# 高可用NAS集群设计

## 一、前言

在大数据时代的今天，云计算、云存储高速发展以及物联网的兴起导致数据呈爆炸式增长，其中非结构数据更是占据了全球数据的90%，于是一种基于横向扩展（Scale-out）存储架构的集群NAS诞生了，有别于传统的SAN和NAS，是一种新的存储架构，主要面向文件级别的存储，不但集中了SAN和NAS的优点，还具备它们不具备的一些优点，如容量和性能线性扩展等。集群NAS随着不断的扩容，其性能也会随之提升，理论上，达到一定规模的集群NAS在性能上可以胜过一个SAN系统，并且其成本远远低于SAN系统。现今，集群NAS已经得到了全球市场的广泛认可，成为了主流的存储技术之一。

在存储领域中，存储系统的高可用性一直是关注的重点，随着用户对于存储系统的可用性需求不断变化，高可用技术也在不断向前发展，从简单的双机热备到多节点的集群高可用，从底层磁盘的高可用到共享层服务的高可用，无论是从硬件层面还是软件层面都有显著的提升。本文将介绍一种可动态扩展、按需部署、具备透明故障切换的高可用集群NAS存储系统架构，主要基于分布式文件系统GLUSTERFS，高可用组件CTDB， Linux虚拟服务器LVS，以及标准协议（如NFS、CIFS、FTP等）部署搭建。

## 二、实践

### （一）实践环境

|  |  |  |
| --- | --- | --- |
| 软件环境： | OS：Centos 7.3.1611、KERNEL: 3.10.0-514、GlusterFS：3.12.15、  CTDB：4.9.1、ipvsadm：1.27、Samba：4.9.1、NFS：1.3.0 | |
| 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\*4\*4（只少不多）；集群目录深度\*单目录文件=10\*20万（只少不多）；不建议使用1M以下小文件 | |
| Hostname： | Node1 | Node2 |
| Private IP： | 192.168.15.130 | 192.168.15.129 |
| Public IP: | 192.168.15.141 | 192.168.15.142 |

### （二）部署glusterfs集群

#### 1、安装gluster

安装gluster通常有三种方式，编译安装、RPM安装、yum安装。

##### （1）编译安装

从网站<https://github.com/gluster/glusterfs/tree/release-3.12>下载glusterfs-3.12.15.tar.gz，然后编译安装。

首先配置编译环境，安装常用的软件包、工具包，命令如下：

#yum -y install autoconf libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel zip unzip ncurses ncurses-devel curl curl-devel e2fsprogs e2fsprogs-devel krb5-devel libidn libidn-devel openssl openssh openssl-devel nss\_ldap openldap openldap-devel openldap-clients openldap-servers libxslt-devel libevent-devel ntp libtool-ltdl bison libtool vim-enhanced python wget lsof iptraf strace lrzsz kernel-devel kernel-headers pam-devel Tcl/Tk cmake ncurses-devel bison setuptool popt-devel net-snmp screen perl-devel pcre-devel net-snmp screen tcpdump rsync sysstat man iptables sudo idconfig git system-config-network-tui bind-utils update arpscan tmux elinks numactl iftop bwm-ng flex

然后解压glusterfs包，使用命令#./configure；#make；#make install进行安装。

##### （2）使用RPM包安装

从网站https://buildlogs.centos.org/centos/7/storage/x86\_64/gluster-3.12/下载glusterfs相应的rpm安装包以及依赖包，使用命令：#rpm -ivh 进行安装。

##### （3）yum安装

首先，搜索gluster的软件源

# yum search centos-release-gluster

然后，安装相应gluster版本的软件源

# yum install centos-release-gluster12

最后安装需要的gluster软件包

# yum install -y glusterfs\*

##### （4）安装好后，启动和设置开机自动启动glusterd服务

#systemctl glusterd start \\启动glusterd服务

#systemct enable glusterd [\\设置开机启动](file:///\\\\设置开机启动)

