## CLUSTER

Day1

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

存储:根据不同的应用环境将数据保存到某些介质上并能保证有效的访问。

SCSI小型计算机系统接口，作为输入/输出接口，主要用于硬盘、光盘等设备

DAS直连接式存储，将存储设备通过SCSI接口或光纤通道直接连接到计算机上

NAS网络技术存储，专用数据存储服务器，存储设备与服务器分离，通过TCP/IP协议访问

SAN光纤通道，通过光纤交换机、光纤路由器等将磁盘阵列等存储设备与服务器连接起来

ISCSI技术，将SCSI数据块映射为以太网数据包，基于IP Storage理论的存储技术

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

存储设备服务端

ISCSI Target

#yum -y install targetcli

#yum info targetcli //查看iscsi target信息

配置ISCSI target

#targetcli

//定义后端存储

/>backstores/block create iscsi\_store /dev/vdb1

//创建iqn对象

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

//授权客户机访问

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

//绑定存储

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

//绑定监听地址

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals/ create 0.0.0.0(默认已自动创建)

//保存配置

/>saveconfig

/>exit

服务管理

#systemctl {start|restart|enable|stop|status} target

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

客户端

ISCSI Initiator，软件实现，成本低、性能较低

ISCSI HBA， 硬件实现，性能好，成本低

安装软件

#yum repolist

#yum -y install iscsi-initiator-utils

//查看信息

#yum info iscsi-initiator-utils

//设置本机iqn名称

#vim /etc/iscsi/initiatorname.iscsi

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

//发现远程target存储(必须执行，否则发现不了硬盘)

#iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.1 --discover

//登陆target

#systemctl restart iscsi

#systemctl enable iscsi

//分区格式化

#parted /dev/sda

(parted) mklabel gpt

(parted) mkpart primary 1M 100%

(parted) quit

# mkfs.xfs /dev/sda1

//挂载

mount /var/lib/msyql /dev/vda1

#blkid /dev/sda1

#vim /etc/fstab

UUID="402d1304-5781-4349-8c43-ba45eba8b427" /var/lib/mysql xfs,defaults,\_netdev 0 0

//不能多台设备同时挂载共享存储。因ext4/xfs是单节点文件系统，如果多个节点同时挂载，就会损坏文件系统，使得数据丢失。红帽GFS才能支持多节点同时挂载

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

Udev设备管理

devfs

--linux早期采用的静态管理方法

--/dev目录下有大量静态文件

--内核版本2.6.13开始被完全取代

udev

--动态管理硬件文件的方法

--只有连到系统上的设备才在/dev下创建设备文件

--与主、次设备编号无关

--为设备提供持久、一致的名字

接入设备事件链

内核发现设备并导入设备状态到sysfs

Udev连接到事件通知

Udev创建设备节点或是运行指定程序

Udev通知hald守护进程

HAL探测设备信息

HAL创建设备对象结构

HAL通过系统消息总线广播该事件

用户程序也可以监控该事件

udev的作用

从内核收到添加/移除硬件事件时，udev将会分析：

-/sys目录下信息

-/etc/udev/rules.d目录中的规则

基于分析结果，udev会：

-处理设备命名

-决定要创建哪些设备文件或链接

-决定如何设置属性

-决定出发哪些事件

配置udev

主配置文件/etc/udev/udev.conf

-udev\_root:创建设备文件位置，默认为/dev

-udev\_rules: udev规则文件位置，默认为/etc/udev/rules.d

-udev\_log:syslog优先级，缺省为err

文件位置及格式/etc/udev/rules.d/<rule\_name>.rules

-例：75-custom.rules（数字大小顺序时规则文件执行的顺序）

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

Udev规则文件介绍

规则文件必须以 .rules 作为扩展名，否则不被当作规则文件。

规则文件的每一行都是 key=value 格式。 key 有两个类型：

1）匹配型 key 2）赋值型 key

当所有匹配型 key 都匹配时，该规则即被采用，赋值型 key 就会获得相应的值。当规则匹配时，可以重命名网络接口， 创建到设备节点的符号链或运行一个指定程序来处理该事件。

