# 冷存储--gluster集群设计

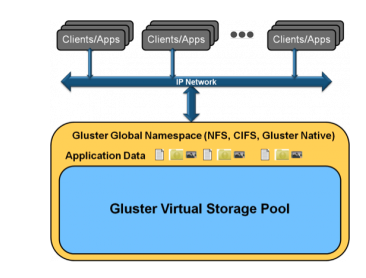
### （一）集群设计说明

|  |  |  |
| --- | --- | --- |
| 软件环境： | client：Centos 7.6.1810、kernel: 3.10.0-957;  Server: ubuntu 14.04.4、kernel：4.4.8-armada  GlusterFS：3.13.2（稳定版，可回退3.12） | |
| 集群配置及应用建议： | 集群规模\*单节点磁盘数\*挂载客户端数=32\*6\*4；集群目录深度\*单目录文件=10\*50万（建议上限值）；不推荐存储1M以下小文件 | |
| Glusterfs自身功能 | 配额 | √ |
| 集群告警 | √ |
| 数据恢复 | √ |
| 对接CIFS/NFS/FTP | √ |
| 用户审计 | × |
| 集群性能（带宽，iops） | √ |
| 存储池状态（带宽，iops，数据重构速度，休眠或唤醒状态及持续时间） | √ |

### （二）gluster原理与框架

#### 1.gluster概述

GlusterFS是Scale-Out存储解决方案Gluster的核心，它是一个开源的分布式文件系统，具有强大的横向扩展能力，通过扩展能够支持数PB存储容量和处理数千客户端。GlusterFS借助TCP/IP或InfiniBand RDMA网络将物理分布的存储资源聚集在一起，使用单一全局命名空间来管理数据。GlusterFS基于可堆叠的用户空间设计，可为各种不同的数据负载提供优异的性能。



#### gluster特性

##### （1）弹性存储系统（Elasticity）

存储系统具有弹性能力，意味着企业可以根据业务需要灵活地增加或缩减数据存储以及增删存储池中的资源，而不需要中断系统运行。GlusterFS设计目标之一就是弹性，允许动态增删数据卷、扩展或缩减数据卷、增删存储服务器等，不影响系统正常运行和业务服务。GlusterFS早期版本中弹性不足，部分管理工作需要中断服务，目前最新的3.X.X版本已经弹性十足，能够满足对存储系统弹性要求高的应用需求，尤其是对云存储服务系统而言意义更大。GlusterFS主要通过存储虚拟化技术和逻辑卷管理来实现这一设计目标。

##### （2）线性横向扩展（Linear Scale-Out）

线性扩展对于存储系统而言是非常难以实现的，通常系统规模扩展与性能提升之间是LOG对数曲线关系，因为同时会产生相应负载而消耗了部分性能的提升。现在的很多并行/集群/分布式文件系统都具很高的扩展能力，Luster存储节点可以达到1000个以上，客户端数量能够达到25000以上，这个扩展能力是非常强大的，但是Lustre也不是线性扩展的。

纵向扩展（Scale-Up）旨在提高单个节点的存储容量或性能，往往存在理论上或物理上的各种限制，而无法满足存储需求。横向扩展（Scale-Out）通过增加存储节点来提升整个系统的容量或性能，这一扩展机制是目前的存储技术热点，能有效应对容量、性能等存储需求。目前的并行/集群/分布式文件系统大多都具备横向扩展能力。

GlusterFS是线性横向扩展架构，它通过横向扩展存储节点即可以获得线性的存储容量和性能的提升。因此，结合纵向扩展GlusterFS可以获得多维扩展能力，增加每个节点的磁盘可增加存储容量，增加存储节点可以提高性能，从而将更多磁盘、内存、I/O资源聚集成更大容量、更高性能的虚拟存储池。GlusterFS利用三种基本技术来获得线性横向扩展能力：

1)        消除元数据服务

2)        高效数据分布，获得扩展性和可靠性

3)        通过完全分布式架构的并行化获得性能的最大化

##### （3）高可靠性（Reliability）

与GFS（Google File System）类似，GlusterFS可以构建在普通的服务器和存储设备之上，因此可靠性显得尤为关键。GlusterFS从设计之初就将可靠性纳入核心设计，采用了多种技术来实现这一设计目标。首先，它假设故障是正常事件，包括硬件、磁盘、网络故障以及管理员误操作造成的数据损坏等。GlusterFS设计支持自动复制和自动修复功能来保证数据可靠性，不需要管理员的干预。其次，GlusterFS利用了底层EXT3/ZFS等磁盘文件系统的日志功能来提供一定的数据可靠性，而没有自己重新发明轮子。再次，GlusterFS是无元数据服务器设计，不需要元数据的同步或者一致性维护，很大程度上降低了系统复杂性，不仅提高了性能，还大大提高了系统可靠性

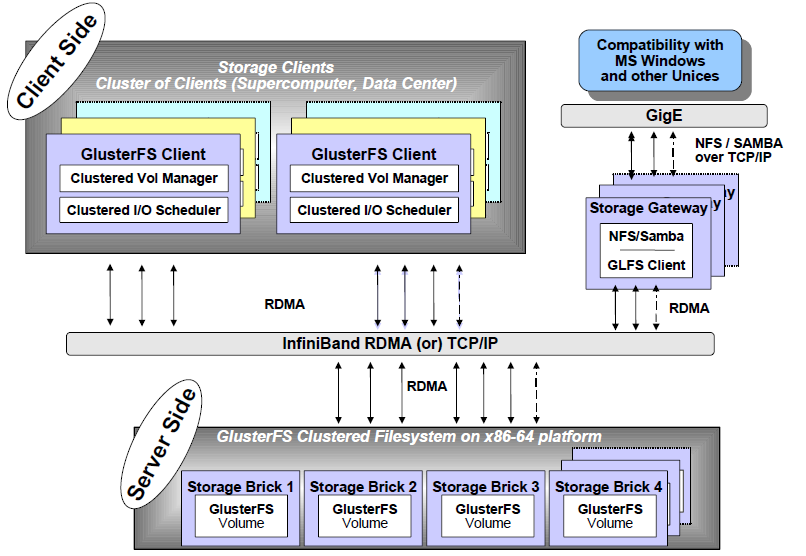
##### （4）全局统一命名空间

Glusterfs采用了全局统一命名空间设计，将磁盘和内存资源聚集成一个单一的虚拟存储池进行管理，并在此命名空间中使用NFS/CIFS等标准协议来访问应用数据。与其他分布式文件系统有所不同的是，GlusterFS中没有专用的元数据服务器，而是独特地采用无元数据服务的设计，取而代之使用算法来定位文件，元数据和数据没有分离而是一起存储。这使得数据访问完全并行化，从而实现真正的线性性能扩展。无数据服务器极大提高了GlusterFS的性能、可靠性和稳定性。