#### 2、部署gluster集群准备工作

##### （1）编辑每个节点的/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：

192.168.15.130 node1

192.168.15.129 node2

##### （2）配置节点间ssh无密码访问

集群中每个节点相互配置ssh无密码访问，使用命令#ssh-keygen，#ssh-copy-id hostname/ip。注：配置ssh无密码访问不是必要的。

##### （3）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

如果安装了防火墙，则需要关闭它，命令如下：

# systemctl stop firewalld.service //停止firewall

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

##### （5）关掉selinux

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

#### 3、创建GlusterFS集群

##### （1）创建集群

# gluster peer probe node2

Probe successful

##### 移除集群

# gluster peer detach node3

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。
* 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。

##### 创建卷，启动卷

**创建纠删卷**，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.基于Glusterfs搭建的集群中，创建EC卷推荐以下几种配置：

a. 冗余度为1时，推荐创建配置为(2+1)的EC卷；

b. 冗余度为2时，推荐创建配置为(4 +2)的EC卷；

c. 冗余度为3时，推荐创建配置为(8 +3)的EC卷；

d. 冗余度为4时，推荐创建配置为(8 + 4)的EC卷。

由于在相同的EC配置下，Distributed Disperse卷比Disperse卷具有更好的IO性能，所以推荐在硬盘数量足够的情况下创建Distributed Disperse卷。在底层的配置中，推荐逻辑盘（如/dev/sda）不分区直接格式化为块大小512B的XFS文件系统，且逻辑盘与brick是一一对应的关系；推荐有n个节点，B就等于n，即同一组的Disperse卷配置中，一个brick对应一个节点。比如，三个节点的gluster集群就推荐创建配为（B=3，R=1）的EC卷，每组Disperse卷配置中，一个brick对应一个节点。

# gluster volume create test-volume disperse 3 redundancy 1 node1:/brick1/data node2:/brick1/data node3:/brick1/data node1:/brick2/data node2:/brick2/data node3:/brick2/data

Creation of test-volume has been successful

Please start the volume to access data.

# gluster volume start test-volume

**创建复制卷**,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。

# gluster volume create test-volume replica 3 node1:/brick1/data node2:/brick1/data node3:/brick1/data

Creation of test-volume has been successful

Please start the volume to access data.

# gluster volume start test-volume

##### 卷相关的参数值说明（建议保持默认值）

****client.event-threads /server.event-threads****

由于每个线程一次处理一个连接，所以不建议使用比brick进程更多的线程连接。配置比可用处理单元更高的事件线程值可能会再次导致这些线程上下文切换

****performance.io-thread-count****

调整事件线程值时可以提高性能的另一个参数是performance.io-thread-count（及其相关的线程数），可以将其设置为更高的值，因为这些线程在底层文件系统上执行实际的IO操作,IO线程translator试图增加服务器后台进程对文件元数据读写I/O的处理能力。由于GlusterFS服务是单线程的，使用IO线程translator可以较大的提高性能,IO线程操作会将读和写操作分成不同的线程。同一时刻存在的总线程是恒定的并且是可以配置的。建议小于或等于你的CPU数量

****cluster.read-hash-mode****

默认值： 1  
解释： 在副本卷情况下，这个值如果设置为0则会选择副本中的第一个启动brick读；1 则会根据gfid去选择，这样所有客户端都会选择同一个brick读； 2 是根据 文件gfid和客户端pid 选择副本中的brick，这样当挂载多个客户端则会选择副本中不同的brick。  
场景： 在副本卷模式下，多个客户端或者并发请求一个文件时候，选择2可以有概率使多个读请求分散到副本中的不同brick从而提升性能，不局限在磁盘的性能瓶颈上。

****performance.client-io-threads****

允许多线程并行，对纠删卷有用，副本卷不开启；

****performance.parallel-readdir****  
尽管文件/目录号保持不变，但目录列表的数量会随着卷/块数量的增加而变慢。 通过启用并行readdir卷选项，可以使目录列表的性能与卷中的节点/块的数量无关。因此，卷的增加不会降低目录列表的性能

****cluster.lookup-unhashed****  
 Lookup-unhash配置是在dht上生效的，是指在查找时候，如果在hash所在节点上没有找到相应文件的话，去所有节点上查找一遍

****cluster.lookup-optimize****  
 分布式xlator（DHT）在处理查找卷中不存在的条目时会有性能损失。因为DHT会试图在所有子卷中查找文件，所以这种查找代价很高，并且通常会减慢文件的创建速度。 这尤其会影响小文件的性能，其中大量文件被快速连续地添加/创建。 查找卷中不存在的条目的查找扇出行为可以通过在一个均衡过的卷中不进行相同的执行进行优化。

