目录

**[CLUSTER](#_Toc7364_WPSOffice_Level1)** **[1](#_Toc7364_WPSOffice_Level1)**

**[Day 01 存储技术与应用 iSCSI技术应用 、 udev配置 NFS网络文件系统 、 Multipath多路径 、 NFS网络文件系统 、 udev配置](#_Toc31283_WPSOffice_Level1)** **[1](#_Toc31283_WPSOffice_Level1)**

**[一、 关于硬盘的二三事](#_Toc30843_WPSOffice_Level1)** **[1](#_Toc30843_WPSOffice_Level1)**

**[二、 准备两台虚拟机](#_Toc9771_WPSOffice_Level1)** **[1](#_Toc9771_WPSOffice_Level1)**

[1. 在proxy上执行](#_Toc31283_WPSOffice_Level2) [1](#_Toc31283_WPSOffice_Level2)

[2. 客户端client](#_Toc30843_WPSOffice_Level2) [2](#_Toc30843_WPSOffice_Level2)

**[三、 附加课外实验：多台FTP或者http主机使用共享存储](#_Toc9263_WPSOffice_Level1)** **[3](#_Toc9263_WPSOffice_Level1)**

[1. 这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备](#_Toc9263_WPSOffice_Level2) [3](#_Toc9263_WPSOffice_Level2)

**[四、 部署multipath多路径环境](#_Toc15939_WPSOffice_Level1)** **[4](#_Toc15939_WPSOffice_Level1)**

[1. 通过multipath,实现以下目标](#_Toc15939_WPSOffice_Level2) [4](#_Toc15939_WPSOffice_Level2)

**[五、 配置并访问NFS共享](#_Toc1066_WPSOffice_Level1)** **[6](#_Toc1066_WPSOffice_Level1)**

[1. 服务器利用NFS机制发布2个共享目录，要求如下：](#_Toc1066_WPSOffice_Level2) [6](#_Toc1066_WPSOffice_Level2)

[2. 配置proxy的nfs文件](#_Toc24255_WPSOffice_Level2) [6](#_Toc24255_WPSOffice_Level2)

[3. 客户端查看](#_Toc20502_WPSOffice_Level2) [6](#_Toc20502_WPSOffice_Level2)

[4. NFS依赖服务](#_Toc9983_WPSOffice_Level2) [6](#_Toc9983_WPSOffice_Level2)

**[六、 编写udev规则](#_Toc24255_WPSOffice_Level1)** **[7](#_Toc24255_WPSOffice_Level1)**

[1. 编写udev规则，实现以下目标：](#_Toc22249_WPSOffice_Level2) [7](#_Toc22249_WPSOffice_Level2)

[2. udev(2.6) 3.10](#_Toc18670_WPSOffice_Level2) [7](#_Toc18670_WPSOffice_Level2)

**[Day02 集群及LVS简介 LVS-NAT集群 LVS-DR集群](#_Toc20502_WPSOffice_Level1)** **[9](#_Toc20502_WPSOffice_Level1)**

**[一、 集群及LVS简介](#_Toc9983_WPSOffice_Level1)** **[9](#_Toc9983_WPSOffice_Level1)**

[1. 集群简介](#_Toc20101_WPSOffice_Level2) [9](#_Toc20101_WPSOffice_Level2)

[2. LVS概述](#_Toc18430_WPSOffice_Level2) [10](#_Toc18430_WPSOffice_Level2)

**[二、 LVS-NAT集群](#_Toc22249_WPSOffice_Level1)** **[13](#_Toc22249_WPSOffice_Level1)**

[1. 软件安装](#_Toc9541_WPSOffice_Level2) [13](#_Toc9541_WPSOffice_Level2)

[2. LVS-NAT案例](#_Toc14091_WPSOffice_Level2) [15](#_Toc14091_WPSOffice_Level2)

**[三、 部署LVS-DR集群](#_Toc18670_WPSOffice_Level1)** **[17](#_Toc18670_WPSOffice_Level1)**

[1. 问题描述](#_Toc12506_WPSOffice_Level2) [17](#_Toc12506_WPSOffice_Level2)

[2. 方案](#_Toc29240_WPSOffice_Level2) [17](#_Toc29240_WPSOffice_Level2)

**[Day03 Keepalived热备 Keepalived+LVS 、 HAProxy服务器](#_Toc20101_WPSOffice_Level1)** **[21](#_Toc20101_WPSOffice_Level1)**

**[一、 nginx与LVS调度器区别](#_Toc18430_WPSOffice_Level1)** **[21](#_Toc18430_WPSOffice_Level1)**

[1. nginx代理(我帮你去访问)](#_Toc11564_WPSOffice_Level2) [21](#_Toc11564_WPSOffice_Level2)

[2. LVS调度(转发数据包) LVS-NAT LVS-DR LVS-TUN](#_Toc7388_WPSOffice_Level2) [21](#_Toc7388_WPSOffice_Level2)

**[二、 Keepalived热备](#_Toc9541_WPSOffice_Level1)** **[21](#_Toc9541_WPSOffice_Level1)**

[1. Keepalived概述](#_Toc21269_WPSOffice_Level2) [21](#_Toc21269_WPSOffice_Level2)

[2. Keepalived服务](#_Toc11197_WPSOffice_Level2) [22](#_Toc11197_WPSOffice_Level2)

**[三、 Keepalived+LVS服务器](#_Toc14091_WPSOffice_Level1)** **[24](#_Toc14091_WPSOffice_Level1)**

[1. 问题](#_Toc20023_WPSOffice_Level2) [24](#_Toc20023_WPSOffice_Level2)

[2. 操作](#_Toc17956_WPSOffice_Level2) [24](#_Toc17956_WPSOffice_Level2)

[3. 部署Keepalived实现LVS-DR模式调度器的高可用](#_Toc21051_WPSOffice_Level2) [25](#_Toc21051_WPSOffice_Level2)

**[四、 配置HAProxy负载均衡集群](#_Toc12506_WPSOffice_Level1)** **[26](#_Toc12506_WPSOffice_Level1)**

[1. 调度器宏观对比](#_Toc23697_WPSOffice_Level2) [26](#_Toc23697_WPSOffice_Level2)

[2. 准备4台Linux服务器,两台做web服务器,1台安装HAproxy,1台做客户端,实现以下功能](#_Toc21296_WPSOffice_Level2) [27](#_Toc21296_WPSOffice_Level2)

[3. 清除前面的实验环境](#_Toc2587_WPSOffice_Level2) [27](#_Toc2587_WPSOffice_Level2)

[4. 配置后端服务器](#_Toc16888_WPSOffice_Level2) [27](#_Toc16888_WPSOffice_Level2)

[5. 两种方式定义集群:](#_Toc15803_WPSOffice_Level2) [29](#_Toc15803_WPSOffice_Level2)

**[Day04 Ceph概述 部署Ceph集群 Ceph块存储](#_Toc29240_WPSOffice_Level1)** **[30](#_Toc29240_WPSOffice_Level1)**

**[一、 Ceph概述](#_Toc11564_WPSOffice_Level1)** **[30](#_Toc11564_WPSOffice_Level1)**

[1. 基础知识](#_Toc29906_WPSOffice_Level2) [30](#_Toc29906_WPSOffice_Level2)

[2. 实验环境准备](#_Toc21275_WPSOffice_Level2) [33](#_Toc21275_WPSOffice_Level2)

**[二、 部署Ceph集群](#_Toc7388_WPSOffice_Level1)** **[35](#_Toc7388_WPSOffice_Level1)**

[1. 准备部署环境](#_Toc25991_WPSOffice_Level2) [35](#_Toc25991_WPSOffice_Level2)

[2. 部署存储集群](#_Toc16025_WPSOffice_Level2) [35](#_Toc16025_WPSOffice_Level2)

**[三、 Ceph块存储](#_Toc21269_WPSOffice_Level1)** **[38](#_Toc21269_WPSOffice_Level1)**

[1. 概述](#_Toc31183_WPSOffice_Level2) [38](#_Toc31183_WPSOffice_Level2)

[2. 块存储集群](#_Toc20124_WPSOffice_Level2) [39](#_Toc20124_WPSOffice_Level2)