##### （5）平衡分布

GlusterFS的哈希分布是以目录为基本单位的，文件的父目录利用扩展属性记录了子卷映射信息，子文件在父目录所属存储服务器中进行分布。由于文件目录事先保存了分布信息，因此新增节点不会影响现有文件存储分布，它将从此后的新创建目录开始参与存储分布调度。这种设计，新增节点不需要移动任何文件，但是负载均衡没有平滑处理，老节点负载较重。GlusterFS在设计中考虑了这一问题，在新建文件时会优先考虑容量负载最轻的节点，在目标存储节点上创建文件链接直向真正存储文件的节点。此外，GlusterFS弹性卷管理工具可以在后台以人工方式来执行负载平滑，将进行文件移动和重新分布，此后所有存储服务器都会均会被调度。

#### 3.总体架构设计



GlusterFS总体架构与组成部分如图所示，它主要由存储服务器（Brick Server）、客户端以及NFS/Samba存储网关组成。不难发现，GlusterFS架构中没有元数据服务器组件，这是其最大的设计这点，对于提升整个系统的性能、可靠性和稳定性都有着决定性的意义。GlusterFS支持TCP/IP和InfiniBand RDMA高速网络互联，客户端可通过原生Glusterfs协议访问数据，其他没有运行GlusterFS客户端的终端可通过NFS/CIFS标准协议通过存储网关访问数据。

存储服务器主要提供基本的数据存储功能，最终的文件数据通过统一的调度策略分布在不同的存储服务器上。它们上面运行着Glusterfsd进行，负责处理来自其他组件的数据服务请求。如前所述，数据以原始格式直接存储在服务器的本地文件系统上，如EXT3、EXT4、XFS、ZFS等，运行服务时指定数据存储路径。多个存储服务器可以通过客户端或存储网关上的卷管理器组成集群，如Stripe（RAID0）、Replicate（RAID1）和DHT（分布式Hash）存储集群，也可利用嵌套组合构成更加复杂的集群，如RAID10。

由于没有了元数据服务器，客户端承担了更多的功能，包括数据卷管理、I/O调度、文件定位、数据缓存等功能。客户端上运行Glusterfs进程，它实际是Glusterfsd的符号链接，利用FUSE（File system in User Space）模块将GlusterFS挂载到本地文件系统之上，实现POSIX兼容的方式来访问系统数据。在最新的3.X.X版本中，客户端不再需要独立维护卷配置信息，改成自动从运行在网关上的glusterd弹性卷管理服务进行获取和更新，极大简化了卷管理。GlusterFS客户端负载相对传统分布式文件系统要高，包括CPU占用率和内存占用。

GlusterFS存储网关提供弹性卷管理和NFS/CIFS访问代理功能，其上运行Glusterd和Glusterfs进程，两者都是Glusterfsd符号链接。卷管理器负责逻辑卷的创建、删除、容量扩展与缩减、容量平滑等功能，并负责向客户端提供逻辑卷信息及主动更新通知功能等。GlusterFS 3.X.X实现了逻辑卷的弹性和自动化管理，不需要中断数据服务或上层应用业务。对于Windows客户端或没有安装GlusterFS的客户端，需要通过NFS/CIFS代理网关来访问，这时网关被配置成NFS或Samba服务器。相对原生客户端，网关在性能上要受到NFS/Samba的制约。

#### 444

GlusterFS是模块化堆栈式的架构设计，如图所示。模块称为Translator，是GlusterFS提供的一种强大机制，借助这种良好定义的接口可以高效简便地扩展文件系统的功能。服务端与客户端模块接口是兼容的，同一个translator可同时在两边加载。每个translator都是SO动态库，运行时根据配置动态加载。每个模块实现特定基本功能，GlusterFS中所有的功能都是通过translator实现，比如Cluster, Storage, Performance, Protocol, Features等，基本简单的模块可以通过堆栈式的组合来实现复杂的功能。这一设计思想借鉴了GNU/Hurd微内核的虚拟文件系统设计，可以把对外部系统的访问转换成目标系统的适当调用。大部分模块都运行在客户端，比如合成器、I/O调度器和性能优化等，服务端相对简单许多。客户端和存储服务器均有自己的存储栈，构成了一棵Translator功能树，应用了若干模块。模块化和堆栈式的架构设计，极大降低了系统设计复杂性，简化了系统的实现、升级以及系统维护。

#### 4.弹性哈希算法

对于分布式系统而言，元数据处理是决定系统扩展性、性能以及稳定性的关键。GlusterFS另辟蹊径，彻底摒弃了元数据服务，使用弹性哈希算法代替传统分布式文件系统中的集中或分布式元数据服务。这根本性解决了元数据这一难题，从而获得了接近线性的高扩展性，同时也提高了系统性能和可靠性。GlusterFS使用算法进行数据定位，集群中的任何服务器和客户端只需根据路径和文件名就可以对数据进行定位和读写访问。换句话说，GlusterFS不需要将元数据与数据进行分离，因为文件定位可独立并行化进行。GlusterFS中数据访问流程如下：

1、计算hash值，输入参数为文件路径和文件名；

2、根据hash值在集群中选择子卷（存储服务器），进行文件定位；

3、对所选择的子卷进行数据访问。

GlusterFS目前使用Davies-Meyer算法计算文件名hash值，获得一个32位整数。Davies-Meyer算法具有非常好的hash分布性，计算效率很高。假设逻辑卷中的存储服务器有N个，则32位整数空间被平均划分为N个连续子空间，每个空间分别映射到一个存储服务器。这样，计算得到的32位hash值就会被投射到一个存储服务器，即我们要选择的子卷。难道真是如此简单？现在让我们来考虑一下存储节点加入和删除、文件改名等情况，GlusterFS如何解决这些问题而具备弹性的呢？