注意：在设置上述选项后，对于新创建的目录配置将立即生效。 对于现有的目录，在DHT对旧目录应用优化之前，需要重新平衡以确保卷处于平衡状态。

****performance.read-ahead****

read-ahead会顺序地预取一些块。当应用忙于处理一些数据的时候，GlusterFS能够预读下一批等待处理的数据。这样能够使的读取操作更加流畅和迅速。而且，工作起来像一个读的集合器一样（read-aggregator），也就是说，将大量的、零散的读取操作集合成少量的、大一些的读操作，这样，减小了网络和磁盘的负载

****performance.io-cache****

如果在client端被加载，能够帮助减小server的负载（如果client正在读一些文件，而且这个文件在两次读操作期间没有被修改）

****performance.quick-read****  
 该选项用来提高小文件读性能。通过网络对文件系统进行操作开销很大，因此，quick-read使用glusterfs内部get接口来一次执行多个posix系统调用open/read/close

****performance.stat-prefetch****  
 Stat-prefetch是一个新的性能translator，用于预取目录条目和统计信息。 这个translator的目的是优化像'ls -l'这样的操作，它会在每个目录条目上生成一个目录读取和统计调用。 这样的命令的总体性能现在会更好，因为从缓存的预取统计数据提供目录条目统计数据，而不是触发网络操作来获得该条目的统计信息

##### 在客户端上mount以上创建的卷

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

开机自动挂载：

#vi /etc/fstab

node1:test-volume /mntpoint glusterfs defaults,\_netdev,backupvolfile-server=node2 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））

#gluster volume add-brick test-volume replica 2 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；收缩卷需要先进行数据迁移，等待完成之后再确认删除。收缩后需要平衡（见（8））

#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 start

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.

##### 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。

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

##### 卷平衡

扩大或缩小卷（分别使用add-brick和remove-brick命令）后，需要重新平衡服务器之间的数据。扩大或收缩后创建的新目录将自动均匀分布。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

#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.

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

#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

##### 清理数据

删除卷之后，推荐格式化磁盘，否则若采用同样的方式建立卷，会访问到原来数据，这样造成数据不安全。

#### GlusterFS卷功能

##### 存储集群状态

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

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

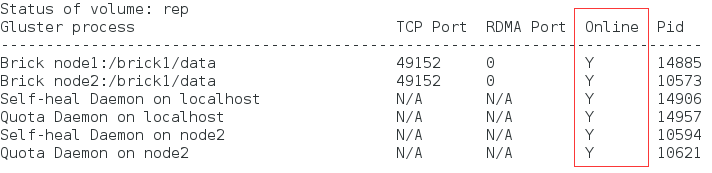


表 3 启动配额，启动恢复，brick全部上线

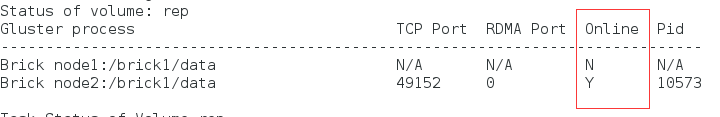


表 4 关闭配额，关闭恢复，brick1掉线

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）

# 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

移除配额：

# gluster volume quota test-volume remove /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/index/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/index/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: 完全修复，只能是手动触发；场景一般是副本模式下出现脑裂时，手动处理掉异常文件后使用full来修复。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

# 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

##### shard功能

Gluster对非常大的文件（超过一个brick）在某些使用场景有限制，因此提出一个新的解决方案shard，它不同于集群stripe，它被放在dht的上层，只将第一个块像普通文件一样存储，其余的块都存储在.shard目录中，以gfid和块索引来命名，（如.shard/gfid1.1 .shard/gfid1.2 ...）,文件的整个大小记录在第一个块的xattr中，这样做大文件可以正常分布，同时自修复时可以分布跨越更多服务器。再已经写入的文件如果调整切片大小，那么原来文件的属性不变，如果不想分片，建议创建新卷将其拷贝过去。