**[四、 总结和答疑](#_Toc11197_WPSOffice_Level1)** **[43](#_Toc11197_WPSOffice_Level1)**

**[Day05 块存储应用案例 分布式文件系统 对象存储](#_Toc20023_WPSOffice_Level1)** **[44](#_Toc20023_WPSOffice_Level1)**

**[一、 块存储应用案例](#_Toc17956_WPSOffice_Level1)** **[44](#_Toc17956_WPSOffice_Level1)**

[1. 准备实验环境](#_Toc28631_WPSOffice_Level2) [44](#_Toc28631_WPSOffice_Level2)

[2. 创建KVM虚拟机](#_Toc21622_WPSOffice_Level2) [44](#_Toc21622_WPSOffice_Level2)

**[二、 分布式文件系统](#_Toc21051_WPSOffice_Level1)** **[45](#_Toc21051_WPSOffice_Level1)**

[1. 基础知识](#_Toc6168_WPSOffice_Level2) [45](#_Toc6168_WPSOffice_Level2)

[2. 元数据服务器](#_Toc22882_WPSOffice_Level2) [45](#_Toc22882_WPSOffice_Level2)

[3. 文件系统服务器](#_Toc3814_WPSOffice_Level2) [45](#_Toc3814_WPSOffice_Level2)

**[三、 对象存储](#_Toc23697_WPSOffice_Level1)** **[45](#_Toc23697_WPSOffice_Level1)**

[1. 概述](#_Toc26718_WPSOffice_Level2) [45](#_Toc26718_WPSOffice_Level2)

[2. 对象存储](#_Toc31576_WPSOffice_Level2) [45](#_Toc31576_WPSOffice_Level2)

**[四、 总结和答疑](#_Toc21296_WPSOffice_Level1)** **[46](#_Toc21296_WPSOffice_Level1)**

CLUSTER

Day 01 存储技术与应用 iSCSI技术应用 、 udev配置 NFS网络文件系统 、 Multipath多路径 、 NFS网络文件系统 、 udev配置

1. 关于硬盘的二三事

DAS(直连) 硬盘接口 IDE,SATA,SAS,SSD

传统网络存储:

NAS:文件系统存储(ext3,ext3,ntfs) samba,nfs

SAN:块共享 iscsi DELL(EMC(vmware))

FC:光纤 做网盘(SAN)时所有设备通过光纤 网卡,交换机,路由器

分布式存储ceph

阿里(去IOE) IBM Oracle EMC

源码 --> 二次开发 --> nginx ==> tengine

1. 准备两台虚拟机
2. 在proxy上执行

[root@proxy ~]# parted /dev/vdb mklabel gpt

信息: You may need to update /etc/fstab.

[root@proxy ~]# parted /dev/vdb mkpart primary 1 100%

信息: You may need to update /etc/fstab.

# targetcli

/> backstores/block create store /dev/vdb1

/> ls

/> /iscsi create iqn.2018-01.cn.tedu:server1

/> /iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1

/> /iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/store

/> saveconfig

/> exit

backstore:

1.store /dev/vbd1 20G

2.db /dev/vdc 80G

3.www

iscsi:

iqn-2018

acl 字串(iqn)

lun store

iqn.2019

acl

lun db

iqn-2020

acl

lun

1. 客户端client
2. 客户端安装软件并启动服务

# yum install iscsi-initiator-utils -y

1. 设置本机的iqn名称

# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1

1. 发现远程target存储

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5 --discover

# systemctl restart iscsi

1. 查看iscsi的加载是否成功

# lsblk

出现sda盘

1. 对sda盘进行分区和格式化,并挂载进行使用

# parted /dev/sda mklabel gpt

# parted /dev/sda mkpart 1 100%

# lsblk

/\*\* sda1出现 \*\*/

# mkfs.xfs /dev/sda1

# mount /dev/sda1 /mnt

1. 附加课外实验：多台FTP或者http主机使用共享存储
2. 这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备

web1(ftp) /var/ftp

iscsi -- sda

mount /sda1 /var/ftp

web2(http)

iscsi -- sda

mount /dev/sda1 /var/www/html

iscsi块共享:

不能同时多人使用

1. 修改vsftpd配置文件，开启匿名上传功能。将下面2行默认的注释行打开。

#anon\_upload\_enable=YES

#anon\_mkdir\_write\_enable=YES

1. 部署multipath多路径环境
2. 通过multipath,实现以下目标

· 在共享存储服务器上配置ISCSI,为应用服务器共享存储空间

· 应用服务器上配置iSCSI，发现远程共享存储

· 应用服务器上配置Multipath，将相同的共享存储映射为同一个名称

web1(iscsi客户端)

eth0

eth1

双网卡访问

· proxy(iscsi服务端)

1. client上操作

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5:3260 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5:3260 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

# systemctl restart iscsi

# lsblk

sda 8:0 0 20G 0 disk

└─sda1 8:1 0 20G 0 part

sdb 8:16 0 20G 0 disk

└─sdb1 8:17 0 20G 0 part

1. 使用多路径将两个硬盘合并为一个
2. 安装多路径软件包

# yum install device-mapper-multipath -y

1. 生成配置文件

# cd /usr/share/doc/device-mapper-multipath-0.4.9/

# cp multipath.conf /etc/

1. 查看两块硬盘的id号

# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sda

3600140541fc17254192472bae0438e5e

# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sdb

3600140541fc17254192472bae0438e5e

1. 修改配置文件

# vim /etc/multipath.conf

跳转至60行

multipaths {

multipath {

wwid 3600140541fc17254192472bae0438e5e

alias my\_scsi

}

}

1. 查看多路径设备是否正常

# multipath -ll

# multipath -rr

my\_scsi (3600140541fc17254192472bae0438e5e) dm-2 LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=rw

|-+- policy='service-time 0' prio=1 status=active

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=enabled

`- 3:0:0:0 sdb 8:16 active ready running

· 关闭proxy的eth1网卡

[root@proxy ~]# ifdown eth1

# multipath -ll

my\_scsi (3600140541fc17254192472bae0438e5e) dm-2 LIO-ORG ,store

size=20G features='0' hwhandler='0' wp=rw

|-+- policy='service-time 0' prio=1 status=active

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=enabled

`- 3:0:0:0 sdb 8:16 failed ready running

· 挂载特殊硬盘my\_scsi

# mount /dev/mapper/my\_scsi1 /mnt

1. 为了防止和ceph存储冲突,需要取消当前实验

1.umount

# umount /mnt

2.stop multipath.service

# systemctl stop multipathd.service

3.logout

# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.4.5:3260 --logout

# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --logout

1. 配置并访问NFS共享
2. 服务器利用NFS机制发布2个共享目录，要求如下：

· 将目录/root共享给192.168.2.100，客户机的root用户有权限写入

· 将/usr/src目录共享给192.168.2.0/24网段，只开放读取权限

1. 配置proxy的nfs文件

# vim /etc/exports

/root 192.168.2.100(rw,no\_root\_squash)

/usr/src 192.168.2.0/24(ro)

/\*\*

\* 用root访问nfs时,默认会被降级,需要在选项中增加no\_root\_squash

\*\*/

1. 客户端查看

# showmount -e 192.168.2.5

/usr/src 192.168.2.0/24

/root 192.168.2.100

1. NFS依赖服务

# systemctl restart rpcbind;systemctl enabled rpcbind

/\*\*

\* NFS:随机端口 如1098

\* rpcbind:111端口

\* 注册端口:

\* nfs:1098

\*\*/

1. 编写udev规则
2. 编写udev规则，实现以下目标：

· 当插入一个U盘时，该U盘自动出现一个链接称为udisk

· U盘上的第1个分区名称为udisk1，以此类推

· 终端上出现提示信息”udisk plugged in”

1. udev(2.6) 3.10

udev设备管理

所有设备都存放在/dev/目录(静态)

udev动态管理

udev动态设备管理/sys /dev

对于Linux kernel 2.6及更新的操作系统版本会将设备的相关信息动态写入/sys文件系统中，而udev程序可以通过读取这些设备系信息，并根据自己的udev规则进行设备管理器，实现如下功能:

· 处理设备命名 ENV{设备属性}=="属性值" NAME="定义设备名称"

· 决定要创建哪些设备文件或链接 SYMLINK+="链接"

· 决定如何设置属性 MODE="权限" OWNER="属主" GROUP="属组"

· 决定触发哪些事件 RUN="命令"