一条规则由多个key=value 组成，以英文逗号隔开。 每个 key 有一个操作，取决于操作符，有效的操作符如下：

== 比较是否相等

!= 比较是否不相等

= 给一个key 赋值。 表示一个列表的key会被重置，并且把这个唯一的值传给它

+= 将一个值增加到key中

:= 将一个值传给一个key，并且不允许再修改这个key。

1>匹配型KEY

下面的key可以匹配设备属性，部分key也可用于匹配sysfs中的父设备属性，不仅仅是产生事件的那个设备。

如果在一个规则中，有多个key匹配了一个父设备，则这些 key 必须匹配同一个父设备：

ACTION 匹配事件的动作名

DEVPATH 匹配事件的设备devpath

KERNEL 匹配事件的设备名

NAME 匹配网络接口或者设备节点的名字。只有在前面的规则赋值之后才可使用。

SYMLINK 匹配设备节点符号链的名字。SYMLINK 只有在前面的规则赋值之后才可以使用。可以有 多个symlinks，只需要匹配一个。

SUBSYSTEM 匹配设备子系统

DRIVER 匹配设备的驱动名。只对绑定到一个驱动的设备有用。

ATTR { filename } 匹配事件设备的 sysfs 属性。

KERNELS 向上搜索devpath，直到找到一个匹配的设备名

SUBSYSTEMS 向上搜索devpath，直到找到一个匹配的子系统名

DRIVERS 向上搜索devpath，直到找到一个匹配的驱动名

ATTRS{ filename } 向上搜索devpath，直到找到一个含匹配 sysfs 属性的设备ENV{ key }

TAG 设备的 tag

TEST{octal mode mask} 测试一个文件是否存在，可以指定一个8进制的模式掩码。

PROGRAM 执行一个程序。如果程序成功返回， key 为 true。设备的属性被放在被执行进

程的环境变量中，该程序的输出为 stdout， 可以从 RESULT 这个 key 读取。

RESULT 匹配最近一次 PROGRAM 调用的返回字符串。它应该在 PROGRAM 之后使用

支持一些shell的通配符：\* (代表[ 0 个到无穷多个 ]任意字符

? (代表[一定有一个]任意字符)

[ ] (代表[一定有一个在括号内]的字符(非任意字符))

2>赋值型KEY

下面的key 是赋值型 key：

NAME

SYMLINK

OWNER， GROUP， MODE

ATTR { key }

ENV { key }

TAG

RUN

LABEL GOTO 可以跳到的地方

GOTO 跳到下一个带有匹配名字的 LABEL 处。

IMPORT { type }

WAIT\_FOR

OPTIONS

NAME、SYMLINK、PROGRAM、OWNER、GROUP、MODE和RUN这些field支持一个简单的、类似于 printf 函数的格式字符串替换。可用的字符替换如下：

$kernel, %k : 该设备的内核名字(%k 替换 $kernel)

$number, %n：该设备的内核号码。例如 sda3 的内核号码是 3。

$devpath, %p：该设备的 devpath

$id, %b：当向上搜索devpath，寻找SUBSYSTEMS，KERNELS，DRIVERS和ATTRS时，被匹配的设备名字

$driver：当向上搜索devpath，寻找SUBSYSTEMS，KERNELS，DRIVERS和ATTRS时，被匹配的驱动名字

$attr {file}, %s{file}：一个被发现的设备的sysfs属性的值。如果该设备没有该属性，且前面的 KERNELS, SUBSYSTEMS, DRIVERS或 ATTRS 测试选择的是一个父设备，那么就用父设备的属性。如果属性是一个符号链，符号链的最后一个元素作为返回值。

$env { key }, %E { key }： 一个设备属性值

$major, %M： 该设备的内核主号码

$minor, %m： 该设备的内核次号码

$result, %c： 由 PROGRAM 调用的外部程序返回的字符串。如果这个字符串包含空格，可以用 %c{N} 选中第N个字段。如果这个数字N，后面有一个 + 字符， 则表示选中从这个字段开始的所有后面的字符串 %c { N + }

$parent, %p：父设备的节点名字

$name：设备节点的名字，用一个空格作为分隔符。该值只有在前面的规则赋值之后才存在，或者是remove事件。

$links：当前符号链的列表，用空格隔开。该值只有在前面的规则赋值之后才存在，或者是remove事件。

$root, %r：udev\_root 的值

$sys, %S：sysfs 挂载点

$tempnode, %N：在真正的设备节点创建之前，创建的一个临时的设备节点的名字，这个临时设备节点供外部程序使用。

$$： '$'字符自己

%%： '%' 字符自己

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

Udev应用

给iscsi磁盘创建一个软链接叫idisk

1、查看iscsi磁盘（sda）在/sys/目录中的位置

# udevadm info --query=path --name=/dev/sda1

2、通过上一步的路径，查看iscsi磁盘的信息（以和其他设备进行区别）

# udevadm info --query=all --attribute-walk

--path=/devices/platform/host2/session1/target2:0:0/2:0:0:0/block/sda/sda1

3、创建规则文件

# vim /etc/udev/rules.d/90-iscsi.rules

KERNEL=="sd[a-z]\*",ACTION=="add",SUBSYSTEMS=="scsi",ATTRS{model}=="formysql", SYMLINK+="idisk%n"