# gluster volume set test-volume features.shard on //开启

# gluster volume set test-volume features.shard-block-size xxMB //默认是64MB，不建议修改

##### tiering分层

分层功能可以利用ssd作为热层来缓存数据，例如：

以副本模式添加两块ssd作为卷test-volume的缓存层即热层（推荐）,也可以使用一块ssd。

# gluster volume tier test-volume attach replica 2 BRICK-NAME BRICK-NAME

设置文件降级和升级的百分比：分为低值和高值，当热层容量在低值下时，不需要升级或降级；当热层容量在低值上高值下时，数据将随机升级或降级；当热层容量在高值上时，数据升级停止降级增加；

# gluster volume set test-volume cluster.watermark-hi value //高值 默认是90

# gluster volume set test-volume cluster.watermark-low value //低值 默认是75

设置检查数据升级和降级的频率：

# gluster volume set test-volume cluster.tier-promote-frequency value //升级 默认是120s

# gluster volume set test-volume cluster.tier-demote-frequency value //降级 默认是3600s

设置文件读写频率，任何读或者写命中的次数低于此值的文件都将被是为冷文件并降级,为0 表示不考虑本参数

# gluster volume set test-volume cluster.read-freq-threshold value //默认是0

# gluster volume set test-volume cluster.write-freq-threshold value //默认是0

删除分层

# gluster volume tier test-volume detach start //开始降级数据

# gluster volume tier test-volume detach status //查看数据降级状态

# gluster volume tier test-volume detach commit //最终确认删除分层 y -确认

##### 快照snapshots

##### To**p**命令

Top命令可以看磁盘的一些性能、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调用，但不能用来获取集群的性能，例如一个双副本的卷，客户端写带宽是30MB/s，而top测出来每个brick有358MB/s

#### 6、配置高可用

##### 1、配置ctdb

（1）在每个节点上安装ctdb：

# yum install ctdb

（2）CTDB需要一个共享区域在创建lock，还需要创建共享目录,我们是在gluster的客户端做高可用，所以共享目录直接使用gluster的挂载目录下的子目录。假如挂载目录是/mnt：

# mkdir /mnt/ctdb

（3）然后编辑每个节点的ctdb配置文件：

# vi /etc/ctdb/ctdb.conf

CTDB\_RECOVERY\_LOCK=/mnt/ctdb/.ctdb\_lock

CTDB\_NODES=/etc/ctdb/nodes

CTDB\_PUBLIC\_ADDRESSES=/etc/ctdb/public\_addresses

CTDB\_LOGGING=file:/var/log/log.ctdb

CTDB\_DEBUGLEVEL=ERR

（4）然后在每个节点创建/etc/ctdb/nodes文件(真实ip)：

192.168.15.129

192.168.15.130

（5）然后在每个节点创建/etc/ctdb/public\_addresses文件（虚拟ip）,注意：public IP所指定的网卡名必须是本机拥有的网卡名。这是由于Centos7中，对于网卡名提供了不同的命名规则，默认是基于固件、拓扑、位置信息来分配命名.

192.168.15.141/24 eth0

192.168.15.142/24 eth0

（7）在每个节点上，设置ctdb服务为开机自启动，然后启动ctdb服务；

# systemctl enable ctdb.service

# systemctl start ctdb.service

（8）使用下列命令来确定服务是否已经正常运行起来；

# ctdb status

##### 2、配置NFS的高可用

由于集群NAS中所使用的集群文件系统是GlusterFS，所以关于nfs存在两种形态，即内核态nfs以及用户态nfs。其中内核态nfs指Linux，内核自带的nfs，用户态nfs指是GlusterFS自带的nfs服务。注：两种形态的nfs服务是互斥的。只推荐内核态nfs服务。配置内核态nfs服务的高可用：

a．修改修改ctdb配置文件，把关于ctdb配置中关于nfs的配置参数值改为yes。

# vi /etc/ctdb/ctdb.conf

CTDB\_NFS\_SKIP\_SHARE\_CHECK=yes

CTDB\_MANAGES\_NFS=yes

b．nfs的配置如下：

# vi /etc/exports

/mnt \*(rw,sync)