/\*\*

\* ==判断 !=

\* =赋值

\* +=原有基础上的赋值

\* RUN+=

\* :=不允许被替换的规则

\* 判断条件1,判断条件2,判断条件3

\* ENV{品牌}==xx,ENV{序列号}==xx,ENV{条件}==值,RUN+=

\* KERNEL=="sd[a-z]1" 判断设备的内核名称

\*\*/

udev默认规则存放在/etc/udev/rules.d目录下，通过修改此目录下的规则实现设备的命名、属性、链接文件等。

1）查看设备属性

加载USB设备的同时实时查看设备的相关属性，可以使用monitor指令。

# udevadm monitor --property

ACTION=add

如果设备已经加载则无法使用monitor查看相关属性。可以使用下面的命令查看设备属性。

# udevadm info --query=path –name=/dev/sdb

# udevadm info --query=property –path=/block/sdb

# vim /etc/udev/rules.d/70-usb.rules

ENV{ID\_VENDOR}=="JMicron",ACTION=="add",RUN+="/usr/bin/systemctl start httpd"

ENV{ID\_VENDOR}=="JMicron",ACTION=="remove",RUN+="/usr/bin/systemctl stop httpd"

Day02 集群及LVS简介 LVS-NAT集群 LVS-DR集群

1. 集群及LVS简介
2. 集群简介

1) 什么是集群

· 高速网互联

· 多服务器集中起来提供一种服务,在客户端看来就像是只有一个服务器

· 节省成本,低成本下获得在性能,可靠性和灵活性方面的相对较高的收益

· 任务调度时集群系统中的核心技术

2) 集群目的

· 提高性能

· 降低成本

· 提高扩展性

· 增强可靠性

3) 集群分类

· 高性能计算集群HPC

- 通过以集群开发的并行应用程序,解决复杂的科学问题

· 负载均衡(LB)集群

- 客户端负载在计算机集群中尽可能平均分摊

· 高可用(HA)集群

- 避免单点故障,当一个系统发生故障时,可以快速迁移

1. LVS概述

1) LVS项目介绍

· Linux虚拟服务器(LVS)是章文嵩在国防科大就读博士期间创建的

· LVs可以实现高可用的,可伸缩的Web,Mail,Cache和Media等网络服务

· 最终目标时利用Linux操作系统和LVS集群软件实现一个高可用,高性能,低成本的服务器应用集群

/\*\*

\* LVS:性能高,功能少,不支持正则

\* nginx:性能中,功能多,正则

\*\*/

1. LVS集群组成

· 前端:负载均衡层

- 由一台或多台负载调度器构成

· 中间:服务器群组层

- 由一组实际运行应用服务的服务器组成

· 底端:数据共享存储层

- 提供共享存储空间的存储区域

/\*\*

\* 用keeplived来保证LVS的高可用

\*\*/

3) LVS术语

· Director Server:调度服务器

- 将负载分发到Real Server的服务器

· Real Server:真实服务器

- 真正提供应用服务的服务器

· VIP[virtual IP]:虚拟IP地址

- Director Server公布给用户访问的虚拟IP地址

· RIP[real IP]:真实IP地址

- 集群节点[Real Server]上使用的IP地址

· DIP:调度器连接节点服务器的IP地址

· CIP[Client IP]:客户端IP

4) LVS工作模式

· NAT模式

- 通过网络地址转换实现的虚拟服务器;

- 用户通过互联网访问VIP,LVS可以看作一个路由器;

- 要求从哪里进,从哪里出;

- 与路由器的区别:路由器只是单纯把内网服务器发布到公网,而LVS可以实现负载均衡;

- 缺点:LVS的负荷高,调度器性能会成为整个集群的瓶颈;

- 适合数据量小的集群环境;

· TUN[隧道]模式

- 用户访问北京的Director Server;

- Real Server在上海;

- 通过LVS转发请求寻找Real Server服务;

· DR模式

- 直接使用路由技术实现虚拟服务器;

- 路由器将数据转给LVS,LVS转给服务器;

- 服务器将处理结果通过路由器传输给用户;

- 适合数据量较大的集群环境;

- 节点服务器需要配置VIP,注意MAC地址广播;

5) 负载均衡调度算法

· LVS目前实现了10种调度算法

· 常用调度算法有4种

- 轮询(Round Robin)

\* 将客户端请求平均分发到Real Server

- 加权轮询(Weight Round Robin)

\* 根据Real Server权重值进行轮询调度

- 最少连接(Least Connections)

\* 选择连接数最少的服务器

- 加权最少连接(Weight Least Connections)

\* 根据Real Server权重值,选择连接数最少的服务器

- 源地址散列(Source Hashing)

\* 根据请求的目标IP地址,作为散列键(Hash Key)从静态分配的散列表找出对应的服务器

- 其他调度算法

\* 给予局部性的最少链接

\* 带复制的基于局部性最少链接

\* 目标地址的三列(Destination Hashing)

\* 最短的期望的延迟

\* 最少队列调度

1. LVS-NAT集群
2. 软件安装

1) 安装前准备

· LVS的IP负载均衡技术是通过IPVS模块实现的

· IPVS模块已成为Linux组成部分

2) 安装ipvsadm

3) ipvsadm用法

ipvsadm -A 添加虚拟服务器

ipvsadm -E 修改虚拟服务器

ipvsadm -D 删除虚拟服务器

ipvsadm -C 清空所有

ipvsadm -a 添加真实服务器

ipvsadm -e 修改真实服务器

ipvsadm -d 删除真实服务器

ipvsadm -L 查看

-s [rr|wrr|lc|wlc] 指定负载调度算法

· 创建虚拟服务器

# ipvsadm -A -t|u 192.168.4.5:80 -s [算法]

-A 添加虚拟服务器

-t 协议为tcp

-u 协议为udp

-s 执行负载调度算法

· 添加/删除服务器节点

# ipvsadm -a -t|u 192.168.4.5:80 -r 192.168.2.100 [-g|i|m] [-w 权重]

-a 添加真实服务器

-d 删除真实服务器

-r 指定真实服务器的地址

-m 使用NAT模式

-g 使用DR模式 /\* 默认模式 \*/

-i 使用TUN模式

-w 为节点服务器设置权重,,默认为1

· 查看IPVS

# ipvsadm -Ln

· 永久保存所有规则

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

# cat /etc/sysconfig/ipvsadm

· 清除无关规则

# ipvsadm -C

# rm -f /etc/sysconfig/ipvsadm

Example:

/\*\* ipvsadm选项用法

\* -A添加集群

\* -a添加服务器

\*\*/

# ipvsadm -A -t 192.168.4.5:80 -s rr

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

-> 192.168.2.100:80 Route 1 0 0

1. LVS-NAT案例

· 使用LVS实现NAT模式的集群调度服务器，为用户提供Web服务：

\* 集群对外公网IP地址为192.168.4.5

\* 调度器内网IP地址为192.168.2.5

\* 真实Web服务器地址分别为192.168.2.100、192.168.2.200

\* 使用加权轮询调度算法，真实服务器权重分别为1和2

· 机器配备:

\* client 192.168.4.10/24

\* proxy 192.168.4.5/24 192.168.2.5/24

\* web1 192.168.2.100/24

\* web2 192.168.2.200/24

/\*\* 注意: web1 和 web2 的eth0网卡需要关闭 \*\*/

\* 服务器节点配置(web1/2):

1) 设置Web服务器

2) 启动Web服务

3) 关闭防火墙与SELinux

# systmctl stop firewalld

# setenforce 0

\* 调度服务器配置(proxy)

1) 确认调度器的路由转发功能

# echo 1 > /proc/sys/net/ipv4/ip\_forward

# cat /proc/sys/net/ipv4/ip\_forward

1

/\*\* 上面为临时配置 \*\*/

# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

/\*\* 这个是永久配置,可以用sysctl -p命令查看规则 \*\*/

2) 创建集群服务器

# yum -y install ipvsadm

# ipvsadm -A -t 192.168.4.5:80 -s wrr

3) 添加真实服务器

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80 -w 1 -m

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200:80 -w 1 -m

4) 查看规则列表,并保存规则

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

\* 客户端测试(client)

1) 利用自带的curl浏览器反复访问Dicretor Server(proxy)