注：KERNEL==””表示内核识别出来的设备名

ACTION=＝“add”表示新接入设备

SUBSYSTEMS和ATTRS{model}是第（2）步查到的

SYMLINK表示创建符号链接，+=表示额外创建，%n是分区号

<<<只有把磁盘移除再接入才能生效或是重启系统>>>

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

NFS

实现网络共享的，用于unix-like（类unix）系统间的共享。端口号是2049，基于RPC（远程过程调用，端口号111）服务。NFS只提供了共享功能，底层数据传输交给RPC服务。

1、服务端

（1）安装软件包

# yum install -y nfs-utils

（2）创建共享目录

# mkdir -pv /nfsroot/nfsro

# cp /etc/hosts /nfsroot/nfsro

（3）修改配置文件

# vim /etc/exports

/nfsroot/nfsro \*(ro) ->允许所有地址以只读方式访问

（4）启动服务并验证

# systemctl start nfs

# showmount -e 192.168.4.1

2、客户端

（1）创建挂载点

# mkdir /mnt/nsfshare

（2）查看共享，并挂载

# showmount -e 192.168.4.1

# mount 192.168.4.1:/nfsroot/nfsro /mnt/nsfshare

# ls /mnt/nsfshare

如果是读写共享，必须注意本地权限和配置文件内的授权

NFS选项

(1)no\_root\_squash表示当root创建文件时，保留文件的属主属组还是root，默认写入到共享目录中的文件属主属组是nfsnobody

/nfsroot/nfsrw 192.168.4.\*(rw,sync,no\_root\_squash)

1. all\_squash：作用是客户端任何用户写入的文件属主属组都是nfsnobody。

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

Multipath多路径

用途

如果客户端到服务器存储只有一条线路，那么该线路出现故障，存储就不可用了。可以再加一条线路。这个时候，客户端就会从两条线路发现同一存储设备，会给这个存储起两个名字，如sda和sdb。

客户端无论使用哪个存储都不合适。我们可以创建一个虚拟磁盘，如mpatha，客户端挂载mpatha，只要底层不要两条路径全部出现故障，mpatha一直可用。原理参考HSRP

主要功能

冗余：主备模式，高可用

改进性能：主主模式，负载均衡

多路径配置

1、在vh01和vh03之间加上192.168.2.0/24网络

2、在vh03上通过192.168.2.0网络也发现一次共享

[root@vh03 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.1 --discover

[root@vh03 ~]# systemctl restart iscsi

[root@vh03 ~]# lsblk 此时本地有了sda和sdb

3、安装多路径软件包

[root@vh03 ~]# yum install -y device-mapper-multipath

4、sda和sdb是同一设备，他们的wwid就是相同的

获得磁盘的wwid：

[root@vh03 ~]# /lib/udev/scsi\_id --whitelisted --device=/dev/sda

3600140589282e00db104fc29d84334a8

5、生成多路径配置文件

[root@vh03 ~]# mpathconf --user\_friendly\_names n

6、在配置文件里声明获得wwid的方法，并且声明WWID是3600140589282e00db104fc29d84334a8的设备就是mpatha

[root@vh03 ~]# vim /etc/multipath.conf

(文件不存在，模版文件为/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf)

defaults {

user\_friendly\_names no

find\_multipaths yes

}

multipaths {

multipath {

wwid "3600140589282e00db104fc29d84334a8"

alias "mpatha"