c．在每个节点上，停止nfs的服务，并关闭服务在操作系统启动时的自动启动，使用ctdb服务来控制nfs服务的启停，重启ctdb服务，命令如下；

# systemctl stop nfs-server.service

# systemctl disable nfs-server.service

# systemctl restart ctdb.service

d．这样一个高可用的内核态nfs共享服务就配置完成了，可在Linux系统下挂载访问，如下：

# mount -t nfs 192.168.15.xx:/mnt/nfs-share /mountpoint

##### 3、配置samba的高可用

（1）在每个客户端节点上编写Samba配置：

# vi /etc/samba/smb.conf

[gluster-share]

comment = gluster-share

path = /mnt

wirteable = yes

browseable = yes

guest ok = yes

public = yes

（2）修改ctdb配置文件，把关于ctdb关于Samba的配置参数值改为yes。

# vi /etc/ctdb/ctdb.conf

CTDB\_SAMBA\_SKIP\_SHARE\_CHECK=yes

CTDB\_MANAGES\_SAMBA=yes

（3）在每个节点上，停止samba的smb和nmb服务，并关闭服务在操作系统启动时的自动启动，使用ctdb服务来控制Samba服务的启停，重启ctdb服务，命令如下；

# systemctl stop smb.service nmb.service

# systemctl disable smb.service nmb.service

# systemctl restart ctdb.service

1. 这样一个高可用的CIFS共享服务就配置完成了，可以在Windows操作系统的电脑或者任何支持SMB/CIFS协议的电脑上通过访问[\\192.168.15.141](file:///\\\\192.168.1.201\\share) 或者[\\192.168.15.142](file:///\\\\192.168.1.201\\share) 使用该存储了。

#### 7、gluster常见问题解答

##### （1）Gluster需要占用哪些端口？

Gluster管理服务使用24007端口，Infiniband管理使用24008端口，每个brick进程占用一个端口。比如4个brick，使用24009-24012端口。

##### （2）创建Gluster资源池出问题？

首先，检查nslookup是否可以正确解析DNS和IP。其次，确认没有使用/etc/hosts直接定义主机名。虽然理论上没有问题，但集群规模一大很多管理员就会犯低级错误，浪费大量时间。再者，验证Gluster服务所需的24007端口是否可以连接(比如telnet)？Gluster其他命令是否可以成功执行？如果不能，Gluster服务很有可能没有启动

##### （3）如何检查Gluster服务是否运行？

可以使用如下命令检查Gluster服务状态：

service glusterd status

systemctl status glusterd.service

/etc/init.d/glusterd status

##### （4）无法在client端挂载(mount)Gluster卷？

检查网络连接是否正常，确认glusterd服务在所有节点上正常运行，确认所挂载volume处于启动状态；确认/etc/hosts正确配置了集群ip

##### （5）运行“glusterpeer probe“，不同节点输出结果可能不一致？

这个通常不是问题。每个节点输出显示其他节点信息，并不包括当前节点；不管在何处运行命令，节点的UUID在所有节点上都是相同和唯一的；输出状态通常显示“Peer in Cluster (Connected)“，这个值应该和/var/lib/glusterd/glusterd.info匹配

##### （6）数据传输过程中意外杀掉gluster服务进程？

所有数据都不会丢失。Glusterd进程仅用于集群管理，比如集群节点扩展、创建新卷和修改旧卷，以及卷的启停和客户端mount时信息获取。杀掉gluster服务进程，仅仅是一些集群管理操作无法进行，并不会造成数据丢失或不可访问。

##### （7）Gluster日志在系统什么位置？

日志都位于/var/log/glusterfs

##### （8）Gluster配置文件在系统什么位置？

3.3以上版本位于/var/lib/glusterd，

##### （9）Gluster系统异常，重启服务后问题依旧。

很有可能是某些服务进程处于僵死状态，使用ps -ax | grep glus命令查看。如果发出shutdown命令后，一些进程仍然处于运行状态，使用killall -9gluster{,d,fs,fsd}杀掉进程，或者硬重启系统。

##### （10）需要在每个节点都运行Gluster命令吗？

这个根据命令而定。一些命令只需要在Gluster集群中任意一个节点执行一次即可，比如“gluster volume create”，而例如“gluster peer status ”命令可以在每个节点独立多次执行。