# curl http://192.168.4.5

1. 部署LVS-DR集群
2. 问题描述

· 使用LVS实现DR模式的集群调度服务器，为用户提供Web服务：

\* 客户端IP地址为192.168.4.10

\* LVS调度器VIP地址为192.168.4.15

\* LVS调度器DIP地址设置为192.168.4.5

\* 真实Web服务器地址分别为192.168.4.100、192.168.4.200

\* 使用加权轮询调度算法，web1的权重为1，web2的权重为2

\* 说明：

\* CIP是客户端的IP地址；

\* VIP是对客户端提供服务的IP地址；

\* RIP是后端服务器的真实IP地址；

1. 方案

· 使用4台虚拟机,1台作为Director调度器,2台作为Real Server

\* client eth0(192.168.4.10/24)

\* proxy eth0(192.168.4.5/24) eth0:0(192.168.4.15/24)

\* web1 eth0(192.168.4.100/24) lo:0(192.168.4.15/32)

\* web2 eth0(192.168.4.200/24) lo:0(192.168.4.15/32)

/\*\*

\* 注意:

\* 1.web1/web2的子网掩码必须是32位

\* 2.VIP必须配在虚拟网卡上

\*\*/

\* DIP是调度器与后端服务器通信的IP地址（VIP必须配置在虚拟接口）。

· proxy配置:

# cp ifcfg-eth0 ifcfg-eth0:0

# vim ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

DEFROUTE=yes

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.15

PREFIX=24

# systemctl restart network

· web1配置:

# cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

# vim /etc/sysconfig/network-scripts/ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.15

NETMASK=255.255.255.255

NETWORK=192.168.4.15

BROADCAST=192.168.4.15

ONBOOT=yes

NAME=lo:0

· 防止地址冲突的问题：

\* 这里因为web1也配置与代理一样的VIP地址，默认肯定会出现地址冲突；

\* sysctl.conf文件写入这下面四行的主要目的就是访问192.168.4.15的数据包，只有调度器会响应，其他主机都不做任何响应，这样防止地址冲突的问题。

# vim /etc/sysctl.conf

\* 手动写入如下4行内容

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_announce = 2

\* 有arp广播问谁是192.168.4.15时，本机忽略该ARP广播，不做任何回应

\* 本机不要向外宣告自己的lo回环地址是192.168.4.15

# sysctl -p

/\*\* 重启网络服务，设置防火墙与SELinux \*\*/

· proxy部署调度器:

# yum -y install ipvsadm

# ipvsadm -C

/\*\* 1.清空规则 \*\*/

# ipvsadm -A -t 192.168.4.15:80 -s wrr

/\*\* 2.添加集群服务器 \*\*/

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 2

/\*\* 3.添加Real Server \*\*/

# ipvsadm -Ln

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

/\*\* 4.查看规则并保存 \*\*/

· 客户端测试

客户端使用curl命令反复连接http://192.168.4.15,查看访问的页面是否会轮询到不同的后端真实服务器;

扩展知识:默认LVS不带健康检查功能,需要自己手动编写动态检测脚本,实现该功能:(参考脚本如下,仅供参考)

# vim check.sh

#!/bin/bash

VIP=192.168.4.15:80

RIP1=192.168.4.100

RIP2=192.168.4.200

while :

do

for IP in $RIP1 $RIP2

do

curl -s http://$IP &>/dev/null

if [ $? -eq 0 ];then

ipvsadm -Ln | grep -q $IP || ipvsadm -a -t $VIP -r $IP

else

ipvsadm -Ln | grep -q $IP && ipvsadm -d -t $VIP -r $IP

fi

done

sleep 1

done

Day03 Keepalived热备 Keepalived+LVS 、 HAProxy服务器

1. nginx与LVS调度器区别
2. nginx代理(我帮你去访问)
3. LVS调度(转发数据包) LVS-NAT LVS-DR LVS-TUN

# yum install ipvsadm -y

# ipvsadm -A -t|u VIP:port -s wrr

# ipvsadm -a -t|u VIP:port -r RIP [-g|m|i]

# ipvsadm -E|D|C /\*\* VIP \*\*/

# ipvsadm -e|d /\*\* RIP \*\*/

# ipvsadm -Ln /\*\* 查看 \*\*/

\* NAT转发一定要有网关,且RIP不能和client在同网段中

/\*\*

\* Keepalived是给LVS写的

\* 1.自动配置LVS规则,做健康检查

\* 2.Keepalived学习了路由器上的功能VRRP HSRP路由热备

\*\*/

1. Keepalived热备
2. Keepalived概述
3. Keepalived概述

· 调度器出现单点故障,如何解决?

· Keepalived实现了高可用集群

· Keepalived最初是为LVS设计的,专门监控各服务器节点的状态

· Keepalived后来加入了VRRP功能,防止单点故障

1. Keepalived运行原理

· Keepalived检测每个服务器节点状态

· 服务器节点异常或工作出现故障,Keepalived将故障节点从集群系统中剔除

· 故障节点恢复后,Keepalived再将其加入到集群系统中

· 所有工作自动完成,无需人工干预

1. Keepalived服务

/\*\*

\* 准备三台 Linux服务器,两台做Web服务器,并部署Keepalived高可用软件,一台作为客户端

\* 使用Keepalived实现web服务器的高可用

\* proxy服务器的IP地址为192.168.4.5

\* web服务器IP地址分别为192.168.4.100和192.168.4.200

\* web服务器的浮动VIP为192.168.4.80

\* 客户端通过访问VIP地址访问web页面

\*\*/

1. Keepalived安装

/\*\* web1和web2 \*\*/

# yum install keepalived

/\*\* 装包 \*\*/

# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

admin@tarena.com.cn //设置报警收件人邮箱

}

notification\_email\_from ka@localhost //设置发件人

smtp\_server 127.0.0.1 //定义邮件服务器

smtp\_connect\_timeout 30

router\_id web1 //设置路由ID号（实验需要修改）

}

vrrp\_instance VI\_1 {

state MASTER //主服务器为MASTER（备服务器需要修改为BACKUP）

interface eth0 //定义网络接口

virtual\_router\_id 50 //主备服务器VRID号必须一致

priority 100 //服务器优先级,优先级高优先获取VIP（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { 192.168.4.80 } //谁是主服务器谁获得该VIP（实验需要修改）

}

/\*\* 配置 \*\*/

# systemctl restart keepalived && iptables -F

/\*\* 起服务 \*\*/

\* 当web1的keepalived服务停止时,192.168.4.80会自动切换到web2上

/\*\* 注意

\* keepalived只监控服务本身,当网卡接口停止时,需要停止keepalived服务VIP才会切换

\*\*/

1. Keepalived+LVS服务器
2. 问题

1) 使用Keepalived为LVS调度器提供高可用功能,防止调度器单点故障,为用户提供Web服务

· LVS1调度器真实IP地址为192.168.4.5

· LVS2调度器真实IP地址为192.168.4.6

· 服务器VIP地址设置为192.168.4.100,192.168.4.200

· 使用加权轮询调度算法,真实iweb服务器权重不同

2) 机器配备:

client eth0(192.168.4.10/24)

proxy1 eth0(192.168.4.5/24)

proxy2 eth0(192.168.4.6/24)

web1 eth0(192.168.4.100/24)

web2 eth0(192.168.4.200/24)

1. 操作

1) 设置web1服务器的网络参数

2) 为web1配置VIP地址lo:0(子网掩码必须是32)

3) 重启网络服务,设置防火墙与SELinux

4) 给web2重复上面操作

5) 配置proxy1与proxy2网络参数(VIP由keepalived自动配置)