}

}

7、启动多路径服务，就可以在/dev/mapper下看到mpatha设备了

[root@vh03 ~]# systemctl start multipathd

[root@vh03 ~]# systemctl enable multipathd

验证

#ls /dev/mapper //mpatha为多路径设备

#multipath -rr //重新加载多路径信息

#multipath -ll //查看多路径信息

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

集群及LVS

一组通过高速网络互联的计算组，并以单一系统的模式加以管理，将很多服务器集中起来一起，提供同一种服务，在客户端看来就像是只有一个服务器。

可以在付出较低成本的情况下获得在性能、可靠性、灵活性方面的相对较高的收益

任务调度是集群系统中的核心技术

需要IPVS模块支持，默认内核已经加载

集群的目的

提高性能，如计算密集型应用，如天气预报、核试验模型

降低成本，相对百万美元级的超级计算机，价格便宜

提高可扩展性，只要增加集群节点即可

增强可靠性，多个节点完成相同功能，避免单点失败

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

LVS（linux虚拟服务器）

可以实现高可用的、可伸缩的web、mail、cache和media等网络服务

最终目标是利用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，数据共享存储空间的存储区域

LVS术语

Director Server调度服务器，将负载分发到real server服务器

Real Server真实服务器，真正提供应用服务的服务器

VIP虚拟IP地址，分布给用户访问的虚拟IP地址

RIP真实IP地址，集群节点上使用的IP地址

DIP调度器连接节点服务器的IP地址

LVS工作模式

VS/NAT：通过网络地址转换实现的虚拟服务器，大并发访问时调度器的性能成为瓶颈

VS/DR：直接使用路由集数实现，节点服务器需要配置VIP，注意MAC地址广播

VS/TUN：通过隧道方式实现虚拟服务器

负载均衡调度算法

轮询（Round Robin）客户端请求平均分发到Real Server

加权轮询（Weighted Round Robin）根据Real Server权重进行轮询调度

最少连接（Least Connections）选择连接数量最少的服务器

加权最少连接（Weighted 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

基于局部性的最少连接

带复制的基于局部性最少链接

目标地址散列（Destination Hashing）

最短的期望的延迟

最少队列调度

-----------------------------------------------------------------------------------------------------------------------LVS-NAT集群

客户端访问director（分配器)(VIP)时，director通过网络地址转换，调度器重写请求报文的目标地址，根据预设的调度算法。将请求分派给后端的真实服务器；真实服务器的响应报文通过调度器时，报文的源地址被重写，再返回给客户机，完成整个负载均衡调度过程，但因为服务器恢复请求全经过director，会造成director巨大负载。

(分配器：公网IP和私有IP(后端网关), real server：私有IP)

1>环境确认

#grep -i ‘ipvs’ /boot/config-3.10.0-327.el7.x86\_64

#IPVS transport protocol load balancing support

#IPVS scheduler

#IPVS SH scheduler

#IPVS application helper

//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

部署流程

1>Real Sever

配置WEB服务器

#yum -y install httpd

echo ‘real server’ > /var/www/html/index.html

#systemctl enable httpd

#systemctl restart httpd

配置网关指向分配器服务器地址192.168.4.4

#nmcli connection modify eth0 ipv4.gateway 192.168.4.4

#nmcli connection up eht0

2>Director Server

vh04添加另一个IP地址（作为虚拟服务器地址）

eth0:192.168.4.4 eth2:201.1.1.4

安装并启用ipvsadm

#yum -y install ipvsadm

#systemctl start ipvsadm

(无法启动保存配置即可，ipvsadm-save > /etc/sysconfig/ipvsadm)

#systemctl enable ipvsadm

创建虚拟服务器

#ipvsadm -A -t VIP地址:端口 -s 调度算法

#ipvsadm -A -t 201.1.1.4:80 -s rr

//-A 添加虚拟服务器

//-t 设置群集地址（VIP，Virtual IP）

//-s 指定负载调度算法

向虚拟服务器中加入节点

#ipvsadm -a -t VIP地址:端口 -r real地址 -m -w 权重

#ipvsadm -a -t 201.1.1.4:80 -r 192.168.7.21:80 -m

从虚拟服务器中删除节点

#ipvsadm -a -t VIP地址:端口 -r real地址 -m -w 权重

#ipvsadm -d -r 201.1.1.4:80 -t 172.16.16.172:80

//-a 添加真实服务器

//-d 删除真实服务器

//-r 指定真实服务器（real server）的地址

//-m 使用NAT模式；-g、-i分别对应DR、TUN模式

//-W 为节点服务器设置权重，默认为1`

查看配置

#ipvsadm -Ln //查看IPVS

保存配置

#ipvsadm-save > /etc/sysconfig/ipvsadm

//LVS调度器需要打开IP路由转发(RHEL7默认已经打开)

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

#vim /etc/sysctl.conf

net.ipv4.ip\_forward=1

3>Client

连接虚拟服务器测试

通过web浏览器访问

使用ab进行大并发测试

#ab -c 10 -n 1000 http://vip地址/index.html

在Director上查看连接数

#ipvsadm -L -n --rate

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

Commands: Either long or short options are allowed.

--add-service -A add virtual service with options

--edit-service -E edit virtual service with options

--delete-service -D delete virtual service

--clear -C clear the whole table

--restore -R restore rules from stdin

--save -S save rules to stdout

--add-server -a add real server with options

--edit-server -e edit real server with options

--delete-server -d delete real server

--list -L|-l list the table

--zero -Z zero counters in a service or all services

--set tcp tcpfin udp set connection timeout values

--start-daemon start connection sync daemon

--stop-daemon stop connection sync daemon

--help -h display this help message

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

LVS-DR集群

director(分配器)分配请求到real server。real server 处理请求后直接回应给用户，director负载均衡器仅处理客户机与服务器一半连接。从而避免性能瓶颈，同样增加系统可伸缩性。Direct Routing由于采用物理层（修改MAC地址）技术，因此所有服务器都必须处理同一物理网段。