逻辑卷中加入一个新存储节点，如果不作其他任何处理，hash值映射空间将会发生变化，现有的文件目录可能会被重新定位到其他的存储服务器上，从而导致定位失败。解决问题的方法是对文件目录进行重新分布，把文件移动到正确的存储服务器上去，但这大大加重了系统负载，尤其是对于已经存储大量的数据的海量存储系统来说显然是不可行的。另一种方法是使用一致性哈希算法，修改新增节点及相邻节点的hash映射空间，仅需要移动相邻节点上的部分数据至新增节点，影响相对小了很多。然而，这又带来另外一个问题，即系统整体负载不均衡。GlusterFS没有采用上述两种方法，而是设计了更为弹性的算法。GlusterFS的哈希分布是以目录为基本单位的，文件的父目录利用扩展属性记录了子卷映射信息，其下面子文件目录在父目录所属存储服务器中进行分布。由于文件目录事先保存了分布信息，因此新增节点不会影响现有文件存储分布，它将从此后的新创建目录开始参与存储分布调度。这种设计，新增节点不需要移动任何文件，但是负载均衡没有平滑处理，老节点负载较重。GlusterFS在设计中考虑了这一问题，在新建文件时会优先考虑容量负载最轻的节点，在目标存储节点上创建文件链接直向真正存储文件的节点。另外，GlusterFS弹性卷管理工具可以在后台以人工方式来执行负载平滑，将进行文件移动和重新分布，此后所有存储服务器都会均会被调度。

GlusterFS目前对存储节点删除支持有限，还无法做到完全无人干预的程度。如果直接删除节点，那么所在存储服务器上的文件将无法浏览和访问，创建文件目录也会失败。当前人工解决方法有两个，一是将节点上的数据重新复制到GlusterFS中，二是使用新的节点来替换删除节点并保持原有数据。

如果一个文件被改名，显然hash算法将产生不同的值，非常可能会发生文件被定位到不同的存储服务器上，从而导致文件访问失败。采用数据移动的方法，对于大文件是很难在实时完成的。为了不影响性能和服务中断，GlusterFS采用了文件链接来解决文件重命名问题，在目标存储服务器上创建一个链接指向实际的存储服务器，访问时由系统解析并进行重定向。另外，后台同时进行文件迁移，成功后文件链接将被自动删除。对于文件移动也作类似处理，好处是前台操作可实时处理，物理数据迁移置于后台选择适当时机执行。

  弹性哈希算法为文件分配逻辑卷，那么GlusterFS如何为逻辑卷分配物理卷呢？GlusterFS3.1.X实现了真正的弹性卷管理，如图4所示。存储卷是对底层硬件的抽象，可以根据需要进行扩容和缩减，以及在不同物理系统之间进行迁移。存储服务器可以在线增加和移除，并能在集群之间自动进行数据负载平衡，数据总是在线可用，没有应用中断。文件系统配置更新也可以在线执行，所作配置变动能够快速动态地在集群中传播，从而自动适应负载波动和性能调优。

弹性哈希算法本身并没有提供数据容错功能，GlusterFS使用镜像或复制来保证数据可用性，推荐使用镜像或3路复制。复制模式下，存储服务器使用同步写复制到其他的存储服务器，单个服务器故障完全对客户端透明。此外，GlusterFS没有对复制数量进行限制，读被分散到所有的镜像存储节点，可以提高读性能。弹性哈希算法分配文件到唯一的逻辑卷，而复制可以保证数据至少保存在两个不同存储节点，两者结合使得GlusterFS具备更高的弹性。

#### 5.场景与方案

GlusterFS是一个具有高扩展性、高性能、高可用性、可横向扩展的弹性分布式文件系统，在架构设计上非常有特点，比如无元数据服务器设计、堆栈式架构等。然而，存储应用问题是很复杂的，GlusterFS也不可能满足所有的存储需求，设计实现上也一定有考虑不足之处，下面我们作简要分析。

##### （1）无元数据服务器 vs 元数据服务器

无元数据服务器设计的好处是没有单点故障和性能瓶颈问题，可提高系统扩展性、性能、可靠性和稳定性。对于海量小文件应用，这种设计能够有效解决元数据的难点问题。它的负面影响是，数据一致问题更加复杂，文件目录遍历操作效率低下，缺乏全局监控管理功能。同时也导致客户端承担了更多的职能，比如文件定位、名字空间缓存、逻辑卷视图维护等等，这些都增加了客户端的负载，占用相当的CPU和内存。

##### （2）用户空间 vs 内核空间

用户空间实现起来相对要简单许多，对开发者技能要求较低，运行相对安全。用户空间效率低，数据需要多次与内核空间交换，另外GlusterFS借助FUSE来实现标准文件系统接口，性能上又有所损耗。内核空间实现可以获得很高的数据吞吐量，缺点是实现和调试非常困难，程序出错经常会导致系统崩溃，安全性低。纵向扩展上，内核空间要优于用户空间，GlusterFS有横向扩展能力来弥补。

##### （3）堆栈式 vs 非堆栈式

这有点像操作系统的微内核设计与单一内核设计之争。GlusterFS堆栈式设计思想源自GNU/Hurd微内核操作系统，具有很强的系统扩展能力，系统设计实现复杂性降低很多，基本功能模块的堆栈式组合就可以实现强大的功能。查看GlusterFS卷配置文件我们可以发现，translator功能树通常深达10层以上，一层一层进行调用，效率可见一斑。非堆栈式设计可看成类似Linux的单一内核设计，系统调用通过中断实现，非常高效。后者的问题是系统核心臃肿，实现和扩展复杂，出现问题调试困难。

##### （4）原始存储格式 vs 私有存储格式

GlusterFS使用原始格式存储文件或数据分片，可以直接使用各种标准的工具进行访问，数据互操作性好，迁移和数据管理非常方便。然而，数据安全成了问题，因为数据是以平凡的方式保存的，接触数据的人可以直接复制和查看。这对很多应用显然是不能接受的，比如云存储系统，用户特别关心数据安全，这也是影响公有云存储发展的一个重要原因。私有存储格式可以保证数据的安全性，即使泄露也是不可知的。GlusterFS要实现自己的私有格式，在设计实现和数据管理上相对复杂一些，也会对性能产生一定影响。