6) 调度器安装Keepalived和ipvsadm

1. 部署Keepalived实现LVS-DR模式调度器的高可用

1) LVS1调度器设置Keepalived，并启动服务

/\*\* 上接VIP \*\*/

virtual\_server 192.168.4.15 80 { //设置ipvsadm的VIP规则（实验需要修改）

delay\_loop 6

lb\_algo wrr //设置LVS调度算法为WRR

lb\_kind DR //设置LVS的模式为DR

#persistence\_timeout 50

#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器

protocol TCP

real\_server 192.168.4.100 80 { //设置后端web服务器真实IP（实验需要修改）

weight 1 //设置权重为1

TCP\_CHECK { //对后台real\_server做健康检查

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 { //设置后端web服务器真实IP（实验需要修改）

weight 2 //设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

/\*\* 健康检查种类 \*\*/

real server {

TCP\_CHECK{

connect\_timeout 3 /\*\* 超时时间 \*\*/

nb\_get\_retry 3 /\*\* 尝试3次 \*\*/

delay\_before\_retry 3 /\*\* 每3秒重试一次 \*\*/

} /\*\* 以端口可用性为依据 \*\*/

HTTP\_CHECK{} /\*\* 以网页为依据 \*\*/

SSL\_GET{

path /

digest md5sum /\*\* 以md5码为验证 \*\*/

} /\*\* 以网页为依据 \*\*/

}

1. 配置HAProxy负载均衡集群
2. 调度器宏观对比
3. 调度器种类Nginx,LVS,HAProxy,F5 big-ip

速度:F5 > LVS > HAProxy > Nginx

\* LVS 4层调度,不支持7层

\* HAProxy 有一定的功能,速度还不错,对正则的支持不如Nginx

HAProxy既能做4层调度,也能做7层调度

\* Nginx 4,7层调度器

4.9以后支持4层调度,但还不够成熟

1. 准备4台Linux服务器,两台做web服务器,1台安装HAproxy,1台做客户端,实现以下功能

1) 客户端访问HAProxy,HAProxy分发请求到后端Real Server

2) 开启HAProxy监控页面,及时查看调度器状态

3) 设置HAProxy为开机启动

1. 清除前面的实验环境

· web1/web2:

# rm -f network/ifcfg-lo:0

# vim /etc/sysctl.conf

# ifdown eth0

# systemctl restart network

· proxy1/proxy2

# systemctl stop keepalived.service

1. 配置后端服务器

· 配置web1/2的httpd

· 在proxy1上安装HAProxy

# yum install haproxy -y

· 修改配置文件

# vim /etc/haproxy/haproxy.cfg

global

log 127.0.0.1 local2 ###[err warning info debug]

chroot /usr/local/haproxy

pidfile /var/run/haproxy.pid ###haproxy的pid存放路径

maxconn 4000 ###最大连接数，默认4000

user haproxy

group haproxy

daemon ###创建1个进程进入deamon模式运行

defaults

mode http ###默认的模式mode { tcp|http|health } log global ###采用全局定义的日志

option dontlognull ###不记录健康检查的日志信息

option httpclose ###每次请求完毕后主动关闭http通道

option httplog ###日志类别http日志格式

option forwardfor ###后端服务器可以从Http Header中获得客户端ip

option redispatch ###serverid服务器挂掉后强制定向到其他健康服务器

timeout connect 10000 #如果backend没有指定，默认为10s

timeout client 300000 ###客户端连接超时

timeout server 300000 ###服务器连接超时

maxconn 60000 ###最大连接数

retries 3 ###3次连接失败就认为服务不可用，也可以通过后面设置

listen stats

bind 0.0.0.0:1080 #监听端口

stats refresh 30s #统计页面自动刷新时间

stats uri /stats #统计页面url

stats realm Haproxy Manager #统计页面密码框上提示文本

stats auth admin:admin #统计页面用户名和密码设置

#stats hide-version #隐藏统计页面上HAProxy的版本信息

listen websrv-rewrite 0.0.0.0:80

balance roundrobin

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5

server web2 192.168.2.200:80 check inter 2000 rise 2 fall 5

参数解析

global /\*\* 全局设置 \*\*/

maxconn=2000 [全局并发量]

defaults /\*\* 默认设置 \*\*/

maxconn=1000 [默认并发量]

listen集群 /\*\* 单集群设置 \*\*/

maxconn=500 [单集群并发]

1. 两种方式定义集群:

frontend+backend

listen

listen xxx \*:80

server ip1

server ip2

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5 /\*\* 健康检查:5次算失败,每20s再次检查,成功两次后放回集群 \*\*/

listen status 0.0.0.0:1080

stats refresh 30s

stats uri /stats

stats realm HAProxy Manager

stats auth admin:admin

帮助文档

# ls /usr/share/doc/haproxy-1.5.18/configuration.txt

Day04 Ceph概述 部署Ceph集群 Ceph块存储

1. Ceph概述
2. 基础知识
3. 什么是分布式文件系统

· 分布式文件系统(Distributed File System)是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

· 分布式文件系统的设计基于客户机/服务器模式

/\*\*

\* 常见分布式网站架构

\* 1.用户访问调度器

\* 2.后台有一组应用服务器，调度器将请求转发给服务器

\* 3.服务器下有各自的页面文件/目录

\* 4.可以在后台做一台共享存储NFS/Sumba，服务器mount此目录

\* 5.当文件过大，NFS无法支持的时候，网站会没法继续做下去

\* 6.还有一台数据库为应用服务器提供数据

\* 7.数据库和共享存储统称为数据端

\*\*/

1. 常用分布式文件系统

· Lustre

· Hadoop

· FastDES

· Ceph

· GlusterFS

1. 什么是Ceph

· Ceph是一个分布式文件系统

· 具有高扩展、高可用、高性能的特点

· Ceph可以提供对象存储、块存储、文件系统存储

/\*\*

\* 对象存储，也叫做基于对象的存储，是用来描述解决和处理离散单元的方法的通用术语，这些离散单元被称作为对象。

\* 就像文件一样，对象包含数据，但是和文件不同的是，对象在一个层结构中不会再有层级结构。每个对象都在一个被称作存储池的扁平地址空间的同一级别里，一个对象不会属于另一个对象的下一级。

\* 文件和对象都有与它们所包含的数据相关的元数据，但是对象是以扩展元数据为特征的。每个对象都被分配一个唯一的标识符，允许一个服务器或者最终用户来检索对象，而不必知道数据的物理地址。这种方法对于在云计算环境中自动化和简化数据存储有帮助。

\*\*/

· Ceph可以提供PB级别的存储空间(PB>TB>GB)

- 1024G\*1024G=1048576G

· 软件定义存储(Software Defined Storage)作为存储行业的一大发展趋势，已经越来越上受到市场的认可

· 帮助文档：<http://docs.ceph.org/start/intro>

1. Ceph组件

· OSDs

- 存储设备

· Monitors

- 集群监控组件

- 至少三台启动，否则集群报错

· RGW

- 对象存储网关

· MDSs

- 存放文件系统的元数据(对象存储和块存储不需要该组件)

· Client

- ceph客户端

iscsi[块]

NFS[文件系统]

· Ceph提供：块，文件系统，对象存储(百度云)

- ceph-osd ceph-mon

\* 必装，用户只能以块的方式访问

- ceph-mds

\* 可以以文件系统方式访问，mount挂载

- ceph-radosgw (RGW，radosgateway)

\* 可以以对象存储的方式访问

/\*\*

\* 分布式存储的算法

\* 以取余算法为例：

\* node1:10T node2:10T node3:10T node4:10T 4台计算机为后端存储OSD，提供真正的存储磁盘

\* Monitor[监控/管理]：算法的执行者，用户访问数据前需要先访问Monitor

\* 用户访问a.txt|md5sum=16进制数%4=1 ==> 存入node2

\* b.txt|md5sum=16进制数%4=0 ==> 存入node1

\* Ceph用的是CRUSH算法

\*\*/

/\*\*

\* 分布式存储的特点

\* 1.数据被分开存储

\* 2.数据的读写是并行的

\* 3.数据会像raid一样，单个文件会被打散

\* 4.Ceph所有数据是3副本

\*\*/

1. 实验环境准备
2. 安装前准备
3. 物理机为所有节点配置yum源服务器

[root@room9pc01 ~]# yum -y install vsftpd

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount -o loop \

rhcs2.0-rhosp9-20161113-x86\_64.iso /var/ftp/ceph

[root@room9pc01 ~]# systemctl restart vsftpd

1. 修改所有节点都需要配置YUM源（这里仅以node1为例）

[root@node1 ~]# cat /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/MON

gpgcheck=0

[osd]

name=osd

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/OSD

gpgcheck=0

[tools]

name=tools

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/Tools

gpgcheck=0

1. 修改/etc/hosts并同步到所有主机

警告：/etc/hosts解析的域名必须与本机主机名一致！！！！

[root@node1 ~]# cat /etc/hosts

... ...