三、总结

整个集群NAS主要由集群文件系统、高可用NAS集群、LVS负载集群三个逻辑部分组成，如图5所示：

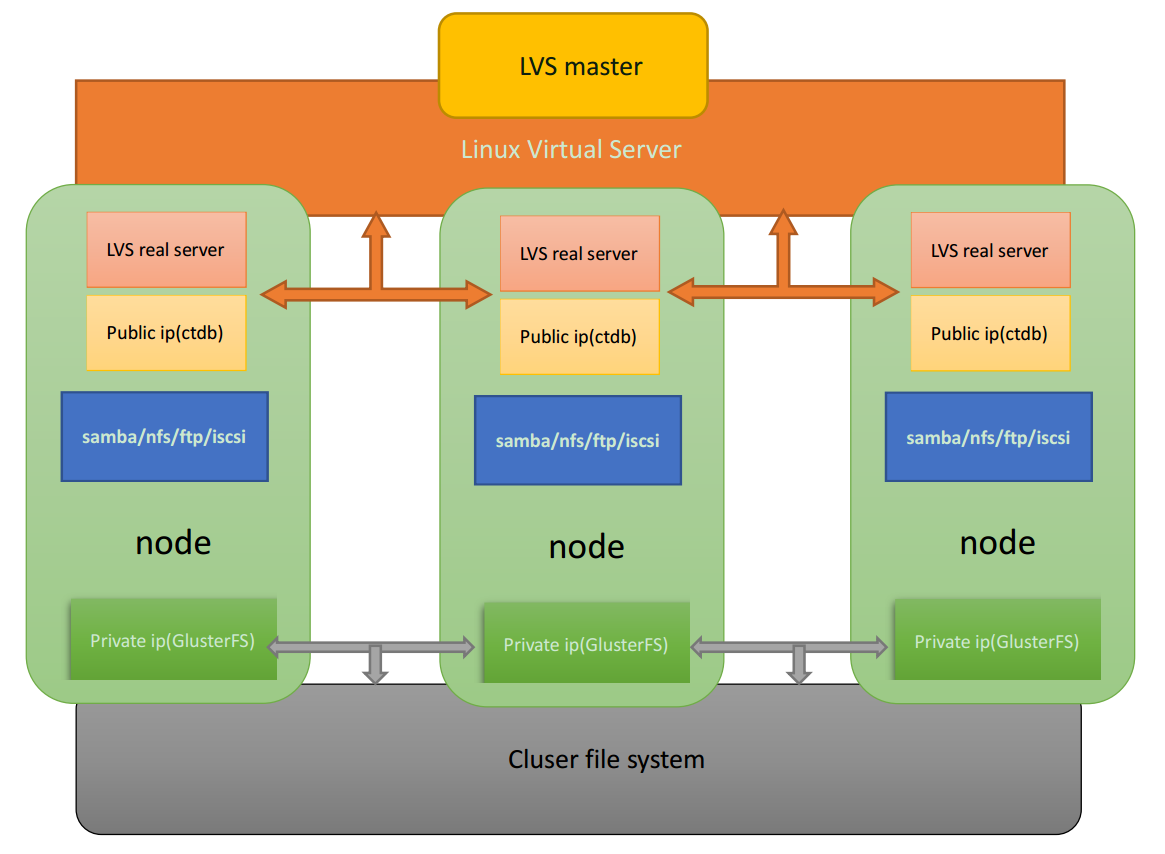


图5 集群NAS系统总体架构

集群文件系统使用GlusterFS分布式存储系统，它具有全局统一命名空间、高性能、高可用、高扩展等特点，它最大特点是采用无元数据服务设计，支持副本和纠删码高可用技术，基于Gluster存储可以实现磁盘级和服务器级的高可用。多个物理节点通过CTDB构建共享层的全活高可用，基于LVS构建自动负载均衡，基于标准的NFS/CIFS/HTTP/FTP/ISCSI等协议来提供数据访问服务，构建出了毫不逊色于商业存储系统的集群NAS存储系统，可以满足企业对于存储系统在高性能、高可靠性、高拓展性、跨平台性等方面的需求，而且由于使用的是开源软件、标准协议，那么也就意味着低成本，高性能，高通用性，相较于昂贵的商业存储产品，这一种存储解决方案对于企业来说也是一种不错的选择。