一、拓扑：LVS调度器只有一个IP地址，它和real server在同一网络。

二、客户机要把数据发给VIP。VIP需要出现在每台服务器上（调度器、web服务器）

三、为了地址不冲突，需要把VIP配置在调度器的eth0上，把VIP配置在web服务器的lo上

四、每台主机都有VIP，客户端发来的请求，只有LVS回应。为了实现这一点，需要改web服务器的内核参数

五、实施

1、仍然使用vh04作为调度器，对它进行清理

[root@vh04 ~]# ifdown eth2

[root@vh04 ~]# ipvsadm -D -t 201.1.1.4:80

2、在vh04上配置vip

[root@vh04 ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0{,:0}

[root@vh04 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.100

PREFIX=24

[root@vh04 ~]# ifup eth0:0

3、在web服务器的lo上配置VIP

[root@vh02 ~]# cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

[root@vh02 ~]# vim /etc/sysconfig/network-scripts/ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.100

NETMASK=255.255.255.255

NETWORK=192.168.4.100

BROADCAST=192.168.4.100

ONBOOT=yes

NAME=lo:0

[root@vh02 ~]# ifup lo:0

4、在web服务器上修改内核参数

[root@vh02 ~]# sysctl -a | grep arp\_ig

[root@vh02 ~]# echo "net.ipv4.conf.all.arp\_ignore = 1" >> /etc/sysctl.conf

[root@vh02 ~]# echo "net.ipv4.conf.lo.arp\_ignore = 1" >> /etc/sysctl.conf

[root@vh02 ~]# sysctl -a | grep arp\_ann

[root@vh02 ~]# echo "net.ipv4.conf.all.arp\_announce = 2" >> /etc/sysctl.conf

[root@vh02 ~]# echo "net.ipv4.conf.lo.arp\_announce = 2" >> /etc/sysctl.conf

[root@vh02 ~]#sysctl -p

5、配置规则

[root@vh04 ~]# ipvsadm -A -t 192.168.4.100:80 -s lc

[root@vh04 ~]# ipvsadm -a -t 192.168.4.100:80 -r 192.168.4.3

[root@vh04 ~]# ipvsadm -a -t 192.168.4.100:80 -r 192.168.4.2 -g

6、ipvsadm服务

如果启动ipvsadm服务时出错，则

[root@vh04 ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

然后再正常控制服务状态

[root@vh04 ~]# systemctl restart ipvsadm

[root@vh04 ~]# systemctl enable ipvsadm

访问验证

curl 192.168.4.100

-----------------------------------------------------------------------------------------------------------------------HAProxy服务器

工作原理

Proxy代理,客户端把请求发送到HAProxy后，HAProxy替用户发送请求到web服务器，web服务器响应HAProxy，把页面发给HAProxy。HAProxy再把页面发给客户端。