192.168.4.10 client

192.168.4.11     node1

192.168.4.12     node2

192.168.4.13     node3

1. 配置无密码连接(包括自己远程自己也不需要密码)

[root@node1 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

[root@node1 ~]# for i in 10 11 12 13

> do

> ssh-copy-id 192.168.4.$i

> done

1. 配置NTP时间同步
2. 真实物理机创建NTP服务器

[root@room9pc01 ~]# yum -y install chrony

[root@client ~]# cat /etc/chrony.conf

server 0.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

[root@room9pc01 ~]# systemctl restart chronyd

如果有防火墙规则，需要清空所有规则

[root@room9pc01 ~]# iptables -F

1. 其他所有节点与NTP服务器同步时间（以node1为例）

[root@node1 ~]# cat /etc/chrony.conf

server 192.168.4.254 iburst

[root@node1 ~]# systemctl restart chronyd

1. 准备存储磁盘
2. 物理机上为每个虚拟机准备3块磁盘。（可以使用命令，也可以使用图形直接添加）

[root@room9pc01 ~]# virt-manager

1. 部署Ceph集群
2. 准备部署环境
3. 安装部署软件

· 使用node1作为部署主机

# yum -y install ceph-deploy

· ceph-deploy命令与子命令都支持--help查看帮助

# ceph-deploy --help

1. 创建目录

· 为部署工具创建目录，存放秘钥与配置文件

# mkdir ceph-cluster

# cd ceph-cluster/

1. 部署存储集群
2. 创建Ceph集群
3. 创建Ceph集群配置(所有节点都为mon)

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3

1. 给所有节点安装Ceph软件包

[root@node1 ceph-cluster]# ceph-deploy install node1 node2 node3

1. 初始化所有节点的mon服务(主机名解析必须对)

[root@node1 ceph-cluster]# ceph-deploy mon create-initial

/\*\* 这里没有指定主机，是因为第一步创建的配置文件中已经有了，所以要求主机名解析必须对，否则连接不到对应的主机 \*\*/

/\*\*

\* 常见错误和解决方法：

\* [node1][ERROR ] admin\_socket: exception getting command descriptions: [Error 2] No such file or directory

\* 先检查自己的命令是否是在ceph-cluster目录下执行的！！！！如果时确认是在该目录下执行的create-initial命令，依然保存，可以使用如下方式修复

\* [root@node1 ceph-cluster]# vim ceph.conf #文件最后追加以下内容

public\_network = 192.168.4.0/24

\* 修改后重新推送配置文件:

\* [root@node1 ceph-cluster]# ceph-deploy --overwrite-conf config push node1 node2 node3

\*\*/

1. 初始化完成后，使用ceph -s与systemctl查看状态

# systemctl status ceph-<tab>

[ceph-create-keys@node1.service](mailto:ceph-create-keys@node1.service) ceph-mon.target

ceph-mds.target ceph-osd.target

ceph-mon@node1.service ceph-radosgw.target

1. 创建OSD
2. 给所有节点准备磁盘分区

[root@node1 ~]# parted /dev/vdb mklabel gpt

[root@node1 ~]# parted /dev/vdb mkpart primary 1M 50%

[root@node1 ~]# parted /dev/vdb mkpart primary 50% 100%

[root@node1 ~]# lsblk

[root@node1 ~]# chown ceph.ceph /dev/vdb1

[root@node1 ~]# chown ceph.ceph /dev/vdb2

[root@node1 ceph-cluster]# vim /etc/udev/rules.d/70-vdb.rules

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="ceph"

/\*\*

\* 我们都爱for循环

\* for i in node{1..3}; do

ssh $i 'parted /dev/vdb mklabel gpt;

parted /dev/vdb mkpart primary 1M 50%;

parted /dev/vdb mkpart primary 50% 100%;

chown ceph.ceph /dev/vdb1;

chown ceph.ceph /dev/vdb2';

ssh $i '

cat > /etc/udev/rules.d/70-vdb.rules << EOF

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="ceph"

EOF'; done

\*\*/

1. 初始化清空磁盘数据

[root@node1 ceph-cluster]# ceph-deploy disk zap node1:vdc node1:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node2:vdc node2:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node3:vdc node3:vdd

/\*\*

\* 我们都爱for循环

\* [root@node1 ceph-cluster]# for i in node{1..3};do ceph-deploy disk zap $i:vd{c,d};done

\*\*/

1. 创建OSD存储空间

[root@node1 ceph-cluster]# ceph-deploy osd create \

> node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph-deploy osd create \

> node2:vdc:/dev/vdb1 node2:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph-deploy osd create \

> node3:vdc:/dev/vdb1 node3:vdd:/dev/vdb2

//创建osd存储设备，vdc为集群提供存储空间，vdb1提供JOURNAL缓存，

//一个存储设备对应一个缓存设备，缓存需要SSD，不需要很大

/\*\*

\* [root@node1 ceph-cluster]# for i in node{1..3};do ceph-deploy osd create $i:vdc:/dev/vdb1 $i:vdd:/dev/vdb2;done

\*\*/

/\*\*

\* vdb1和vdb2用来做存储服务器的日志journal盘

\* vdb1做vdc的缓存，vdb2做vdd缓存

\* 缓存盘对速度的要求较高，因此最好是用固态硬盘，而ceph做在传统磁盘上

\*\*/

1. 验证

· 查看集群状态

[root@node1 ~]# ceph -s

/\*\*

\* 常见错误：

health: HEALTH\_WARN

clock skew detected on node2, node3…

表示时间不同步，将所有主机时间同步即可

Ceph要求所有主机时间差不能超过0.05s,否则就会提示WARN，如果使用NTP还不能精确同步时间，可以手动修改所有主机的ceph.conf，在[MON]下面添加如下一行：

mon clock drift allowed = 1

如果状态还是失败，可以尝试执行如下命令，重启ceph服务：

[root@node1 ~]# systemctl restart ceph\\*.service ceph\\*.target

\* 错误2：

权限配置完毕而ceph -s 显示EOR，可能是因为磁盘信息没有刷新

# partprobe

状态根据ceph主机数量和存储容量大小来确定

HEALTH状态会有延迟

\*\*/

1. Ceph块存储
2. 概述
3. 什么是块存储

· 单机块设备

- 光盘

- 磁盘

· 分布式块存储

- Ceph

- Cinder

· Ceph块设备也叫做RADOS块设备

- RADOS block device：RBD

· RDB驱动已经很好的集成在了Linux内核中

· RBD提供了企业功能，如快照、COW克隆等

· RBD还支持内存缓存，从而能够大大提高性能

· Linux内核可用直接访问Ceph块存储

· KVM可用借助于librbd访问



图例1

1. 块存储集群
2. 创建镜像

· 查看存储池(默认有一个rbd池)

[root@node1 ~]# ceph osd lspools

· 创建镜像、查看镜像

[root@node1 ~]# rbd create demo-image --image-feature layering --size 10G

[root@node1 ~]# rbd create rbd/image --image-feature layering --size 10G

/\*\* image-feature支持哪些功能; layering 支持分层快照功能 \*\*/

/\*\* 帮助文档 rbd help create \*\*/

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info demo-image

1. 动态调整大小

· 缩小容量

[root@node1 ~]# rbd resize --size 7G image --allow-shrink

[root@node1 ~]# rbd info image

· 扩容容量

[root@node1 ~]# rbd resize --size 15G image

[root@node1 ~]# rbd info image

/\*\*

\* 三台ceph-mon 三台ceph-osd

\* rbd create image 创建镜像

\*\*/

1. 集群内通过KRBD访问

· 集群内将镜像映射为本地磁盘

[root@node1 ~]# rbd map demo-image

/dev/rbd0

[root@node1 ~]# lsblk

… …

rbd0 251:0 0 10G 0 disk

[root@node1 ~]# mkfs.xfs /dev/rbd0

[root@node1 ~]# mount /dev/rbd0 /mnt

1. 客户端通过KRBD访问
2. 客户端需要安装ceph-common软件包
3. 拷贝配置文件(否则不知道集群在哪)
4. 拷贝连接密钥(否则无连接权限)

[root@client ~]# yum install ceph-common -y

[root@node1 ceph]# scp ceph.conf client:/etc/ceph/

[root@node1 ceph]# scp ceph.client.admin.keyring client:/etc/ceph/

1. 将镜像映射为本地磁盘

[root@client ~]# rbd map image

[root@client ~]# lsblk

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

1. 客户端格式化、挂载分区

[root@client ~]# mkfs.xfs /dev/rbd0

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# echo "test" > /mnt/test.txt

1. 创建镜像快照

/\*\* 快照 COW(Copy on Write)写时复制 \*\*/

/\*\* 快照使用写时复制技术，对大数据快照速度会很快

\* 原始盘100G

\* a.txt:111--XXX 1G

\* b.txt:222

\* 快照备份50G

\* a.txt:111

\* b.txt:222

\* 10T数据库 先快照 --> 后期再cp和tar

\* Ceph支持快照，但不支持在线快照，在线快照会使Ceph盘崩溃

\*\*/

1. 查看镜像快照

[root@node1 ~]# rbd snap ls image

1. 创建镜像快照

[root@node1 ~]# rbd snap create image --snap image-snap1

[root@node1 ~]# rbd snap ls image

SNAPID NAME SIZE

4 image-snap1 15360 MB

1. 删除客户端写入的测试文件

[root@client ~]# rm -rf /mnt/test.txt

1. 使用快照恢复数据
2. 还原快照

[root@node1 ~]# rbd snap rollback image --snap image-snap1

1. 客户端重新挂载分区

[root@client ~]# umount /mnt

[root@client ~]# mount /dev/rbd0 /mnt/

/\*\* 在线还原会导致坏块，此时挂载会报错，需要重新回滚一次 \*\*/

[root@node1 ~]# rbd snap rollback image --snap image-snap1

Rolling back to snapshot: 100% complete...done.