##### （5）大文件 vs 小文件

GlusterFS适合大文件还是小文件存储？弹性哈希算法和Stripe数据分布策略，移除了元数据依赖，优化了数据分布，提高数据访问并行性，能够大幅提高大文件存储的性能。对于小文件，无元数据服务设计解决了元数据的问题。但GlusterFS并没有在I/O方面作优化，在存储服务器底层文件系统上仍然是大量小文件，本地文件系统元数据访问是一个瓶颈，数据分布和并行性也无法充分发挥作用。因此，GlusterFS适合存储大文件，小文件性能较差，还存在很大优化空间。

##### （6）可用性 vs 存储利用率

GlusterFS使用复制技术来提供数据高可用性，复制数量没有限制，自动修复功能基于复制来实现。可用性与存储利用率是一个矛盾体，可用性高存储利用率就低，反之亦然。采用复制技术，存储利用率为1/复制数，镜像是50%，三路复制则只有33%。其实，可以有方法来同时提高可用性和存储利用率，比如RAID5的利用率是(n-1)/n，RAID6是(n-2)/n，而纠删码技术可以提供更高的存储利用率。但是，鱼和熊掌不可得兼，它们都会对性能产生较大影响。

#### 6.名词解释

|  |  |
| --- | --- |
| 名称 | 说明 |
| Glusterd | Gluster文件系统的管理服务，启动glusterd守护进程，此服务会创建两个socket分别用来和本地cli、集群其他节点、glusterfsd服务通信，并且使用多进程恢复本节点所有（peer,volume,brick等）状态（pipe通信），最后使用多线程调用epoll一直等待数据流接入；存储集群节点间通过peer操作建立两条通信链路（服务器端口是24007），即一个节点既是服务器又是客户端，以此保证集群无中心化 |
| Glusterfsd | Brick服务，每个磁盘运行一个这样的服务，此服务启动后与glusterd建立链接（链接在24007端口），同时glusterfsd还有有两个本地socket通信端口，用来和本机其他bricks-glusterfsd通信做数据传输 |
| Glusterfs | 客户端的服务，恢复功能的服务（两者参数不一样）客户端通过挂载服务器卷启动glusterfs服务，首先与挂载点的glusterd服务建立通信获取挂载卷的信息，然后与挂载卷的每个brick服务通过建立tcp/ip通信进行数据传输；恢复功能原理与之类似 |
| Gluster | 命令行工具，用来执行集群，卷相关操作，通过glusterd.socket文件与glusterd进程通信，并执行相应的动作（如果glusterd 的socket端口没有建立listen，那cli命令无法通信直接返回失败；如果listen建立但epoll没有处理或处理函数有异常会导致cli卡死） |
| Peer | 集群的概念，以ip节点为单位 |
| Volume | 卷的概念，可以将集群peer中的任何brick创建为一个卷，提供给客户端挂载 |

#### 7.组件日志

Server端

Glusterd --/var/log/glusterfs/glusterd.log

Glusterfsd --/var/log/glusterfs/bricks/brick-name.log

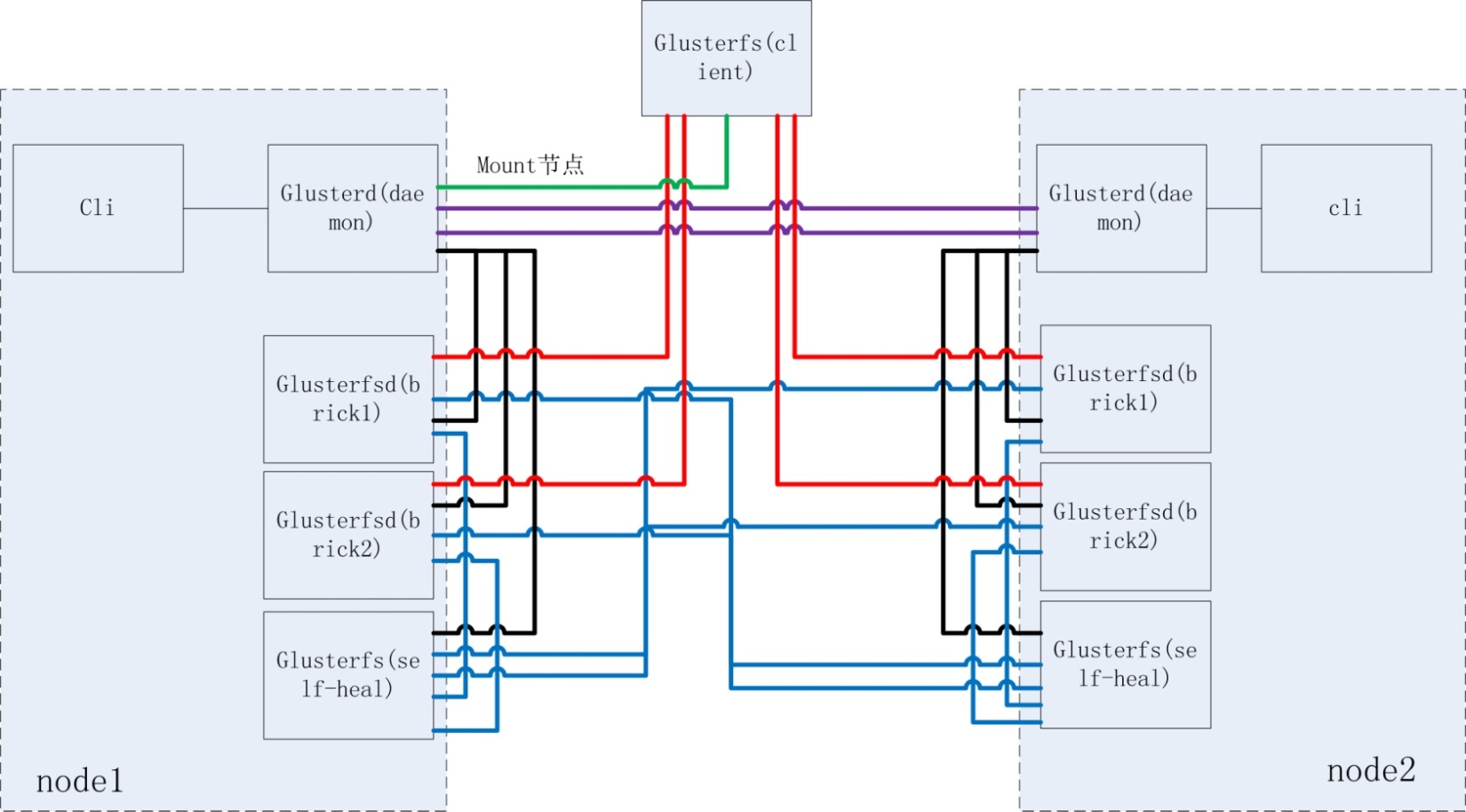
Glusterfs --/var/log/glusterfs/glustershd.log