提供高可用、负载均衡以及基于TCP和HTTP应用的代理

负载均衡器：nginx/haproxy/lvs/F5

代理：

正向代理：帮助客户端缓存服务器上的数据

反向代理：帮助服务器缓存数据

衡量负载均衡性能因素

Session rate 会画率 每秒钟产生的会话数

Session concurrency 并发会话数 服务器处理会话的时间越长，并发会话数越多

Date rate 数据速率 以MB/s或Mbps衡量，大的对象导致并发会话数增加，高会话 数、高数据速率要求更多的内存

HAProxy工作模式

mode http，客户端请求被深度分析后再发往服务器

mode tcp，客户端与服务器之间建立会话，不检查第七层信息

mode health，仅做健康状态检查，已经不建议使用

HTTP协议解析

当HAProxy运行在HTTP模式下，HTTP请求（Request）和响应（Reponse）均被完全分析和索引，这样便于创建恰当的匹配guize

理解HTTP请求和响应，对于更好的创建匹配规则至关重要

HTTP事务模型

HTT P协议是事务驱动的

每一个请求（Request）仅能对应一个响应（Response）

常见模型：

HTTP close

客户端向服务器建立一个TCP连接

客户端发送请求给服务器

服务器响应客户端请求后即断开连接

如果客户端到服务器的请求不只一个，那么要不断建立连接

TCP三次握手消耗相对较大的系统投资源，同时延迟较大

Keep-alive

一次连接可传递多个请求

客户端需要知道传输内容的长度，以避免无期限的等带传输结束

降低两个http事务间的延迟

需要相对较少的服务器资源

Pipelining

仍然使用Keep-alive

在发送后续请求前，不管等前面的求求已经得到回应

适用于有大量图片的页面

降低了多次请求只见的网络延迟

HTTP头部信息

包含许多有关的客户端环境和请求正文的有用信息，如浏览器使用的语言、请求正文的长度等

HAProxy配置实例

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

1、安装

[root@vh04 bin]# yum install -y haproxy

2、修改配置文件

[root@vh04 bin]# vim /etc/haproxy/haproxy.cfg

//把# main frontend which proxys to the backends后面部分全部删除，增加以下内容：定义一个监控页面

listen stats

bind 0.0.0.0:1080 # 监控页面的端口号

stats refresh 30s # 页面自动刷新时间是30秒

stats uri /monitor # 页面网址

stats realm HaManager

stats auth admin:admin # 用户名：密码

listen myweb 0.0.0.0:80

cookie SERVERID rewrite

balance roundrobin # 调度算是rr

server web1 192.168.4.2:80 cookie a1i1 check inter 2000 rise 2 fall 5

#每2000ms检查一次服务器，2次成功是好的，5次失败表示故障

server web2 192.168.4.3:80 cookie a1i2 check inter 2000 rise 2 fall 5

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

配置文件/etc/haproxy/haproxy.cfg构成

default 为后续的其他部分设置缺省参数，缺省参数可以被后续部分重置

frontend 描述接收客户端侦听套接字（socket）集

backend 描述转发链接的服务器集

listen 把frontend和backend结合到一起的完整声明

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

3、启服务

[root@vh04 bin]# systemctl start haproxy

1. 访问http://192.168.4.4可以实现负载均衡轮询调度，

访问http://192.168.4.4:1080/monitor可以看到监控页面

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

日志构成

# man 5 rsyslog.conf

Facility.priority 相当于是服务.优先级

Facility可以是这些关键字：auth, authpriv, cron, daemon, kern, lpr, mail, mark, news,

security，syslog, user,uucp以及local0到local7

Local0到local7是预留出来的接口，供第三方应用调用。

Priority可以使用的关键字：debug, info,notice, warning, warn，err, error，crit,

alert,emerg, panic

Debug是最不严重的级别，panic是最严重的级别。如果日志记录优先级是info，表示

比info严重的日志都需要记录。

配置haproxy日志

1、配置本机接受通过网络发来的日志

[root@node4 ~]# vim /etc/rsyslog.conf

# Provides UDP syslog reception

$ModLoad imudp

$UDPServerRun 514

# Provides TCP syslog reception

$ModLoad imtcp

$InputTCPServerRun 514

[root@node4 ~]# systemctl restart rsyslog

查看日志

[root@node4 ~]# tail -f /var/log/messages

访问haproxy调度器，可以看到日志信息。

=======================================================================Keepalived热备份

VRRP：虚拟冗余路由协议，IETF公共标准

HSRP：热备份路由协议，，思科私有

运行原理

Keepalived检测每一个服务器节点状态，服务器节点异常或工作出现故障，Keepalive将故障节点从集群系统中剔除。故障节点恢复候，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

部署

软件安装

#yum -y install keepalived

配置文件/etc/keepalived/keepalived.conf

# vim /etc/keepalived/keepalived.conf

# vrrp\_strict

vrrp\_instance VI\_1 {

state MASTER #辅助写BACKUP

interface eth0

virtual\_router\_id 51 #虚拟路由器ID号

priority 150 #优先级

advert\_int 1 # 心跳消息发送间隔

authentication { # 集群成员共享密码

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.200 # vip

}

}

Heartbeat：心跳

[root@node2 ~]# systemctl start keepalived

4、查看、验证

[root@node2 ~]# ip address show eth0

双主配置：两台主机，两个服务，每个主机是一个服务的主

-----------------------------------------------------------------------------------------------------------------------Keepalived+LVS

高可用调度器

使用Keepalived高可用解决调度器单点失败问题，主、备调度器上配置LVS，主调度器异常时，Keepalived启用备用调度器

搭建高可用、负载均衡的web集群

1、新建node5虚拟机，作为额外的lvs服务器。虽然有两台调度器，也是DR模式。

2、在web服务器的lo上配置VIP

3、调整web服务器的内核参数

4、调度器上不要再手工的为eth0配置VIP了。因为VIP出现在活跃的调度器上，活跃设备由keepalived决定。应该把已经存在的调度器VIP移除。

[root@node4 ~]# ifdown eth0:0

[root@node4 ~]# mv /etc/sysconfig/network-scripts/ifcfg-eth0:0 ~

5、把node4上的haproxy停掉

[root@node4 ~]# systemctl stop haproxy

6、将node4上的lvs规则清掉。因为lvs规则将由keepalived配置文件指定。

[root@node4 ~]# ipvsadm -D -t 192.168.4.100:80

7、在node5上安装ipvsadm

[root@node5 ~]# yum install -y ipvsadm

8、在node4/5安装keeaplived

[root@node4 ~]# yum install -y keepalived

9、在node4上配置keepalived

[root@node4 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

root@localhost # 收件人

}

notification\_email\_from admin@tedu.cn # 发件人

smtp\_server 127.0.0.1 # 邮件服务器地址

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.100

}

}

virtual\_server 192.168.4.100 80 {

delay\_loop 6

lb\_algo rr

lb\_kind DR

persistence\_timeout 50

protocol TCP

real\_server 192.168.4.2 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.3 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