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# ls /mnt

1. 快照克隆

· 如果想从快照恢复出来一个新的镜像，则可以使用克隆

· 注意，克隆前，需要对快照进行<保护>操作

· 被保护的快照无法删除，取消保护(unprotect)

[root@node1 ~]# rbd snap protect image --snap image-snap1

[root@node1 ~]# rbd snap rm image --snap image-snap1 //会失败

[root@node1 ~]# rbd clone image \

--snap image-snap1 image-clone --image-feature layering

//使用image的快照image-snap1克隆一个新的image-clone镜像

· 查看克隆镜像与父镜像快照的关系

[root@node1 ~]# rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d36a3d1b58ba

format: 2

features: layering

flags:

parent: rbd/image@image-snap1

overlap: 15360 MB

· 克隆镜像很多数据都来自于快照链

· 如果希望克隆镜像可以独立工作，就需要将父快照中的数据全部拷贝一份，比较耗时

[root@node1 ~]# rbd flatten image-clone

Image flatten: 100% complete...done.

[root@node1 ~]# rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d36a3d1b58ba

format: 2

features: layering

flags:

//注意，父快照信息没了

1. 客户端撤销磁盘映射
2. umount挂载点

[root@client /]# umount /mnt/

1. 取消RBD磁盘映射

[root@client /]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

/\*\*

\* 语法格式：

\* rbd unmap /dev/rbd/{poolname}/{imagename}

\*\*/

[root@client /]# rbd unmap /dev/rbd/rbd/image

[root@client /]# rbd showmapped

1. 删除快照与镜像

· 删除快照

[root@node1 ~]# rbd snap rm image --snap image-snap1

rbd: snapshot 'image-snap1' is protected from removal.

2019-03-08 09:48:09.492277 7fd031781d80 -1 librbd::Operations: snapshot is protected

[root@node1 ~]# rbd snap unprotect image --snap image-snap1

[root@node1 ~]# rbd snap rm image --snap image-snap1

· 删除镜像

[root@node1 ~]# rbd list

demo-image

image

image-clone

[root@node1 ~]# rbd rm image

Removing image: 100% complete...done.

[root@node1 ~]# rbd list

Day05 块存储应用案例 分布式文件系统 对象存储

1. 块存储应用案例
2. 准备实验环境
3. 创建磁盘镜像

KVM:

/var/lib/libvirt/images/镜像文件

/etc/libvirt/qemu/XML文件

cp xx.qcow2 new.qcow2

cp xx.xml new.xml

# cd /var/lib/libvirt/images/

# qemu-img create -f qcow2 -b .rh7\_template.img mycp.img 10G

# cp .rhel7.xml /etc/libvirt/qemu/rhel7.xml

# vim /etc/libvirt/qemu/rhel7.xml

<name>mycp</name>

<source file='/var/lib/libvirt/images/mycp.img'/>

# virsh define /etc/libvirt/qemu/rhel7.xml

# virsh start mycp

1. 为虚拟机创建磁盘镜像

[root@node1 ~]# rbd create vm1-image --image-feature layering --size 10G

1. 查看镜像

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info vm1-image

[root@node1 ~]# qemu-img info rbd:rbd/vm1-image

image: rbd:rbd/vm1-image

file format: raw

virtual size: 10G (10737418240 bytes)

disk size: unavailable

1. Ceph认证账户

· Ceph默认开启用户认证，客户端需要账户才可以访问

- 默认账户名称为client.admin，key是账户的密钥

- 可以使用ceph auth添加新账户(案例中使用默认账户)

[root@node1 ~]# cat /etc/ceph/ceph.conf //配置文件

[global]

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.2.10,192.168.2.20,192.168.2.30

auth\_cluster\_required = cephx //开启认证

auth\_service\_required = cephx //开启认证

auth\_client\_required = cephx //开启认证

[root@node1 ~]# cat /etc/ceph/ceph.client.admin.keyring //账户文件

[client.admin]

key = AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

1. 部署客户端环境

· 注意：这里使用物理机当客户端

· 客户端需要安装ceph-common软件包

· 拷贝配置文件，否则不知道集群在哪

· 拷贝连接密钥，否则无连接权限

[root@yyzh\_pc ~]# yum install ceph-common -y

[root@yyzh\_pc ~]# ssh node1

[root@node1 ~]# scp /etc/ceph/ceph.conf /etc/ceph/ceph.client.admin.keyring 192.168.4.254:/etc/ceph/

[root@node1 ~]# exit

1. 创建KVM虚拟机
2. 创建初始化虚拟机

· 使用virt-manager创建2台普通的KVM虚拟机

- 这里以1个虚拟机为例

1. 配置libvirt secret

· KVm虚拟机需要使用librbd才可以访问ceph集群

· Librbd访问ceph需要账户认证

· 所以这里需要给libvirt设置账户信息



原理图示

1. 编写账户信息文件(物理机操作)

[root@room9pc01 ~]# vim secret.xml //新建临时文件，内容如下

<secret ephemeral='no' private='no'>

<usage type='ceph'>

<name>client.admin secret</name>

</usage>

</secret>

[root@yyzh\_pc ~]# ceph auth get-key client.admin

AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

[root@yyzh\_pc ~]# cat /etc/ceph/ceph.client.admin.keyring

[client.admin]

key = AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

1. 使用XML配置文件创建secret

[root@room9pc01 ~]# virsh secret-define --file secret.xml

生成 secret 9877ff83-2ca6-48c1-982b-50a1def1bd45

//随机的UUID，这个UUID对应的有账户信息

1. 设置secret，添加账户密钥

[root@yyzh\_pc ~]# virsh secret-set-value \

> --secret 9877ff83-2ca6-48c1-982b-50a1def1bd45 \

> --base64 AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

secret 值设定

//这里secret后面是之前创建的secret的UUID

//base64后面是client.admin账户的密码

//现在secret中既有账户信息又有密钥信息

1. 虚拟机的XML配置文件

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/mycp.img'/>

<target dev='vda' bus='virtio'/>

<boot order='1'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</disk>

1. 修改XML配置文件

# virsh edit mycp

<disk type='network' device='disk'>

<driver name='qemu' type='raw'/>

<auth username='admin'>

<secret type='ceph' uuid='9877ff83-2ca6-48c1-982b-50a1def1bd45'/>

</auth>

<source protocol='rbd' name='rbd/vm1-image'>

<host name='192.168.4.11' port='6789'/>

</source>

<target dev='vda' bus='virtio'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</disk>

/\*\* 硬盘接口可以是virtio,ide,sata盘中的任意一种，不可以是SCSI和USB \*\*/

· 关键词说明

<secret type='ceph' uuid='9877ff83-2ca6-48c1-982b-50a1def1bd45'/>

//这里的uuid就是secret的uuid，有client.admin账户和密语额信息

<source protocol='rbd' name='rbd/vm1-image'>

<host name='192.168.4.11' port='6789'/>

</source>

//这里说明使用账户连接哪台ceph主机和端口，访问哪个池和镜像

<target dev='vda' bus='virtio' />

//这里将获取的镜像设置为虚拟机的vda磁盘

1. 分布式文件系统
2. 基础知识
3. 什么是CephFS

/\*\* 什么是文件系统？NTFS,fat32,ext4,xfs文件系统 \*\*/

/\*\*

\* 文件系统格式化过程：

\* 1.inode索引+block

\* 2.4K对齐

\*

\* 索引中记录了权限，大小等以及对应文件的指针

\*\*/

· 分布式文件系统(Distributed Fiile System)是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

