，这种协议通常会引起全双工流量，这可能使网络不同于单向输入/输出流量。例如，尝试将此组流添加到前一个流：

(s1, s2),.(s2, s3),.(s3, s4),.(s4, s1)

脚本顶部的注释描述了输入语法，但这里有一些关于如何最好地利用它的建议。您通常从头节点或测试驱动程序运行此脚本，该驱动程序对正在测试的计算机集具有无密码ssh访问权限。运行测试的主机不需要相互之间的ssh访问 - 它们只需要允许来自头节点的无密码ssh访问。该脚本不依赖于root权限，因此您可以从非root帐户运行它。只需在右侧帐户的头节点上创建一个公钥（通常在\ $ HOME / .ssh / id\_rsa.pub中），然后将此公钥附加到参与测试的每个主机上的\ $ HOME / .ssh / authorized\_keys。

我们使用单独的文本文件输入发送者和接收者，每行1个主机。对于pair（发送者[j]，receiver [j]），你从发送者文件中的第j行获得发送者[j]，从接收者文件中的第j行获得接收者[j]。您必须使用与要测试的接口对应的IP地址/名称，并且必须能够使用此接口从头节点ssh到每个主机。

**结果**

性能结果有三种基本形式，不是按重要性排序：

1）throughput - 在一个单位时间内完成了多少工作？最佳指标通常取决于工作负载：

对于大文件随机：IOPS

对于大文件顺序：MB / s

对于小文件：files / sec

2）response time - 重要的是，文件系统请求需要多长时间才能完成？

3）utilization - 工作负载运行时硬件有多忙？

4）scalability - 当我们将服务器添加到Gluster卷时，我们能否在不牺牲响应时间的情况下线性扩展吞吐量？

通常吞吐量结果最受关注，但在分布式存储环境中，实现最难的目标可能是响应时间一直很短，而不是吞吐量。

虽然存在响应时间无关紧要的非交互式工作负载，但在用户必须直接与文件系统交互的任何情况下都应注意响应时间。调整文件系统以实现绝对最高吞吐量可能导致文件系统因高响应时间而无法使用。除非您处于基准测试状态，否则您希望实现良好的吞吐量和响应时间的平衡。通常，交互式用户希望始终在5秒内看到响应时间，大多数响应时间远低于此。为了控制响应时间（包括系统管理！），您不希望任何硬件组件以最大利用率运行，通常60-80％的利用率是一个很好的峰值利用率目标。另一方面，为避免浪费硬件，