10、启动服务验证

[root@node4 ~]# systemctl restart keepalived.service

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

集群调度软件对比

nginx分析

优点：

工作在7层，可以针对http做分流策～

正则表达式比HAProxy强大

安装、配置、测试间大，通过日志可以解决多数问题

并发量可以达到几万次

Nginx还可以作为Web服务器使用

缺点：

仅支持http、https、mail协议，应用面小

监控检查仅通过端口，无法使用url检查

LVS分析

优点

负载能力强，工作在4层，对内存、CPU消耗低

配置性低，没有大多可配置性，减少人为错误

应用面广，几乎可以为所有应用提供负载均衡

缺点

不支持正则表达式，不能实现动静分离

如果网站架构庞大，LVS-DR配置比较繁琐

HAProxy分析

优点

支持session、cookie功能

可以通过url进行健康检查

效率、负载均衡速度，高于nginx、低于LVS

HAProxy支持TCP，可以对Mysql进行负载均衡

调度算法丰富

缺点

正则弱于nginx

日志依赖于syslogd，不支持apache日志

-----------------------------------------------------------------------------------------------------------------------分布式文件系统

分布式文件系统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

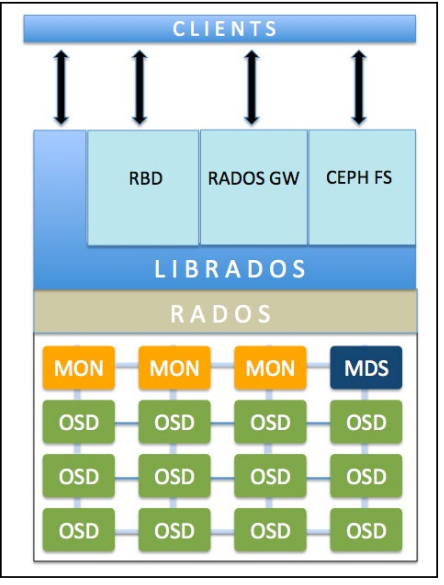
存储：

RAID：独立磁盘冗余阵列

性能、容错、空间

分布式存储：CEPH

组成



1. MON：监视器。MON通过保存一系列集群状态map来监视集群的组件。MON因为保存集群状态，要防止单点故障，所以需要多台；另外，MON需要是奇数，如果出现意见分岐，采用投票机制，少数服从多数。
2. OSD：对象存储设备。真正存储数据的组件。一般来说，每块参与存储的磁盘都需要一个OSD进程。
3. （3）MDS：元数据服务器。只有CephFS需要它。

元数据：metadata，存储数据的数据。比如一本书内容是数据，那么书的作者、出版社、出版时间之类的信息就是元数据。

1. RADOS：可靠自主分布式对象存储。它是ceph存储的基础，保证一切都以对象形式存储。
2. RBD：RADOS块设备，提供块存储
3. CephFS：提供文件系统级别存储
4. RGW：RADOS网关，提供对象存储

存储分类：

块存储：提供硬盘，如iSCSI

文件级别存储：共享文件夹

对象存储：一切皆对象

http://storage.ctocio.com.cn/281/12110781\_2.shtml

CEPH环境准备

1. 准备6台虚拟机

主机名、IP地址

1. 在物理主机上配置名称解析

[root@room8pc16 nsd2018]# for i in {1..6}

> do

> echo -e "192.168.4.$i\tnode$i.tedu.cn\tnode$i" >> /etc/hosts

> done

3、提前将服务器的密钥保存，不需要ssh时回答yes

[root@room8pc16 nsd2018]# ssh-keyscan node{1..6} >> /root/.ssh/known\_hosts

4、实现免密登陆

[root@room8pc16 nsd2018]# for i in {1..6}

> do

> ssh-copy-id node$i

> done

5、配置yum源

[root@room8pc16 nsd2018]# mkdir /var/ftp/ceph/

[root@room8pc16 nsd2018]# vim /etc/fstab

/ISO/rhcs2.0-rhosp9-20161113-x86\_64.iso /var/ftp/ceph iso9660 defaults 0 0

[root@room8pc16 nsd2018]# mount -a

[root@room8pc16 nsd2018]# vim /tmp/server.repo

[rhel7.4]

name=rhel7.4

baseurl=ftp://192.168.4.254/rhel7.4

enabled=1

gpgcheck=0

[mon]

name=mon

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

enabled=1

gpgcheck=0

[osd]

name=osd

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

enabled=1

gpgcheck=0

[tools]

name=tools

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

enabled=1

gpgcheck=0

[root@room8pc16 nsd2018]# for vm in node{1..6}

> do

> scp /tmp/server.repo ${vm}:/etc/yum.repos.d/

> done

1. 配置node1节点为管理节点
2. 配置名称解析

[root@node1 ~]# for i in {1..6}; do echo -e "192.168.4.$i\tnode$i.tedu.cn\tnode$i" >> /etc/hosts; done