Gluster --/var/log/glusterfs/cli.log

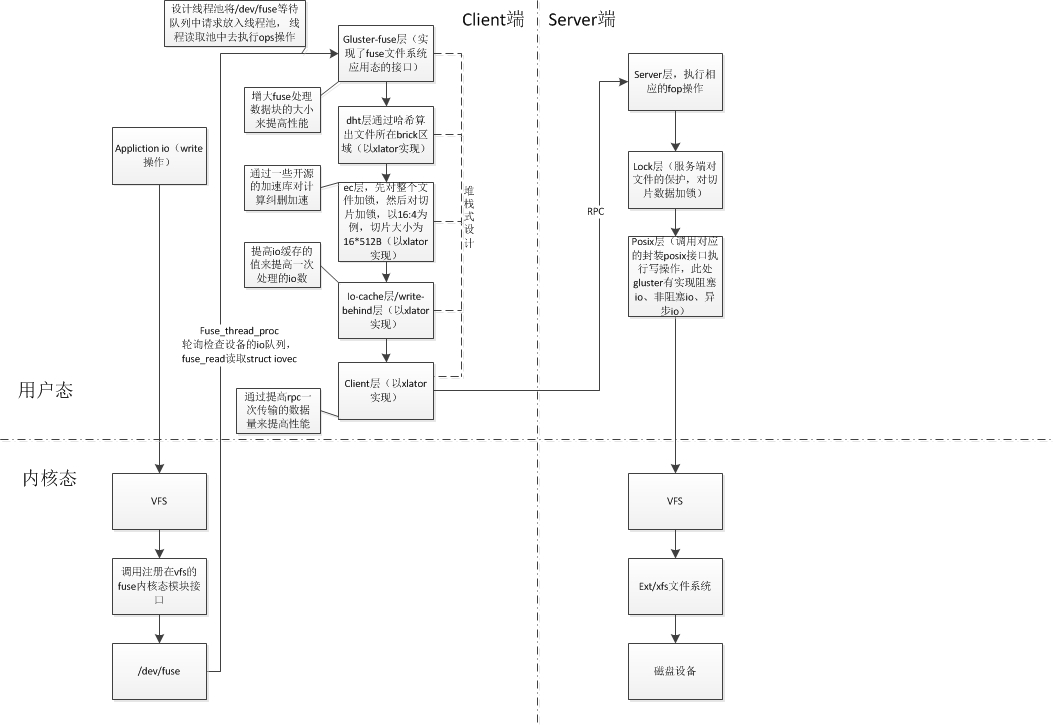
Client端

Glusterfs --/var/log/glusterfs/mount-point.log

#### 8.组件之间通信关系



#### 9.gluster读写流程



#### 10.版本选择

目前最新版本已经到7，我们选择3.13；最新新版本相对于3.12新增特性：

1.glusterfind与gfid2path的集成

2.客户端inode垃圾回收 lru-limit

2.增强fuse挂载，可以处理锁请求的中断

3.非分布式数据分发跳过不必要的检测和操作

4.增加内核缓存页无效的开关，当多个客户端不同时操作卷中的文件时，可以保持更长时间的内核缓存页performance.global-cache\_invalidation

5.关闭smb的默认选项

6.禁用ctime开关

7.准备python3环境

8.增强文件系统中条目操作的一致性

9.ctime在数据改变时也改变，则将其设为quick read选项ctime-invalidation

10.shard删除并行分片数 shard-deletion-rate

11.删除md5的使用，替换为SHA256

12.删除lock-heal，grace-timeout功能

13.glusterd2管理服务（实验版）

    -嵌入etcd来存储池配置，

    -直接读取xlators.so避免与xlator共享库的信息不匹配问题

    -新的自修复功能

    -简化异地容灾方式

14.ec元数据getfattr性能改进

15.md-cache，允许运行时将xattrs添加到md-cache的缓存列表

    -xattr-cache-list "xattr-name"

16.disperse可以缓存最后一个写条带的数据，有益于追加顺序io的操作

    -disperse.stripe-cache N

17.加强self-heal，识别daemon正在处理的inode，防止其他线程等待特定对象

18.ec并行xatrrrop

19.增强文件系统一致性 features.sdfs enable

20.限制针对inode创建的硬链接数 storage.max-hardlinks

    -默认100，0表示关闭保留本地文件系统默认，1关闭硬链接

21.更改平衡brick操作的方式，优先处理子目录

22.平衡过程中如有应用程序的写入，将跳过文件迁移，

    -cluster.force-migration on/off，打开则强制迁移

=======3.13==========

23.新增heal info summary，可以显示brick中等待回复和正在恢复数目的统计信息

24.保留brick存储空间

    -storage.reserve n  默认为1，设置为0表示禁用

25.ec模式下得并行写入

    -disperse.parallel-writes

26.包含关于statedumps中内存池得信息

### （三）部署glusterfs集群

#### 1、安装gluster

##### （1）主机提供服务端与客户端的安装包脚本（已包含arm服务端安装后启动服务，客户端不需要启动该服务）

# sh installer.sh --install

##### （2）手动启动/关闭/查看状态 glusterd服务

#/etc/init.d/glusterd start/stop/status [\\启动/关闭glusterd服务](\\\\启动glusterd服务)

#### 2、部署gluster集群准备工作

##### （1）编辑server端和client端每个节点的/etc/hosts文件

# vi /etc/hosts

添加集群中每个节点的主机名和主机的IP地址的对应关系，如下：

Server-ip node1

Server-ip node2

...