· CephFS使用Ceph集群提供与POSIX兼容的文件系统

· 允许Linux直接将Ceph存储mount到本地

1. 环境准备

· 环境拓扑



拓扑图例

1. 准备一台新的虚拟机，作为GW服务器
2. 要求如下：

- IP地址：192.168.4.14

- 主机名：node4

- 配置yum源(包括rhel、ceph的源)

- 与Client主机同步时间

- node1允许无密码远程node4

- 修改node1的/etc/hosts，并同步到所有node主机

1. 元数据服务器
2. 什么是元数据

· 元数据(Metadata)

- 任何文件系统中的数据分为数据和元数据

- 数据是指普通文件中的实际依据

- 而元数据指用来描述一个文件的特征的系统数据

- 比如：访问权限、文件拥有者以及文件数据块的分步信息(inode...)等

- 所以CephFS必须有MDSs节点

1. 部署元数据服务器
2. 登录node4，安装ceph-mds软件包

[root@node4 ~]# yum install ceph-mds -y

1. 登录node1部署节点操作

[root@node1 ~]# cd /root/ceph-cluster

//该目录是最早部署ceph集群时，创建的目录

[root@node1 ceph-cluster]# ceph-deploy mds create node4

//给node4拷贝配置文件，启动mds服务

1. 同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node4

1. 实际操作

实际操作：

[root@node1 ceph-cluster]# ceph-deploy mds create node3

/\*\*

\* 实际生产环境中推荐新建一台计算机提高性能

\* 实验中mds做在了node3上

\*\*/

1. 文件系统服务器
2. 创建存储池

[root@node3 ~]# ceph osd pool create cephfs\_data 128

pool 'cephfs\_data' created

[root@node3 ~]# ceph osd pool create cephfs\_metadata 128

pool 'cephfs\_metadata' created

//创建两个存储池，对应128个PG

1. 创建Ceph文件系统

[root@node3 ~]# ceph fs new myfs1 cephfs\_metadata cephfs\_data

new fs with metadata pool 2 and data pool 1

//metadata作inode，data作block

//注意先写metadata再写data

//ceph默认只能创建1个文件系统，多余的会报错

1. 查看创建状态

[root@node3 ~]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node3 ~]# ceph mds stat

e5: 1/1/1 up {0=node3=up:active}

1. 客户端挂载

[root@client ~]# mount -t ceph 192.168.4.11:6789:/ /media/ -o name=admin,secret=AQAcuoBcgyPWFxAAyOblGtEQpKdRK/VaB85zkA==

//注意：文件系统类型为ceph

//192.168.4.11为MON节点的IP(不是MDS节点)

//admin是用户名，secret是密钥

//密钥可以在/etc/ceph.ceph.client.admin.keyring中找到

1. 对象存储
2. 概述

/\*\*

\* ceph 支持的存储方案：

\* 1.块(磁盘) osd

\* 2.文件系统 mds

\* 3.对象存储 rgw(radosgateway)

\*

\* 对象存储必须开发人员写程序

\*\*/

1. 什么是对象存储

· 对象存储

- 也就是键值存储，通过其接口指令，也就是简单的GET、PUT、DEL和其他扩展，向存储服务器上传下载数据

- 对象存储中所有数据都被认为是一个对象，所以，任何数据都可以存入对象存储服务器，如图片、视频、音频

· RGW全程是Rados Gateway

· RGW是Ceph对象存储网关，用于向客户端应用呈现存储界面，提供RESTful API访问接口

· RGW可以部署多台，拓扑如下：



RGW拓扑

1. 环境准备

· 准备一台新的虚拟机，作为元数据服务器

· 要求如下：

- IP地址：192.168.4.15

- 主机名：node5

- 配置yum源(包括rhel、ceph的源)

- 与Client主机同步时间

- node1允许无密码远程node5

- 修改node1的/etc/hosts，并同步到所有node主机

1. 对象存储
2. 部署RGW软件包

[root@node1 ~]# ceph-deploy install --rgw node5

/\*\* 同步配置文件与密钥到node5 \*\*/

[root@node1 ~]# cd /root/ceph-cluster

[root@node1 ~]# ceph-deploy admin node5

1. 新建网关实例

· 启动一个rgw服务

[root@node1 ~]# ceph-deploy rgw create node5

· 登录node3验证服务是否启动

[root@node3 ~]# ps aux |grep radosgw

root 8848 0.4 2.3 1073560 11788 ? Ssl 17:07 0:00 /usr/bin/radosgw -f --cluster ceph --name client.rgw.node3 --setuser ceph --setgroup ceph

[root@node5 ~]# systemctl status ceph-radosgw@\\*

1. 修改服务端口

· 登陆node3，RGW默认服务端口为7480，修改为8000或80更方便客户端记忆和使用

[root@node5 ~]# vim /etc/ceph/ceph.conf

[client.rgw.node5]

host = node5

rgw\_frontends = "civetweb port=8000"

//node5为主机名

//civetweb是RGW内置的一个web服务

1. 客户端测试
2. curl测试

[root@client ~]# curl 192.168.4.15:8000

<?xml version="1.0" encoding="UTF-8"?>

<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">

<Owner>

<ID>anonymous</ID>

<DisplayName></DisplayName>

</Owner>

<Buckets></Buckets>

</ListAllMyBucketsResult>

1. 使用第三方软件访问

· 登录node3(RGW)创建账户

[root@node3 ~]# radosgw-admin user create \

> --uid="testuser" --display-name="First User"

{

"user\_id": "testuser",

"display\_name": "First User",

"email": "",

"suspended": 0,

"max\_buckets": 1000,

"auid": 0,

"subusers": [],

"keys": [

{

"user": "testuser",

"access\_key": "JUB7AVZB2JHS5YHO67LF",

"secret\_key": "lRW5PJSPaVRekA9CH9W8Fg1gezIYsOi3yvE5rQBB"

}

],

"swift\_keys": [],

"caps": [],

"op\_mask": "read, write, delete",

"default\_placement": "",

"placement\_tags": [],

"bucket\_quota": {

"enabled": false,

"max\_size\_kb": -1,

"max\_objects": -1

},

"user\_quota": {

"enabled": false,

"max\_size\_kb": -1,

"max\_objects": -1

},

"temp\_url\_keys": []

}

[root@node5 ~]# radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

1. 使用第三方软件访问
2. 客户端安装软件

[root@client ~]# yum install s3cmd-2.0.1-1.el7.noarch.rpm

1. 修改软件配置(注意，除了下面设置的内容，其他提示都默认回车)

[root@client ~]# s3cmd --configure

Access Key: JUB7AVZB2JHS5YHO67LF  
 Secret Key: lRW5PJSPaVRekA9CH9W8Fg1gezIYsOi3yvE5rQBB

S3 Endpoint [s3.amazonaws.com]: 192.168.4.13:8000

[%(bucket)s.s3.amazonaws.com]: %(bucket)s.192.168.4.13:8000

Use HTTPS protocol [Yes]: No

Test access with supplied credentials? [Y/n] n

Save settings? [y/N] y

//注意，其他提示都默认回车

1. 创建存储数据的bucket(类似于存储数据的目录)

[root@client ~]# s3cmd ls

[root@client ~]# s3cmd mb s3://my\_bucket

Bucket 's3://my\_bucket/' created

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd put /var/log/messages s3://my\_bucket/log/

[root@client ~]# s3cmd ls

2019-03-08 10:56 s3://my\_bucket

[root@client rpm]# s3cmd ls s3://my\_bucket/log

DIR s3://my\_bucket/log/

[root@client ~]# s3cmd ls s3://my\_bucket/log/

2019-03-08 10:59 1318215 s3://my\_bucket/log/messages

1. 测试下载功能

[root@client ~]# s3cmd get s3://my\_bucket/log/messages /tmp/

[root@client rpm]# ls /tmp/messages

/tmp/messages

1. 测试删除功能

[root@client ~]# s3cmd del s3://my\_bucket/log/messages

[root@client rpm]# s3cmd ls s3://my\_bucket/log/messages

[root@client rpm]# s3cmd ls s3://my\_bucket/log/

[root@client rpm]# s3cmd ls s3://my\_bucket/

1. 总结和答疑