1. 配置免密登陆

[root@node1 ~]# ssh-keyscan node{1..6} >> /root/.ssh/known\_hosts

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

[root@node1 ~]# for i in {1..6}; do ssh-copy-id node$i; done

[root@node1 ~]# for vm in node{1..6}

> do

> scp /etc/hosts ${vm}:/etc/

> done

7、NTP网络时间协议，基于UDP123端口。用于时间同步

时区：地球一圈360度，经度每15度角一个时区，共24个时区。以英国格林威治这个城市所在纵切面为基准。北京在东八区。

夏季节约时间：夏令时。DST

Stratum：时间服务器的层级。

时间准确度：原子钟。

8、配置node6为时间服务器

（1）配置

[root@node6 ~]# yum install -y chrony

[root@node6 ~]# vim /etc/chrony.conf

server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

1. 启动服务

[root@node6 ~]# systemctl enable chronyd

[root@node6 ~]# systemctl restart chronyd

将node1-5配置为NTP的客户端

（1）配置

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

#server 0.rhel.pool.ntp.org iburst

#server 1.rhel.pool.ntp.org iburst

#server 2.rhel.pool.ntp.org iburst

#server 3.rhel.pool.ntp.org iburst

server 192.168.4.6 iburst

[root@node1 ~]# systemctl restart chronyd

1. 测试

[root@node1 ~]# date -s "2018-7-13 12:00:00"

[root@node1 ~]# ntpdate 192.168.4.6

[root@node1 ~]# date

（3）同步其他主机

[root@node1 ~]# for i in {2..5}

> do

> scp /etc/chrony.conf node$i:/etc/

> done

[root@node1 ~]# for vm in node{2..5}

> do

> ssh $vm systemctl restart chronyd

> done

9、为node1-3各添加3块10GB的磁盘

可以在虚拟机不关机的情况下，直接添加硬盘

安装ceph

1. 在node1上安装部署软件

[root@node1 ~]# yum install -y ceph-deploy

1. 创建ceph部署工具的工作目录

[root@node1 ~]# mkdir ceph-clu

1. 创建参与集群节点的配置文件

[root@node1 ceph-clu]# ceph-deploy new node{1..3}

[root@node1 ceph-clu]# ls

4、在3个节点上安装软件包

[root@node1 ceph-clu]# ceph-deploy install node{1..3}

1. 初始化mon服务

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

如果出现以下错误：

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

解决方案：

[root@node1 ceph-clu]# vim ceph.conf 最下面加入行：

public\_network = 192.168.4.0/24

再执行以下命令：

[root@host1 ceoh-clu]# ceph-deploy --overwrite-conf config push node1 node2 node3

1. 把node1-3的vdb作为日志盘。Ext／xfs都是日志文件系统，一个分区分成日志区和数据区。为了更好的性能，vdb专门作为vdc和vdd的日志盘。

[root@node1 ceph-clu]# for vm in node{1..3}

> do

> ssh $vm parted /dev/vdb mklabel gpt

> done

[root@node1 ceph-clu]# for vm in node{1..3}; do ssh $vm parted /dev/vdb mkpart primary 1M 50% ; done

[root@node1 ceph-clu]# for vm in node{1..3}; do ssh $vm parted /dev/vdb mkpart primary 50% 100% ; done

[root@node1 ceph-clu]# for vm in node{1..3}; do ssh ${vm} chown ceph.ceph /dev/vdb? ; done

1. 创建OSD设备

[root@node1 ceph-clu]# for i in {1..3}

> do

> ceph-deploy disk zap node$i:vdc node$i:vdd

> done

[root@node1 ceph-clu]# for i in {1..3}

> do

> ceph-deploy osd create node$i:vdc:/dev/vdb1 node$i:vdd:/dev/vdb2

> done

1. 验证

到第7步为止，ceph已经搭建完成。查看ceph状态

[root@node1 ceph-clu]# ceph -s 如果出现health HEALTH\_OK表示正常

1. 排错

https://www.zybuluo.com/dyj2017/note/920621

CEPH应用

1. 块存储：使用最多的一种方式
2. cephFS：了解，不建议在生产环境中使用，因为还不成熟
3. 对象存储：了解，使用亚马逊的s3

使用RBD(Rados块设备)

1. 查看存储池

[root@node1 ~]# ceph osd lspools

可以查看到0号镜像池，名字为rbd

1. 创建名为demo-img的镜像大小为10GB

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

[root@node1 ~]# rbd list

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

3、创建第2个镜像，名为image，指定它位于rbd池中

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