Server-ip node120

127.0.0.1 localhost

##### 对每个节点进行时间同步

1. 手动设置时间

# date //查看系统时间是否正确，正确的话则忽略下面两步

# date -s "2019-10-30 19:05:05" //设置系统时间

# hwclock -w //写入硬件时间

B．使用ntpdate服务同步时间

# yum install -y ntp ntpdate //安装ntp

# systemctl start ntpd //主节点启动ntpd服务

# ntpdate server-ip //其他节点同步主节点时间

##### （4）关闭防火墙(centos7)

命令如下：

# systemctl stop firewalld.servic //停止firewall

# systemctl disable firewalld.service //禁止firewall开机启动

##### （5）关掉selinux(centos7)

命令查看出selinux的状态，命令# sestatus -v或者# getenforce

目前 SELinux 支持三种模式，分别如下：

•enforcing：强制模式，代表 SELinux 运作中，且已经正确的开始限制 domain/type 了；

•permissive：宽容模式：代表 SELinux 运作中，不过仅会有警告讯息并不会实际限制 domain/type 的存取。这种模式可以运来作为 SELinux 的 debug 之用；

•disabled：关闭，SELinux 并没有实际运作。

A．永久方法 – 需要重启服务器（推荐）

编辑# vi /etc/selinux/config，改为SELINUX=disabled，然后重启服务器# reboot。

B．临时方法 – 设置系统参数，服务器重启后失效

使用命令# setenforce 0

附：

# setenforce 1 设置SELinux 成为enforcing模式

# setenforce 0 设置SELinux 成为permissive搜索模式

#### 创建GlusterFS集群

##### 集群组网

由于集群规模限制问题，现考虑如下方式组网：现有10台arm服务器，每台服务器12节点，每节点2块磁盘，以40-40-40方式组成3个集群，node1~node40, node41~node80,node81~node120; 配置为固定模式；

如果用户自由配置集群，集群数规范为偶数，不要超过50。

##### 创建集群

在node1上执行：

# gluster peer probe node2

# gluster peer probe node3

...

gluster peer probe node40

在node41上执行：

gluster peer probe node42

gluster peer probe node43

...

在node81上执行;

gluster peer probe node82

gluster peer probe node83

...

gluster peer probe node120

##### 移除集群

# gluster peer detach node2

Probe successful

##### ****挂载分区****

格式化磁盘（建议为xfs），创建挂载点，名称建议为/brick{1..n}

#mkdir /brick1

使用#lsblk命令可以清晰的获取全局的块设备布局，一般服务器都有多个硬盘分区，在重启后，这些分区的逻辑位置加载时可能会发生变动，如果使用传统的设备名称(例如：/dev/sda)方式挂载磁盘，就可能因为磁盘顺序变化而造成混乱。Linux环境中每个Block Device都有一个全局唯一的UUID，可以标识这个设备，#blkid命令可以获取设备的UUID，然后复制记下设备的UUID。例如：

#mount UUID="b510d40d-c28a-49c2-a6fc-db8c97adecd8" /brick1

设置开机启动自动挂载，# vi /etc/fstab，在最后加上一行，然后:wq保存退出。

UUID=b510d40d-c28a-49c2-a6fc-db8c97adecd /brick1 xfs defaults 0 0

不能直接使用挂载目录/brick1直接来作为gluster的卷参数，挂载好后需要增加一层：

#mkdir /brick1/data

#### 创建GlusterFS卷

##### （1）卷模式说明（数据冗余）

* 复制卷 - 在高可用性和高可靠性环境中使用复制卷。副本数越多，数据可用性越好，可靠性也越高，但也意味着更低的空间利用率以及更高的成本。复制卷最少配置复制卷数为2。如果考虑可能出现的脑裂问题（几乎主要由于客户端与brick程序断连导致），可以参考多配置一个仲裁盘arbiter 1或者配置为3副本；建议直接配置为3副本最好。
* 纠删卷 - 纠删码机制具有更高的存储效率，在提供相同存储可靠性的条件下，可以最小化冗余存储开销，但是相对于复制卷其IOPS性能下降较多。Disperse卷的创建与节点个数无关（节点个数大于等于1），只与bricks(B)、冗余度redunancy(R)相关；其中bricks(B)必须大于等于3，disperse-data 的个数必须要大于等于2，redunancy(R)的值最小为1，最大为(B-1)/2、必须小于bricks(B)的一半且值是不能改变的。disperse-data值要小于等于16 。

纠删卷中，会把每个读写请求切分为大小相同的Chunk块，而每个Chunk块又被分割成(B-R)个大小为512bytes的Fragment数据分片；然后使用算法Rabin IDA计算生成R个大小为512bytes的Fragment校验分片；最后把(B-R)个数据分片和R个校验分片以条带的方式存储在一起，即分别存储于每个Brick上，从而降低访问热点；其中R个校验分片会以轮询轮的方式存储于卷的每个brick上，用以提高卷的可靠性。（注：Fragment的大小在GlusterFS的源码中是一个宏定义，其大小等于EC\_METHOD\_WORD\_SIZE\* EC\_GF\_BITS=64\*8=512 bytes）Disperse卷中，Chunk的大小可配置，其大小与具体的Redundancy配置有关，其大小等于512\*(B-R) bytes。可通过调整Redundancy的配置（注：Redundancy的配置在Disperse卷创建之后就确定，不可修改），来修改Chunk的大小。那么以官方经典的配置B=6，R=2的Disperse卷为例，得出Chunk的大小为（6-2）\*512=2048 bytes。

##### 卷的模式

1. 冷存采用纠删卷方式，目前按照推荐组网一个集群40\*2个brick，则推荐2种纠删比：6：2和8：2；并推荐两种卷方式为：

A.分布式纠删卷（对客户端只提供一个卷）

10\*（6：2）或者8\*（8：2）

B.非分布式纠删卷（对客户端只提供多个卷）

10个独立的6：2或者 8个独立的8：2

1. 如果用户自由配置卷比，纠删比给计算公式，数据盘规范为偶数，数据盘数不能超过16，校验盘数计算公式为1<redundancy <disperse-data；redundancy=brick-disperse-data

##### 创建卷，启动卷

**创建纠删卷：（以8：2为例）**

# gluster volume create test-volume disperse-data 8 redundancy 2 node1:/brick1/data node2:/brick1/data node3:/brick1/data ... node40:/brick1/data ... node1:/brick2/data node2:/brick2/data ... node40:/brick2/data

# gluster volume start test-volume

##### 在client上分别mount以上创建的卷

#mount -t glusterfs node1:test-volume /mntpoint1

#mount -t glusterfs node40:test-volume /mntpoint2

#mount -t glusterfs node80:test-volume /mntpoint3

开机自启动：

Vi /etc/fstable

server:/test-volume /mountpoint glusterfs defaults,\_netdev 0 0

##### 扩展卷

当需要容量扩展时，对于副本卷和纠删卷需要添加其倍数。添加后需要平衡（见（8）卷平衡）

###### 副本

#gluster volume add-brick test-volume replica 2 node1:/brick2/data node2:/brick2/data

###### 纠删（不写纠删比）

#gluster volume add-brick test-volume node1:/brick2/data node2:/brick2/data

##### 收缩卷

当因为磁盘原因需要收缩容量时，对于副本和纠删需要删除其倍数，并且是同一组的需要全部删除，可以一次删除多组；收缩卷需要先进行数据迁移，等待完成之后再确认删除。

###### 副本

#gluster volume remove-brick test-volume replica 2 node1:/brick2/data node2:/brick2/data start

查看状态（此时brick依然存在于卷中）

#gluster volume remove-brick test-volume replica 2 node1:/brick2/data node2:/brick2/data status

确认状态完成之后再commit

#gluster volume remove-brick test-volume replica 2 node1:/brick2/data node2:/brick2/data 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.

###### 纠删(不用写纠删比,以2：1为例)

#gluster volume remove-brick test-volume node1:/brick2/data node2:/brick2/data node3:/brick2/data start

查看状态（此时brick依然存在于卷中）

#gluster volume remove-brick test-volume node1:/brick2/data node2:/brick2/data node3:/brick2/data status

确认状态完成之后再commit

#gluster volume remove-brick test-volume node1:/brick2/data node2:/brick2/data node3:/brick2/data 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.

##### Brick替换

场景发生在当磁盘故障时需要替换磁盘，在原盘已经故障的情况下（数据已丢失），重新换上新盘存储池并不能识别，因为数据已丢失所以不存在等待迁移数据问题，直接强制提交即可。同时新换上的盘挂载目录名不能与故障目录同名。替换成功之后不需要其他动作冗余卷自身的恢复功能会自动恢复数据。推荐一次替换一个盘，替换之后等待数据恢复之后再替换其他盘以防数据丢失。关于数据恢复可以参照本文档自恢复功能模块。

#gluster volume replace-brick test-volume node1:/brick1/data node1:/brick1/data-new commit force

##### 卷平衡

扩容卷（add-brick命令）或者重命名大量文件后，需要重新平衡服务器之间的数据。扩大后创建的新目录将自动均匀分布。GlusterFS的哈希分布是以目录为基本单位的，文件的父目录利用扩展属性记录了子卷映射信息，子文件在父目录所属存储服务器中进行分布。由于文件目录事先保存了分布信息，因此新增节点不会影响现有文件存储分布，它将从此后的新创建目录开始参与存储分布调度。这种设计，新增节点不需要移动任何文件，但是负载均衡没有平滑处理，老节点负载较重。GlusterFS在设计中考虑了这一问题，在新建文件时会优先考虑容量负载最轻的节点，在目标存储节点上创建文件链接直向真正存储文件的节点。此外，GlusterFS弹性卷管理工具可以在后台以人工方式来执行负载平滑，将进行文件移动和重新分布，此后所有存储服务器都会均会被调度。

对于所有现有的目录中在平衡可以修改分布式出现的不均衡。分为修复布局和迁移数据两步：

###### 修复布局：

gluster文件定位是按照哈希来确定文件位置，brick记录了哈希的范围，不管是add-brick还是remove-brick原有的哈希分布布局都不变，这样即使新写的文件也无法定位到新brick上去，这时就需要修复布局.原理流程为：

1. 对mount point递归调用sys\_lgetxattr(fullpath, "trusted.distribute.fix.layout", &value, 128)
2. 每次调用触发fuse translator，并传递触发dht translator
3. 触发调用dht translator接口函数dht\_getxattr
4. 由于指定了trusted.distribute.fix.layout，触发dht\_selfheal\_new\_directory进行目录layout修复

#gluster volume rebalance test-volume fix-layout start

volume rebalance: rep: success: Rebalance on rep has been started successfully. Use rebalance status command to check status of the rebalance process.

查看平衡状态：

#gluster volume rebalance test-volume status

###### 数据迁移：

布局修复完成之后，对原来集群中的数据需要进行迁移，避免分布不均。新版本中已经没有了migrate-date命令，直接start是直接将两者进行了合并，不过原理流程都是一样。

1. 对mount point递归遍历目录两遍
2. 第一遍只对文件进行操作，进行文件迁移
3. copy文件至临时文件（临时文件需要位于mount point下）
4. 复制属性，迁移扩展属性，更新uid/gid/time
5. rename临时文件名为原文件名
6. 第二遍只对子目录进行操作，递归对子目录调用gf\_glusterd\_rebalance\_move\_data

#gluster volume rebalance test-volume start

volume rebalance: rep: success: Rebalance on rep has been started successfully. Use rebalance status command to check status of the rebalance process.

查看平衡状态：

#gluster volume rebalance test-volume status

##### 停止卷

#gluster volume stop test-volume

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

volume stop: rep: success

##### 删除卷

#gluster volume delete test-volume

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

volume delete: rep: success

#### GlusterFS卷功能

##### 存储集群状态（频次在5分钟以内，暂定为3分钟）

1. 集群状态主要看各节点链接状态，正常状态为connected，异常状态为disconnected; 对应节点看hostname

# gluster pool list



表 1 正常状态

1. Node2掉线或者glusterd管理服务挂死时，node2的状态为disconnected，但是只能在正常状态的节点上查看，如果node2是glusterd管理服务挂死，那么集群状态和存储池状态（见（2））都查询不到，会显示“Connection failed. Please check if gluster daemon is operational”，此时要显示状态信息就需要ssh登陆其他节点来查看状态（优先配置好ssh无密码访问）：



表 2 异常状态

##### 存储池（卷）状态（频次在5分钟以内，暂定为3分钟）

Gluster存储池的状态并不像ceph有一个healthy状态，所以需要具体的来判断。分基础状态和功能状态，Y为上线，N或者不存在为下线：

1. 基础状态：所有的brick全部上线online-Y，这个判断是保证gluster存储池读写数据的基本状态要求，brick掉线或者down掉则显示online-N
2. 功能状态：

B1. 类似纠删卷和副本卷如果开启修复功能均有self-heal服务处于上线online-Y；如果关闭不显示该服务。（数据恢复功能默认是启动的）

B2. 如果开启配额功能有quota服务处于上线online-Y。关闭quota功能则不显示该服务

# gluster volume status test-volume

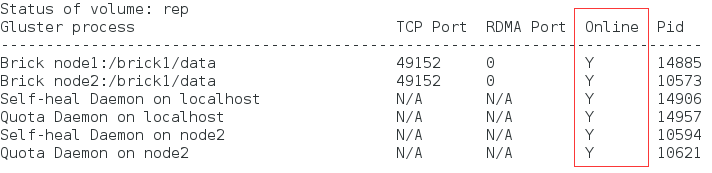


表 3 启动配额，启动恢复，brick全部上线

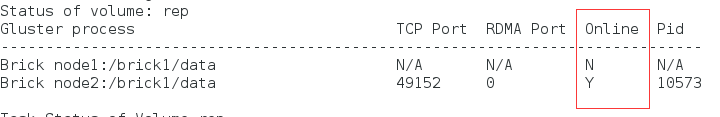


表 4 关闭配额，关闭恢复，brick1掉线

1. 以上状态均基于gluster集群处于正常状态，当发生（1）中B状态时，在正常节点上查询存储池状态不显示异常节点的任何信息（包括brick服务和功能服务）

##### 目录配额

启动配额功能

# gluster volume quota test-volume enable

volume quota : success

由于性能原因quota会缓存客户端上的目录大小，导致写入可能会大于配额。所以我们需要设置超时来指示缓存目录的最大有效实际时间，当超时发生时，缓存中的大小将从服务器更新，并将同步，不允许进一步写入。设置为0s将从服务器获取修改文件数据的每个操作的目录大小，并将有效地禁用客户端的目录大小缓存。

# gluster volume quota test-volume hard-timeout 0

设置配额，不能设置挂载目录，设置的目录可以是挂载目录下的新目录，也可以是原来（开启quota之前）的目录，以‘/’开始，例如挂载目录是/mnt，下面新建data1目录，同时gluster本身允许设置子目录的配额大于父目录，但是实际写入时超过父目录大小会检验出来显示‘disk quota exceeded’，（建议平台可以自行判断子目录大小，比如如果设置一个目录10G的话，遍历数据库，他的所有父目录配额必须大于等于10G，不然不让设置，他的所有子目录配额均小于等于10G，不然不让设置；文件数同理）

1. 目录容量配额(可以与文件数同时设置)

# gluster volume quota test-volume limit-usage /data1 100MB

volume quota : success

# gluster volume quota test-volume limit-usage /data1/data11 300MB

volume quota : success

查看配额列表：

# gluster volume quota test-volume list

移除目录容量配额（直接删除目录也会移除配额）：

# gluster volume quota test-volume remove /data1

volume quota : success

1. 目录文件数配额（可以与容量同时设置）

# gluster volume quota test-volume limit-objects /data1 100

volume quota : success

# gluster volume quota test-volume limit-objects /data1/data11 50

volume quota : success

查看文件数配额列表：

# gluster volume quota test-volume list-objects

移除目录文件数配额（直接删除目录也会移除配额）：

# gluster volume quota test-volume remove-objects /data1

volume quota : success

关闭配额功能

# gluster volume quota test-volume disable

Disabling quota will delete all the quota configuration. Do you want to continue? (y/n) y

volume quota : success

##### 卷自修复

副本卷和纠删卷都有数据修复服务self-heal，默认都是开启的，可手动关闭或开启

# gluster volume heal test-volume enable/disable

卷自修复分为两种：index-heal、full-heal

1. Index-heal：触发条件有3种（每隔10分钟，通过gluster volume heal rep显示触发，当一个brick掉线又重新上线时），index-heal修复的是.glusterfs/indices/xattrop文件夹中的对象，都是需要修复的文件的硬链接，当客户端对文件操作时，每个brick进程的index-xlator都会在fop操作前添加硬链接，如果操作成功则完后将其删除，所以文件操作异常时.glusterfs/indices/xattrop/下就会存在一个对象，也就意味着该对象需要修复。

# gluster volume heal test-volume

1. Full-heal: （加开关，默认关闭）完全修复，只能是手动触发；场景一般是正常读写后的文件属性异常，或者在后端brick上直接删除/丢失了数据（.glusterfs配置目录正常）。与heal不同的是full-heal只在子卷的一个节点上运行，并且是uuid最大的一个节点。与index不同的是从卷的根开始修复。

# gluster volume heal test-volume full

1. 查看修复的条目（冷存集群规模较大，不要使用此命令）

# gluster volume heal test-volume info

1. 脑裂split-brain:在2副本的场景下可能会出现脑裂现象（互相认为对方有错，可以通过三副本或者仲裁来规避）再已经出现这种问题的情况下，修复heal是无法自动解决的，需要人为干预，首先查看是否脑裂信息，然后我们自己判断哪个文件异常，手动将其删除，然后执行heal来修复。

# gluster volume heal test-volume info split-brain

##### 客户端性能（带宽/iops）

参考interface-specification文档，平台控制调用频次定为1min（要求是5分钟以内）

##### 存储卷性能（带宽/iops）

参考interface-specification文档，平台控制调用频次定为1min（要求是5分钟以内）

##### 存储卷容量（总大小，空闲大小）

参考interface-specification文档，平台控制调用频次定为1min（要求是5分钟以内）

##### （8）存储卷恢复（等待恢复的文件总数）

参考interface-specification文档，平台控制调用频次定为3min（要求是5分钟以内）